

TreecEva (1331)

CAOYE

TreecEva (1331)

问题背景

评估体系建立

种子任务构建 (114 SEED)

1 - 语句级推理 [Statement-Level] (31)

1A - 算数运算 [Arithmetic] (6)

1B - 布尔运算 [Boolean] (5)

1C - API/函数调用 [API/Function Call] (7)

1D - 变量赋值 [Assignment] (8)

1E - 常量赋值 [Constant] (5)

1F - 大混合 [mix] (2)

2 - 代码块级推理 [Block-Level] (24)

2A - 线性代码块 [Linear] (6)

2B - 条件代码块 [Conditional] (8)

2C - 迭代代码块 [Iterative] (8)

2D - 大混合 (2)

3 - 代码属性推理 [Property-Level] (27)

3A - 循环属性 [Loop] (6)

3B - 分支属性 [Branch] (7)

3C - 内存属性 [Memory] (6)

3D - 作用域属性 [Scope] (6)

3E - 大混合 (2)

4 - 跨函数推理 [MultiFunc-Level] (32)

4A - 函数调用链推理 [Call Chain] (8)

4B - 参数传递推理 [Parameter Passing] (6)

4C - 状态传播推理 [State Propagation] (8)

4D - 副作用推理 [Side Effect] (8)

4E - 大混合 (2)

变式生成叶子任务构建

1 - 语句级推理 [Statement-Level] (323)

1A - 算数运算 [Arithmetic] (49)

1B - 布尔运算 [Boolean] (50)

1C - API/函数调用 [API/Function Call] (57)

1D - 变量赋值 [Assignment] (80)

1E - 常量赋值 [Constant] (37)

1F - 大混合 (50)

2 - 代码块级推理 [Block-Level] (326)

2A - 线性代码块 [Linear] (72)

2B - 条件代码块 [Conditional] (96)

2C - 迭代代码块 [Iterative] (115)

2D - 大混合 (43)

3 - 代码属性推理 (283)

3A - 循环属性 [Loop] (52)

3B - 分支属性 [Branch] (62)

3C - 内存属性 [Memory] (59)

3D - 作用域属性 [Scope] (59)

3E - 大混合 (51)

4 - 跨函数推理 [MultiFunc-Level] (285)

- 4A - 函数调用链推理 [Call Chain] (64)
- 4B - 参数传递推理 [Parameter Passing] (57)
- 4C - 状态传播推理 [State Propagation] (64)
- 4D - 副作用推理 [Side Effect] (59)
- 4E - 大混合 (41)

问题背景

| CodeSense论文结论回顾 | |
|-----------------|--|
| 数据来源 | 744个真实世界GitHub项目 |
| 数据集规模 | Python 2125 C 876 JAVA 875 每个测试数据 3-516 行 |
| 评估任务分层 | Task 1: 代码块级语义 (Block-level) [预测代码块的执行结果, 输入到输出 or 输出到输入] Task 2: 语句级语义 (Statement-level) [判断某个语句执行后影响, 算数、布尔、第三方库调用、变量赋值、常量赋值] Task 3: 代码属性 (Code Properties) [循环属性、指针属性、分支属性] Task 4: 语义近似 (Approximation) [抽象值预测] |
| 独特之处 | 真实的项目数据、细粒度评估、多编程语言、数目多 |
| 缺陷 | 【复杂性干扰】: 真实项目包含大量依赖和无关路径, 难以精确定位模型失败的根本原因, 也难以有学习价值 【不均衡】: 可以看到每个task的分配很不均衡, 以及语言分布也很不均衡 |
| INSIGHT | 基于这种评估任务分层的设计与真实世界代码的例子, 设计出一颗评估树测试集——TreecEva。思路为: 根据分层的逻辑, 将代码评估任务分为几类, 每类再细分到子类, 最终对每个子类给出真实世界代码案例, 再人工去设计此案例的其他语言, 其他情况, 极限的边界情况的变式, 最终一步一步地构建出完备的数据集它有如下优势 |

| TreecEva思路展开——具体构建流 | |
|---------------------|--|
| 【阶段1】评估树结构设计 | 1.1. 主干设计 ——> 1.2. 分支设计 |
| 【阶段2】种子任务收集与分析 | 2.1 CodeSense中寻找种子, 稍作改造 |
| 【阶段3】变式生成策略 | 3.1 为每个分支设计变式生成 |
| 【阶段4】自动化生成流水线设计 | 4.1 结合分支种子代码与任务+变式生成范例4.2 设计mutiagent 生成结构 [数据集生成+数据集完备性检查+单数据质量检查]4.3 生成最终数据集 |
| 【阶段5】汇总与整理 | 5.1 将所有数据集按树形结构存储, 有完备编号, 并维护好树形结构5.2 进行实验验证数据集的训练能力与评估能力 |

评估体系建立

TreecEva

- 语句级推理 [Statement-Level]
 - 算数运算 [Arithmetic] (四则运算、高级运算、位运算、复合运算)
 - 布尔运算 [Boolean] (比较运算、逻辑运算、短路求值)
 - API/函数调用 [API/Function Call] (内置函数、数学库、字符串操作、容器操作)
 - 变量赋值 [Assignment] (简单赋值、多重赋值、解包赋值)
 - 大混合
- 代码块级推理 (Block-Level)
 - 线性代码块 [Linear] (顺序语句、独立语句、依赖语句)
 - 条件代码块 [Conditional] (if条件块、switch条件块、嵌套条件块)
 - 迭代代码块 [Iterative] (for循环块、While循环快、递归调用块)
 - 大混合
- 代码属性推理 (Property-Level)
 - 循环属性 [Loop] (迭代计数、变量追踪、终止条件)
 - 分支属性 [Branch] (条件求值、路径选择、分支效果)
 - 内存属性 [Memory] (引用关系、生命周期、访问模式)
 - 作用域属性 [Scope] (可见性、生存期、变量遮蔽)
 - 大混合
- 跨函数推理 (MultiFunc-Level)
 - 函数调用链推理 [Call Chain] (直接调用、链式调用、递归调用、回归调用)
 - 参数传递推理 [Parameter Passing] (值传递、引用传递)
 - 状态传播推理 [State Propagation] (全局变量、静态变量、闭包捕获、对象状态)
 - 副作用推理 [Side Effect] (I/O操作、异常处理、内存操作、时间依赖)
 - 大混合

种子任务构建 (114 SEED)

1 - 语句级推理 [Statement-Level] (31)

1A - 算数运算 [Arithmetic] (6)

```
{
  "id": "SL-AR-S001",
  "metadata": {
    "name": "StatementLevel-Arithmetic-Seed1",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "seed",
    "source": "CodeSense",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
}
```

```

"task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
    "code": "a = [1, 2, 4, 4, 5, 6, 7, 23, 8, 9, 20, 11, 13, 34, 66]\naux = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nlo = 0\nmid = 3\nhi = 7\ni = 1\nj = mid + 1\nfor k in range(lo, hi + 1):\n    aux[k] = a[k]\nfor k in range(lo, hi + 1):\n    if i > mid:\n        a[k] = aux[j]\n        j += 1\n    elif j > hi:\n        a[k] = aux[i]\n        i += 1\n    elif util.less(aux[i], aux[j]):\n        a[k] = aux[i]\n        i += 1\n    else:\n        a[k] = aux[j]\n        j += 1",
    "answer": 4
}
}

```

```

{
    "id": "SL-AR-S002",
    "metadata": {
        "name": "StatementLevel-Arithmetic-Seed2",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = total // count`?",
        "code": "items = [15, 25, 30, 20]\ntotal = sum(items)\ncount = len(items)\nresult = total // count\nremainder = total % count",
        "answer": 22
    }
}

```

```

{
    "id": "SL-AR-S003",
    "metadata": {
        "name": "StatementLevel-Arithmetic-Seed3",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** 2`?",
        "code": "import math\nradius = 5\n\ndiameter = 2 * radius\nncircumference = 2 * math.pi\n* radius\nnarea = 3.14159 * radius ** 2\n\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 78.53975
    }
}

```

```
    }
}
```

```
{
  "id": "SL-AR-S004",
  "metadata": {
    "name": "StatementLevel-Arithmetic-Seed4",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << position)`?",
    "code": "flags = 0b1010\nposition = 2\nmask = flags | (1 << position)\ncheck = mask & (1 << position)\ntoggle = flags ^ (1 << position)",
    "answer": 14
  }
}
```

```
{
  "id": "SL-AR-S005",
  "metadata": {
    "name": "StatementLevel-Arithmetic-Seed5",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset // 3 - power ** 2 % 5`?",
    "code": "base = 10\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset // 3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2 + offset // 3 - power ** 2 % 5",
    "answer": 21
  }
}
```

```
{
  "id": "SL-AR-S006",
  "metadata": {
    "name": "StatementLevel-Arithmetic-TypeConversion",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset // 3 - power ** 2 % 5`?",
    "code": "base = 10\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset // 3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2 + offset // 3 - power ** 2 % 5",
    "answer": 21
  }
}
```

```

    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight + bonus)`?",
    "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nfinal_score = int(average * weight + bonus)\nrounded_score = round(average * weight + bonus)",
    "answer": 75
}
}

```

1B - 布尔运算 [Boolean] (5)

```

{
    "id": "SL-BL-S001",
    "metadata": {
        "name": "StatementLevel-Boolean-Seed1",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "seed",
        "source": "CodeSense-libjpeg-turbo",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `inbuffer == NULL || insize == 0` when `inbuffer` points to a valid buffer and `insize` is 1024?",
        "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nif (inbuffer == NULL || insize == 0) {\n    printf(\"Input validation failed\\n\");\n} else {\n    printf(\"Input is valid\\n\");\n}",
        "answer": false
    }
}

```

```

{
    "id": "SL-BL-S002",
    "metadata": {
        "name": "StatementLevel-Boolean-Seed2",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "seed",
        "source": "CodeSense-cryptsetup",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    }
}

```

```
},
"task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `!data` when `data` is NULL?",
    "code": "void *data = NULL;\nsize_t size = 1024;\nif (!data) {\n    return;\n} else\n{\n    memset(data, 0, size);\n}",
    "answer": true
}
}
```

```
{
    "id": "SL-BL-S003",
    "metadata": {
        "name": "StatementLevel-Boolean-Seed3",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "seed",
        "source": "CodeSense-libdwarf",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res != DW_DLV_OK` when `res` is -1 and `DW_DLV_OK` is 0?",
        "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif\n(res != DW_DLV_OK) {\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n}",
        "answer": true
    }
}
```

```
{
    "id": "SL-BL-S004",
    "metadata": {
        "name": "StatementLevel-Boolean-Seed4",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "seed",
        "source": "CodeSense-krb5",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 65 (ASCII 'A')?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (c < 0x80) {\n    printf(\"ASCII character\");\n} else {\n    printf(\"Non-ASCII character\");\n}",
        "answer": true
    }
}
```

```

}

{
  "id": "SL-BL-S005",
  "metadata": {
    "name": "StatementLevel-Boolean-Seed5",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "seed",
    "source": "CodeSense-postfix",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR || state == IN_CHAR_SPACE` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
    "code": "#define IN_CHAR 1#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif (state == IN_CHAR || state == IN_CHAR_SPACE) {\n    return len;\n} else {\n    return 0;\n}",
    "answer": true
  }
}

```

1C - API/函数调用 [API/Function Call] (7)

```

{
  "id": "SL-API-S001",
  "metadata": {
    "name": "StatementLevel-APICall-Seed1",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "seed",
    "source": "CodeSense",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the return value of the function call `strlen(str)` when `str` is \"hello\"?",
    "code": "#include <string.h>\nchar *str = \"hello\";\nsize_t len =\nstrlen(str);\nprintf(\"Length: %zu\\n\", len);",
    "answer": 5
  }
}

```

```
{
  "id": "SL-API-S002",
  "metadata": {
```

```
        "name": "StatementLevel-APICall-Seed2",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "seed",
        "source": "CodeSense",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of the function call `pow(2.0, 3)` when called with base 2.0 and exponent 3?",
        "code": "#include <math.h>\ndouble base = 2.0;\nint exponent = 3;\ndouble result =\npow(base, exponent);\nprintf(\"Result: %.1f\\n\", result);",
        "answer": 8.0
    }
}
```

```
{
    "id": "SL-API-S003",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the exact number of bytes requested by the function call `malloc(sizeof(int) * 10)` assuming sizeof(int) is 4?",
        "code": "#include <stdlib.h>\\n// Assume sizeof(int) = 4 bytes\nint *arr = (int *)malloc(sizeof(int) * 10);\\nif (arr != NULL) {\\n    arr[0] = 42;\\n    printf(\"Memory allocated successfully\\n\");\\n}",
        "answer": 40
    }
}
```

```
{
    "id": "SL-API-S004",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "seed",
        "source": "CodeSense",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },

```

```
"task": {
    "description": "Given the following code snippet, what type of value does the function call `getpid()` return?",
    "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t current_pid = getpid();\nprintf(\"Process ID: %d\\n\", current_pid);",
    "answer": "pid_t"
}
}
```

```
{
    "id": "SL-API-S005",
    "metadata": {
        "name": "StatementLevel-APICall-Seed5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "seed",
        "source": "CodeSense",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `strcmp(str1, str2)` when `str1` is \"apple\" and `str2` is \"banana\"?",
        "code": "#include <string.h>\nchar *str1 = \"apple\";\nchar *str2 = \"banana\";\nint result = strcmp(str1, str2);\nprintf(\"Comparison result: %d\\n\", result);",
        "answer": -1
    }
}
```

```
{
    "id": "SL-API-S006",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `fopen(filename, \"r\")` when the file \"data.txt\" does NOT exist?",
        "code": "#include <stdio.h>\nchar *filename = \"nonexistent.txt\"; // This file does not exist\nFILE *fp = fopen(filename, \"r\");\nif (fp == NULL) {\n    printf(\"File could not be opened\\n\");\n} else {\n    printf(\"File opened successfully\\n\");\n}\nfclose(fp);",
        "answer": "NULL"
    }
}
```

```
}
```

```
{
  "id": "SL-API-S007",
  "metadata": {
    "name": "StatementLevel-APICall-Seed7",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "seed",
    "source": "CodeSense",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the return value of `snprintf(buffer, size, format, value)` when buffer size is 10, format is \"%d\", and value is 12345?",
    "code": "#include <stdio.h>\nchar buffer[10];\nint size = sizeof(buffer);\nconst\nchar *format = \"%d\";\nint value = 12345;\nint result = snprintf(buffer, size, format,\nvalue);\nprintf(\"Buffer: %s, Result: %d\\n\", buffer, result);",
    "answer": 5
  }
}
```

1D - 变量赋值 [Assignment] (8)

```
{
  "id": "SL-AS-S001",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed1",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "seed",
    "source": "CodeSense",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable x after executing the assignment statement `x = y`?",
    "code": "y = 42\nz = 100\nx = y\nprint(f\"x = {x}\")",
    "answer": 42
  }
}
```

```
{
  "id": "SL-AS-S002",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed2",
```

```
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "seed",
        "source": "CodeSense",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable a after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 10\nb = 20\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\na\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 20
    }
}
```

```
{
    "id": "SL-AS-S003",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable first after executing the unpacking assignment `first, second, *rest = numbers`?",
        "code": "numbers = [1, 2, 3, 4, 5]\n\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 1
    }
}
```

```
{
    "id": "SL-AS-S004",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "seed",
        "source": "CodeSense",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet, what is the value that *ptr points to after executing the assignment statement `ptr = &value`?",  
        "code": "int value = 100;\nint *ptr;\nptr = &value;\nprintf(\"Value: %d, Address: %p\\n\", *ptr, ptr);",  
        "answer": 100  
    }  
}
```

```
{  
    "id": "SL-AS-S005",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "seed",  
        "source": "Manual",  
        "language": "python",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet, what is the value of variable x after executing the chained assignment `x = y = z = 5`?",  
        "code": "x = 1\ny = 2\nz = 3\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = 5\nprint(f\"After: x={x}, y={y}, z={z}\")",  
        "answer": 5  
    }  
}
```

```
{  
    "id": "SL-AS-S006",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed6",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "seed",  
        "source": "Manual",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet, what is the value of variable count after executing the compound assignment statement `count += increment * 2`?",  
        "code": "count = 10\nincrement = 3\nmultiplier = 2\nprint(f\"Initial count: {count}\")\ncount += increment * 2\nprint(f\"Final count: {count}\")",  
        "answer": 16  
    }  
}
```

```
{
```

```

{
  "id": "SL-AS-S007",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed7",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = 10\nb = 20\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\nprint(f\"After swap: a={a}, b={b}\")",
    "answer": 10
  }
}

```

```

{
  "id": "SL-AS-S008",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed8",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = numbers`?",
    "code": "numbers = [1, 2, 3, 4, 5]\n\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={rest}\")",
    "answer": 3
  }
}

```

1E - 常量赋值 [Constant] (5)

```

{
  "id": "SL-CT-S001",
  "metadata": {
    "name": "StatementLevel-Constant-Seed1",
    "category": "Statement-Level",
    "subcategory": "Constant",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  }
}

```

```
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = 42`?",
        "code": "num = 42\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
        "answer": 42
    }
}
```

```
{
    "id": "SL-CT-S002",
    "metadata": {
        "name": "StatementLevel-Constant-Seed2",
        "category": "Statement-Level",
        "subcategory": "Constant",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value assigned to variable message in the statement `message = \"Hello, world!\"`?",
        "code": "message = \"Hello, World!\"\nprint(f\"Message: {message}\")\nprint(f\"Length: {len(message)}\")",
        "answer": "Hello, World!"
    }
}
```

```
{
    "id": "SL-CT-S003",
    "metadata": {
        "name": "StatementLevel-Constant-Seed3",
        "category": "Statement-Level",
        "subcategory": "Constant",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value assigned to variable is_valid in the statement `is_valid = True`?",
        "code": "is_valid = True\nprint(f\"Valid: {is_valid}\")\nprint(f\"Type: {type(is_valid)}\")",
        "answer": true
    }
}
```

```
}
```

```
{
  "id": "SL-CT-S004",
  "metadata": {
    "name": "StatementLevel-Constant-Seed4",
    "category": "Statement-Level",
    "subcategory": "Constant",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value assigned to variable pi in the statement `pi = 3.14159`?",
    "code": "pi = 3.14159\nradius = 5\narea = pi * radius ** 2\nprint(f\"Pi: {pi}\")\nprint(f\"Area: {area}\")",
    "answer": 3.14159
  }
}
```

```
{
```

```

  "id": "SL-CT-S005",
  "metadata": {
    "name": "StatementLevel-Constant-Seed5",
    "category": "Statement-Level",
    "subcategory": "Constant",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the integer value of the pointer status after executing the statement `status = NULL`?",
    "code": "#include <stdio.h>\nvoid *status = NULL;\nint value = 100;\nif (status ==\nNULL) {\n    printf(\"status is null\\n\");\n} else {\n    printf(\"status is not\nnull\\n\");\n}",
    "answer": 0
  }
}
```

1F - 大混合 [mix] (2)

```
{
  "id": "SL-MIX-S001",
  "metadata": {
    "name": "StatementLevel-Mix-Seed1",
```

```

    "category": "Statement-Level",
    "subcategory": "Mix",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "hard",
    "intervention": 2
},
"task": {
    "description": "Given the following complex code snippet involving arithmetic operations, boolean logic, API calls, variable assignments, and constants, what is the final value of variable result after executing all statements?",
    "code": "import math\nimport random\n\n# Constants initialization\nPI = 3.14159265359\nMAX_SIZE = 1024\nDEBUG_MODE = True\nERROR_CODE = -1\nSUCCESS_CODE = 0\n\n# Data structures initialization\nnumbers = [15, 42, 87, 23, 91, 56, 34, 78, 12, 65]\nweights = [0.1, 0.2, 0.15, 0.25, 0.3]\nconfig = {'threshold': 50, 'multiplier': 2.5, 'enabled': True}\nmatrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]\n\n# Variable assignments and arithmetic operations\nntotal_sum = sum(numbers)\naverage = total_sum / len(numbers)\nmax_value = max(numbers)\nmin_value = min(numbers)\nrange_value = max_value - min_value\n\n# Boolean operations and conditional logic\nis_above_threshold = average > config['threshold']\nis_enabled = config['enabled'] and DEBUG_MODE\nhas_valid_range = range_value > 0 and range_value < MAX_SIZE\n\n# API/Function calls with complex parameters\nnsquared_numbers = [x ** 2 for x in numbers if x > min_value]\nfiltered_sum = sum(nsquared_numbers)\nsqrt_result = math.sqrt(filtered_sum)\nlog_result = math.log10(sqrt_result) if sqrt_result > 0 else 0\n\n# Complex arithmetic with multiple operations\nweighted_score = sum(w * n for w, n in zip(weights, numbers[:len(weights)]))\nnormalized_score = weighted_score / sum(weights)\nbonus_multiplier = config['multiplier'] if is_above_threshold else 1.0\nfinal_score = normalized_score * bonus_multiplier\n\n# Bitwise operations\nflags = 0b1010\nposition = 3\nmask = flags | (1 << position)\ncheck_bit = (mask & (1 << position)) != 0\n\n# String operations and API calls\nstatus_message = \"Processing\" if is_enabled else \"Disabled\"\nmessage_length = len(status_message)\nuppercase_message = status_message.upper()\nreversed_message = status_message[::-1]\n\n# Multiple assignments and tuple unpacking\nfirst, second, *rest = numbers\na, b = b if 'b' in locals() else 10, a\nif 'a' in locals() else 20\n\n# Complex expression evaluation\ncomplex_expr = (final_score * 0.8 + log_result * 0.2) ** 0.5\nrounded_expr = round(complex_expr, 2)\nint_expr = int(complex_expr * 100) % 1000\n\n# Array operations and slicing\nsliced_numbers = numbers[2:8:2]\nreversed_slice = sliced_numbers[::-1]\nsum_slice = sum(reversed_slice)\n\n# Matrix operations\nmatrix_sum = sum(sum(row) for row in matrix)\ndiagonal_sum = sum(matrix[i][i] for i in range(len(matrix)))\nmatrix_flatten = [item for row in matrix for item in row]\n\n# Conditional assignments\nstatus_code = SUCCESS_CODE if all([is_enabled, has_valid_range, check_bit]) else ERROR_CODE\nerror_count = 0 if status_code == SUCCESS_CODE else 1\n\n# Final result calculation combining all elements\nresult = (\n    int_expr +\n    sum_slice +\n    diagonal_sum +\n    (status_code * 100) +\n    error_count +\n    len(rest) +\n    message_length +\n    (toggles_flags if 'toggles_flags' in locals() else toggled_flags)\n) % 10000\n\nprint(f\"Final result: {result}\")",
    "answer": 307
}
}

```

{

```
"id": "SL-MIX-S002",
"metadata": {
    "name": "StatementLevel-Mix-Seed2",
    "category": "Statement-Level",
    "subcategory": "Mix",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3
},
"task": {
    "description": "Given the following comprehensive C code snippet involving memory management, pointer arithmetic, struct operations, bit manipulation, and complex calculations, what is the final value stored in result->final_value?"
},
```

```

"code": "#include <stdio.h>\n#include <stdlib.h>\n#include <string.h>\n#include <math.h>\n#include <time.h>\n#define MAX_BUFFER_SIZE 256\n#define MAGIC_NUMBER 0xDEADBEEF\n#define SUCCESS 0\n#define FAILURE -1\n#define PI 3.14159265359\n#define E 2.71828182846\n\ntypedef struct {\n    int id;\n    double value;\n    char name[32];\n    int flags;\n    void *data;\n} DataNode;\n\ntypedef struct {\n    DataNode *nodes;\n    int count;\n    int capacity;\n    double average;\n    int final_value;\n} ResultSet;\n\n// Function prototypes\nint calculate_hash(const char *str);\ndouble compute_statistics(int *array, int size);\nint validate_data(const DataNode *node);\n\nint main() {\n    // Memory allocation and initialization\n    ResultSet *result = (ResultSet *)\n        malloc(sizeof(ResultSet));\n    if (!result) return FAILURE;\n    result->capacity = 10;\n    result->nodes = (DataNode *)\n        calloc(result->capacity, sizeof(DataNode));\n    if (!result->nodes) {\n        free(result);\n        return FAILURE;\n    }\n    // Constants and variables initialization\n    const int PRIME_NUMBERS[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};\n    const double COEFFICIENTS[] = {1.5, 2.3, 0.8, 1.2, 3.1};\n    int data_array[20] = {0};\n    char buffer[MAX_BUFFER_SIZE];\n    unsigned int seed = 12345;\n\n    // Initialize data array with computed values\n    for (int i = 0; i < 20; i++) {\n        data_array[i] = (PRIME_NUMBERS[i % 10] * (i + 1)) % 100;\n    }\n    // Node initialization with complex operations\n    result->count = 5;\n    for (int i = 0; i < result->count; i++) {\n        DataNode *node = &result->nodes[i];\n        node->id = (MAGIC_NUMBER >> (i * 4)) & 0xFF;\n\n        // Value calculation with mathematical functions\n        node->value = sin(i * PI / 4) * cos(i * PI / 6) * COEFFICIENTS[i];\n        node->value += sqrt(data_array[i * 2] + data_array[i * 2 + 1]);\n        node->value = round(node->value * 1000) / 1000.0;\n\n        // Name generation and string operations\n        snprintf(node->name, sizeof(node->name), \"Node_%d_%x\", i, node->id);\n\n        // Flags computation with bitwise operations\n        node->flags = 0;\n        if (node->value > 0) node->flags |= (1 << 0); // Positive flag\n        if (node->id % 2 == 0) node->flags |= (1 << 1); // Even ID flag\n        if (strlen(node->name) > 8) node->flags |= (1 << 2); // Long name flag\n\n        // Conditional memory allocation\n        if (validate_data(node)) {\n            node->data = malloc(sizeof(double) * 10);\n            if (node->data) {\n                double *data_ptr = (double *)node->data;\n                for (int j = 0; j < 10; j++) {\n                    data_ptr[j] = node->value * (j + 1) + COEFFICIENTS[j % 5];\n                }\n            } else {\n                node->data = NULL;\n            }\n        }\n\n        // Statistical calculations\n        double sum = 0.0;\n        int valid_count = 0;\n        int hash_sum = 0;\n        for (int i = 0; i < result->count; i++) {\n            DataNode *node = &result->nodes[i];\n            if (node->data != NULL) {\n                sum += node->value;\n                valid_count++;\n                hash_sum += calculate_hash(node->name);\n            }\n        }\n\n        // Average calculation with error handling\n        result->average = (valid_count > 0) ? (sum / valid_count) : 0.0;\n\n        // Complex final value computation\n        int temp_value = 0;\n\n        // Add statistical components\n        temp_value += (int)(result->average * 100);\n        temp_value += hash_sum % 1000;\n        temp_value += valid_count * 50;\n\n        // Add array statistics\n        double array_stats = compute_statistics(data_array, 20);\n        temp_value += (int)(array_stats * 10);\n\n        // Add bit manipulation results\n        unsigned int combined_flags = 0;\n        for (int i = 0; i < result->count; i++) {\n            combined_flags ^= result->nodes[i].flags;\n        }\n        temp_value += combined_flags;\n\n        // Add prime number operations\n        int prime_sum = 0;\n        for (int i = 0; i < 10; i++) {\n            prime_sum += PRIME_NUMBERS[i];\n        }\n        temp_value += (prime_sum % 256);\n\n        // Memory pattern analysis\n        int memory_pattern = 0;\n        for (int i = 0; i < result->count; i++) {\n            if (result->nodes[i].data != NULL) {\n                memory_pattern += (int)((uintptr_t)result->nodes[i].data & 0xFF);\n            }\n        }\n        temp_value += (memory_pattern % 128);\n\n        // String operations contribution\n        strcpy(buffer, \"ResultCalculation\");\n        int str_contrib = strlen(buffer);\n        for (int i = 0; buffer[i]; i++) {\n            str_contrib +=\n\n        }\n    }\n}
```

```

(int)buffer[i];\n    }\\n    temp_value += (str_contrib % 512);\\n    // Final modular\n    arithmetic\\n    result->final_value = temp_value % 9999;\\n    // Cleanup memory\\n\n    for (int i = 0; i < result->count; i++) {\\n        if (result->nodes[i].data) {\\n            free(result->nodes[i].data);\\n        }\\n    }\\n    free(result);\\n\n    int return_value = result->final_value;\\n\n    return return_value;\\n}\\n\n// Helper function\nimplementations\\nint calculate_hash(const char *str) {\\n    unsigned int hash = 5381;\\n\n    int c;\\n    while ((c = *str++)) {\\n        hash = ((hash << 5) + hash) + c;\\n    }\\n\n    return hash % 1000;\\n}\\n\ndouble compute_statistics(int *array, int size) {\\n    if (!array || size <= 0) return 0.0;\\n\n    int sum = 0;\\n    int max_val = array[0];\\n    int min_val = array[0];\\n\n    for (int i = 0; i < size; i++) {\\n        sum += array[i];\\n\n        if (array[i] > max_val) max_val = array[i];\\n        if (array[i] < min_val) min_val = array[i];\\n    }\\n\n    double mean = (double)sum / size;\\n    double variance = 0.0;\\n\n    for (int i = 0; i < size; i++) {\\n        variance += (array[i] - mean) * (array[i] - mean);\\n    }\\n\n    variance /= size;\\n\n    return sqrt(variance) + (max_val - min_val) * 0.1;\\n}\\n\nint validate_data(const DataNode *node) {\\n    if (!node) return 0;\\n\n    return (node->id > 0 && node->value >= -1000.0 && node->value <= 1000.0);\\n},\n\n    "answer": 2875\n}\n}

```

2 - 代码块级推理 [Block-Level] (24)

2A - 线性代码块 [Linear] (6)

```

{\n\n    "id": "BL-LN-S001",\n\n    "metadata": {\n\n        "name": "BlockLevel-Linear-Sequential",\n        "category": "Block-Level",\n        "subcategory": "Linear",\n        "type": "seed",\n        "source": "CodeSense-krb5",\n        "language": "c",\n        "difficulty": "medium",\n        "intervention": 1\n    },\n\n    "task": {\n\n        "description": "Given the following sequential code block, what is the final value of buf->len after executing all statements?",\n\n        "code": "void k5_buf_init_dynamic(struct k5buf *buf) {\\n            buf->buftype =\n            K5BUF_DYNAMIC;\\n            buf->space = 128;\\n            buf->data = malloc(buf->space);\\n            if (buf->data == NULL) {\\n                set_error(buf);\\n                return;\\n            }\\n            buf->len = 0;\\n        }",\n\n        "answer": 0\n    }\n}

```

```

{\n\n    "id": "BL-LN-S002",\n\n    "metadata": {\n\n        "name": "BlockLevel-Linear-Independent",\n        "category": "Block-Level",\n\n    }\n}

```

```

        "subcategory": "Linear",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following independent statements block, what is the final value of max_retries?",
        "code": "max_retries = 3\ntimeout_seconds = 30.0\nlog_level = 'INFO'\nis_debug = False\ndefault_encoding = 'utf-8'",
        "answer": 3
    }
}

```

```

{
    "id": "BL-LN-S003",
    "metadata": {
        "name": "BlockLevel-Linear-Dependent",
        "category": "Block-Level",
        "subcategory": "Linear",
        "type": "seed",
        "source": "CodeSense-util-linux",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Given the following dependent statements block, what is the return value when blkid_do_safeprobe returns 0?",
        "code": "static int process_file(const char *name) {\n    int rc = -1;\n\n    blkid_probe pr = blkid_new_probe_from_filename(name);\n    if (pr != NULL) {\n        blkid_probe_enable_partitions(pr, TRUE);\n        rc = blkid_do_safeprobe(pr) == -1 ? -1 : 0;\n    }\n    blkid_free_probe(pr);\n    return rc;\n}",
        "answer": 0
    }
}

```

```

{
    "id": "BL-LN-S004",
    "metadata": {
        "name": "BlockLevel-Linear-Chain",
        "category": "Block-Level",
        "subcategory": "Linear",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {

```

```
        "description": "Given the following calculation chain, what is the final value of result?",  
        "code": "base = 10\nmultiplier = 3\noffset = 5\n\ttemp = base * multiplier\n\tresult = temp + offset\n\nprintf(f\"Final result: {result}\")",  
        "answer": 35  
    }  
}
```

```
{  
    "id": "BL-LN-S005",  
    "metadata": {  
        "name": "BlockLevel-Linear-Accumulation",  
        "category": "Block-Level",  
        "subcategory": "Linear",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "medium",  
        "intervention": 1  
    },  
    "task": {  
        "description": "Given the following accumulation block, what is the final value of total?",  
        "code": "int total = 0;\ntotal += 10;\ntotal += 15;\ntotal += 7;\ntotal *= 2;\n\nprintf(\"Total: %d\\n\", total);",  
        "answer": 64  
    }  
}
```

```
{  
    "id": "BL-LN-S006",  
    "metadata": {  
        "name": "BlockLevel-Linear-Transform",  
        "category": "Block-Level",  
        "subcategory": "Linear",  
        "type": "seed",  
        "source": "CodeSense-tmux",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 2  
    },  
    "task": {  
        "description": "Given the following data transformation block, what is the final value of s->ccolour?",  
        "code": "void screen_init(struct screen *s, u_int sx, u_int sy, u_int hlimit) {\n    s->grid = grid_create(sx, sy, hlimit);\n    s->saved_grid = NULL;\n    s->cstyle = SCREEN_CURSOR_DEFAULT;\n    s->default_cstyle = SCREEN_CURSOR_DEFAULT;\n    s->ccolour = -1;\n    s->default_ccolour = -1;\n}",  
        "answer": -1  
    }  
}
```

2B - 条件代码块 [Conditional] (8)

```
{  
  "id": "BL-CD-S001",  
  "metadata": {  
    "name": "BlockLevel-Conditional-SimpleIf",  
    "category": "Block-Level",  
    "subcategory": "Conditional",  
    "type": "seed",  
    "source": "CodeSense-cryptsetup",  
    "language": "c",  
    "difficulty": "medium",  
    "intervention": 1  
  },  
  "task": {  
    "description": "Given the following conditional block where sysconf returns 4096, what is the return value?",  
    "code": "size_t crypt_getpagesize(void) {\n      long r = sysconf(_SC_PAGESIZE);\n      if (r <= 0) {\n        return DEFAULT_MEM_ALIGNMENT;\n      } else {\n        return (size_t)r;\n      }\n    }",  
    "answer": 4096  
  }  
}
```

```
{  
  "id": "BL-CD-S002",  
  "metadata": {  
    "name": "BlockLevel-Conditional-NestedIf",  
    "category": "Block-Level",  
    "subcategory": "Conditional",  
    "type": "seed",  
    "source": "Manual",  
    "language": "python",  
    "difficulty": "hard",  
    "intervention": 2  
  },  
  "task": {  
    "description": "Given the following nested conditional block where score is 85, what is the final value of grade?",  
    "code": "score = 85\nif score >= 90:\n    if score >= 95:\n        grade = 'A+'\n    else:\n        grade = 'A'\nelse:\n    if score >= 80:\n        grade = 'B'\n    else:\n        grade = 'C'\nprint(f\"Grade: {grade}\")",  
    "answer": "B"  
  }  
}
```

```
{  
  "id": "BL-CD-S003",  
  "metadata": {  
    "name": "BlockLevel-Conditional-ElseIf",  
    "category": "Block-Level",  
    "subcategory": "Conditional",  
    "type": "seed",  
    "source": "CodeSense-cryptsetup",  
    "language": "c",  
    "difficulty": "medium",  
    "intervention": 1  
  },  
  "task": {  
    "description": "Given the following conditional block where sysconf returns 4096, what is the return value?",  
    "code": "size_t crypt_getpagesize(void) {\n      long r = sysconf(_SC_PAGESIZE);\n      if (r <= 0) {\n        return DEFAULT_MEM_ALIGNMENT;\n      } else {\n        return (size_t)r;\n      }\n    }",  
    "answer": 4096  
  }  
}
```

```

    "subcategory": "Conditional",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Given the following else-if chain where value is 15, what is the final value of category?",
    "code": "int value = 15;\nchar *category;\nif (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
    "answer": "medium"
  }
}

```

```

{
  "id": "BL-CD-S004",
  "metadata": {
    "name": "BlockLevel-Conditional-Switch",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Given the following switch block where day is 3, what is the final value of day_name?",
    "code": "int day = 3;\nchar *day_name;\nswitch (day) {\n    case 1: day_name = \"Monday\"; break;\n    case 2: day_name = \"Tuesday\"; break;\n    case 3: day_name = \"Wednesday\"; break;\n    case 4: day_name = \"Thursday\"; break;\n    case 5: day_name = \"Friday\"; break;\n    default: day_name = \"Weekend\";\n}\nprintf(\"Day: %s\\n\", day_name);",
    "answer": "Wednesday"
  }
}

```

```

{
  "id": "BL-CD-S005",
  "metadata": {
    "name": "BlockLevel-Conditional-SwitchFallthrough",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "hard",
  }
}

```

```

        "intervention": 2
    },
    "task": {
        "description": "Given the following switch block with fallthrough where input is 2, what is the final value of flags?",
        "code": "int input = 2;\nint flags = 0;\nswitch (input) {\n    case 1:\n        flags |= 0x01;\n    case 2:\n        flags |= 0x02;\n    case 3:\n        flags |= 0x04;\n    default:\n        flags = -1;\n}\nprintf(\"Flags: 0x%02x\\n\", flags);",
        "answer": 6
    }
}

```

```

{
    "id": "BL-CD-S006",
    "metadata": {
        "name": "BlockLevel-Conditional-TernaryChain",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Given the following ternary conditional chain where x is 7 and y is 5, what is the final value of result?",
        "code": "int x = 7, y = 5;\nint result = (x > y) ? ((x % 2 == 0) ? x * 2 : x + 10) : ((y % 2 == 0) ? y * 3 : y - 2);\nprintf(\"Result: %d\\n\", result);",
        "answer": 17
    }
}

```

```

{
    "id": "BL-CD-S007",
    "metadata": {
        "name": "BlockLevel-Conditional-NullCheck",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "seed",
        "source": "CodeSense-libssh",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Given the following null check block where malloc succeeds, what is the final value of ptr after the block execution?",
        "code": "ssh_poll_handle ssh_poll_new(socket_t fd, short events) {\n    ssh_poll_handle p;\n    p = malloc(sizeof(struct ssh_poll_handle_struct));\n    if (p == NULL) {\n        return NULL;\n    }\n    p->x.fd = fd;\n    p->events = events;\n    return p;\n}",
        "answer": 1
    }
}

```

```
        "answer": "valid_pointer"
    }
}
```

```
{
    "id": "BL-CD-S008",
    "metadata": {
        "name": "BlockLevel-Conditional-ComplexLogic",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3
    },
    "task": {
        "description": "Given the following complex conditional logic where age is 25, income is 50000, and has_degree is True, what is the final value of status?",
        "code": "age = 25\nincome = 50000\nhas_degree = True\nstatus = None\n\nif age >= 18\nand age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            if age >= 21 and has_degree:\n                status = \"approved_standard\"\n            else:\n                status = \"approved_basic\"\n    else:\n        status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_premium"
    }
}
```

2C - 迭代代码块 [Iterative] (8)

```
{
    "id": "BL-IT-S001",
    "metadata": {
        "name": "BlockLevel-Iterative-simpleFor",
        "category": "Block-Level",
        "subcategory": "Iterative",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following simple for loop block, what is the final value of sum?",
        "code": "int sum = 0;\nfor (int i = 1; i <= 5; i++) {\n    sum += i;\n}\nprintf(\"Sum: %d\\n\", sum);",
        "answer": 15
    }
}
```

```
{
  "id": "BL-IT-S002",
  "metadata": {
    "name": "BlockLevel-Iterative-ArrayInit",
    "category": "Block-Level",
    "subcategory": "Iterative",
    "type": "seed",
    "source": "CodeSense-apache-httdp",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Given the following array initialization loop where GB_SIZE is 5, what is the value of pointer_arr[3] after execution?",
    "code": "void af_gb_init() {\n    pointer_idx = 0;\n    for (int i = 0; i < GB_SIZE;\n        i++) {\n        pointer_arr[i] = NULL;\n    }\n}",
    "answer": "NULL"
  }
}
```

```
{
  "id": "BL-IT-S003",
  "metadata": {
    "name": "BlockLevel-Iterative-WhileLoop",
    "category": "Block-Level",
    "subcategory": "Iterative",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Given the following while loop block where initial value is 16, what is the final value of count?",
    "code": "int value = 16;\nint count = 0;\nwhile (value > 1) {\n    value = value / 2;\n    count++;\n}\nprintf(\"Count: %d\\n\", count);",
    "answer": 4
  }
}
```

```
{
  "id": "BL-IT-S004",
  "metadata": {
    "name": "BlockLevel-Iterative-NestedLoop",
    "category": "Block-Level",
    "subcategory": "Iterative",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "intervention": 1
  }
}
```

```

        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Given the following nested loop block, what is the final value of total?",
        "code": "total = 0\nfor i in range(3):\n    for j in range(2):\n        total += (i + 1) * (j + 1)\nprint(f\"Total: {total}\")",
        "answer": 18
    }
}

```

```

{
    "id": "BL-IT-S005",
    "metadata": {
        "name": "BlockLevel-Iterative-BreakContinue",
        "category": "Block-Level",
        "subcategory": "Iterative",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Given the following loop with break and continue, what is the final value of sum?",
        "code": "int sum = 0;\nfor (int i = 1; i <= 10; i++) {\n    if (i % 2 == 0) {\n        continue;\n    }\n    if (i > 7) {\n        break;\n    }\n    sum += i;\n}\nprintf(\"Sum: %d\\n\", sum);",
        "answer": 16
    }
}

```

```

{
    "id": "BL-IT-S006",
    "metadata": {
        "name": "BlockLevel-Iterative-DoWhile",
        "category": "Block-Level",
        "subcategory": "Iterative",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Given the following do-while loop where initial n is 0, what is the final value of n?",
        "code": "int n = 0;\ndo {\n    n += 3;\n} while (n < 10);\nprintf(\"n: %d\\n\", n);",
        "answer": 12
    }
}

```

```
    }  
}
```

```
{  
    "id": "BL-IT-S007",  
    "metadata": {  
        "name": "BlockLevel-Iterative-RecursiveFunction",  
        "category": "Block-Level",  
        "subcategory": "Iterative",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "expert",  
        "intervention": 3  
    },  
    "task": {  
        "description": "Given the following recursive function call factorial(4), what is the return value?",  
        "code": "int factorial(int n) {\n            if (n <= 1) {\n                return 1;\n            }\n            return n * factorial(n - 1);\n        }\n\nint result = factorial(4);\nprintf(\"Result: %d\\n\", result);",  
        "answer": 24  
    }  
}
```

```
{  
    "id": "BL-IT-S008",  
    "metadata": {  
        "name": "BlockLevel-Iterative-ComplexAccumulation",  
        "category": "Block-Level",  
        "subcategory": "Iterative",  
        "type": "seed",  
        "source": "Manual",  
        "language": "python",  
        "difficulty": "expert",  
        "intervention": 3  
    },  
    "task": {  
        "description": "Given the following complex iterative accumulation with filtering, what is the final value of result?",  
        "code": "numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]\nresult = 0\nmultiplier = 1\n\nfor num in numbers:\n    if num % 2 == 0:\n        result += num * multiplier\n    multiplier += 1\nelse:\n    result -= num\n\nprint(f\"Result: {result}\")",  
        "answer": 15  
    }  
}
```

2D - 大混合 (2)

```
{  
  "id": "BL-MIX-S001",  
  "metadata": {  
    "name": "BlockLevel-Mix-Comprehensive",  
    "category": "Block-Level",  
    "subcategory": "Mix",  
    "type": "seed",  
    "source": "Manual",  
    "language": "c",  
    "difficulty": "expert",  
    "intervention": 3  
  },  
  "task": {  
    "description": "Given the following comprehensive block mixing linear, conditional, and iterative patterns, what is the final value of status_code?",  
    "code": "int process_data_batch(int *data, int size) {\n    // Linear initialization\n    int processed = 0;\n    int errors = 0;\n    int status_code = 0;\n\n    // Conditional validation\n    if (data == NULL || size <= 0) {\n        return -1;\n    }\n\n    // Iterative processing with nested conditions\n    for (int i = 0; i < size; i++) {\n        if (data[i] < 0) {\n            errors++;\n            continue;\n        }\n\n        // Linear computation\n        data[i] = data[i] * 2 + 1;\n\n        processed++;\n\n        // Conditional break\n        if (errors > size / 2) {\n            status_code = -2;\n            break;\n        }\n    }\n\n    // Final conditional status\n    if (status_code == 0) {\n        status_code = (processed > 0) ? 1 : 0;\n    }\n\n    return status_code;\n}\n\n// Test case\nint test_data[] = {1, 2, 3, 4, 5};\nint result = process_data_batch(test_data, 5);",  
    "answer": 1  
  }  
}
```

```
{  
  "id": "BL-MIX-S002",  
  "metadata": {  
    "name": "BlockLevel-Mix-Realworld",  
    "category": "Block-Level",  
    "subcategory": "Mix",  
    "type": "seed",  
    "source": "CodeSense-libssh",  
    "language": "c",  
    "difficulty": "expert",  
    "intervention": 3  
  },  
  "task": {  
    "description": "Given the following real-world configuration parsing block, what is the final value of parser_flags when input contains 3 lines?",  
    "code": "parser_flags = 0;  
for (int i = 0; i < 3; i++) {  
    if (line[i] == '1') {  
        parser_flags |= 1;  
    }  
    if (line[i] == '2') {  
        parser_flags |= 2;  
    }  
    if (line[i] == '3') {  
        parser_flags |= 4;  
    }  
    if (line[i] == '4') {  
        parser_flags |= 8;  
    }  
    if (line[i] == '5') {  
        parser_flags |= 16;  
    }  
    if (line[i] == '6') {  
        parser_flags |= 32;  
    }  
    if (line[i] == '7') {  
        parser_flags |= 64;  
    }  
    if (line[i] == '8') {  
        parser_flags |= 128;  
    }  
    if (line[i] == '9') {  
        parser_flags |= 256;  
    }  
    if (line[i] == 'A') {  
        parser_flags |= 512;  
    }  
    if (line[i] == 'B') {  
        parser_flags |= 1024;  
    }  
    if (line[i] == 'C') {  
        parser_flags |= 2048;  
    }  
    if (line[i] == 'D') {  
        parser_flags |= 4096;  
    }  
    if (line[i] == 'E') {  
        parser_flags |= 8192;  
    }  
    if (line[i] == 'F') {  
        parser_flags |= 16384;  
    }  
    if (line[i] == 'G') {  
        parser_flags |= 32768;  
    }  
    if (line[i] == 'H') {  
        parser_flags |= 65536;  
    }  
    if (line[i] == 'I') {  
        parser_flags |= 131072;  
    }  
    if (line[i] == 'J') {  
        parser_flags |= 262144;  
    }  
    if (line[i] == 'K') {  
        parser_flags |= 524288;  
    }  
    if (line[i] == 'L') {  
        parser_flags |= 1048576;  
    }  
    if (line[i] == 'M') {  
        parser_flags |= 2097152;  
    }  
    if (line[i] == 'N') {  
        parser_flags |= 4194304;  
    }  
    if (line[i] == 'O') {  
        parser_flags |= 8388608;  
    }  
    if (line[i] == 'P') {  
        parser_flags |= 16777216;  
    }  
    if (line[i] == 'Q') {  
        parser_flags |= 33554432;  
    }  
    if (line[i] == 'R') {  
        parser_flags |= 67108864;  
    }  
    if (line[i] == 'S') {  
        parser_flags |= 134217728;  
    }  
    if (line[i] == 'T') {  
        parser_flags |= 268435456;  
    }  
    if (line[i] == 'U') {  
        parser_flags |= 536870912;  
    }  
    if (line[i] == 'V') {  
        parser_flags |= 1073741824;  
    }  
    if (line[i] == 'W') {  
        parser_flags |= 2147483648;  
    }  
    if (line[i] == 'X') {  
        parser_flags |= 4294967296;  
    }  
    if (line[i] == 'Y') {  
        parser_flags |= 8589934592;  
    }  
    if (line[i] == 'Z') {  
        parser_flags |= 17179869184;  
    }  
    if (line[i] == 'a') {  
        parser_flags |= 34359738320;  
    }  
    if (line[i] == 'b') {  
        parser_flags |= 68719476640;  
    }  
    if (line[i] == 'c') {  
        parser_flags |= 137438953280;  
    }  
    if (line[i] == 'd') {  
        parser_flags |= 274877906560;  
    }  
    if (line[i] == 'e') {  
        parser_flags |= 549755813120;  
    }  
    if (line[i] == 'f') {  
        parser_flags |= 1099511626240;  
    }  
    if (line[i] == 'g') {  
        parser_flags |= 2199023252480;  
    }  
    if (line[i] == 'h') {  
        parser_flags |= 4398046504960;  
    }  
    if (line[i] == 'i') {  
        parser_flags |= 8796093009920;  
    }  
    if (line[i] == 'j') {  
        parser_flags |= 17592186019840;  
    }  
    if (line[i] == 'k') {  
        parser_flags |= 35184372039680;  
    }  
    if (line[i] == 'l') {  
        parser_flags |= 70368744079360;  
    }  
    if (line[i] == 'm') {  
        parser_flags |= 140737488158720;  
    }  
    if (line[i] == 'n') {  
        parser_flags |= 281474976317440;  
    }  
    if (line[i] == 'o') {  
        parser_flags |= 562949952634880;  
    }  
    if (line[i] == 'p') {  
        parser_flags |= 1125899905269760;  
    }  
    if (line[i] == 'q') {  
        parser_flags |= 2251799810539520;  
    }  
    if (line[i] == 'r') {  
        parser_flags |= 4503599621079040;  
    }  
    if (line[i] == 's') {  
        parser_flags |= 9007199242158080;  
    }  
    if (line[i] == 't') {  
        parser_flags |= 18014398484316160;  
    }  
    if (line[i] == 'u') {  
        parser_flags |= 36028796968632320;  
    }  
    if (line[i] == 'v') {  
        parser_flags |= 72057593937264640;  
    }  
    if (line[i] == 'w') {  
        parser_flags |= 144115187874529280;  
    }  
    if (line[i] == 'x') {  
        parser_flags |= 288230375749058560;  
    }  
    if (line[i] == 'y') {  
        parser_flags |= 576460751498117120;  
    }  
    if (line[i] == 'z') {  
        parser_flags |= 1152921502996234240;  
    }  
    if (line[i] == 'A') {  
        parser_flags |= 2305843004992468480;  
    }  
    if (line[i] == 'B') {  
        parser_flags |= 4611686009984936960;  
    }  
    if (line[i] == 'C') {  
        parser_flags |= 9223372019969873920;  
    }  
    if (line[i] == 'D') {  
        parser_flags |= 18446744039939747840;  
    }  
    if (line[i] == 'E') {  
        parser_flags |= 36893488079879495680;  
    }  
    if (line[i] == 'F') {  
        parser_flags |= 73786976159758991360;  
    }  
    if (line[i] == 'G') {  
        parser_flags |= 147573952319517982720;  
    }  
    if (line[i] == 'H') {  
        parser_flags |= 295147904638535965440;  
    }  
    if (line[i] == 'I') {  
        parser_flags |= 590295809277071930880;  
    }  
    if (line[i] == 'J') {  
        parser_flags |= 1180591618554143861760;  
    }  
    if (line[i] == 'K') {  
        parser_flags |= 2361183237108287723520;  
    }  
    if (line[i] == 'L') {  
        parser_flags |= 4722366474216575447040;  
    }  
    if (line[i] == 'M') {  
        parser_flags |= 9444732948433150894080;  
    }  
    if (line[i] == 'N') {  
        parser_flags |= 18889465896866301788160;  
    }  
    if (line[i] == 'O') {  
        parser_flags |= 37778931793732603576320;  
    }  
    if (line[i] == 'P') {  
        parser_flags |= 75557863587465207152640;  
    }  
    if (line[i] == 'Q') {  
        parser_flags |= 151115727174930414305280;  
    }  
    if (line[i] == 'R') {  
        parser_flags |= 302231454349860828610560;  
    }  
    if (line[i] == 'S') {  
        parser_flags |= 604462908699721657221120;  
    }  
    if (line[i] == 'T') {  
        parser_flags |= 1208925817399443314442240;  
    }  
    if (line[i] == 'U') {  
        parser_flags |= 2417851634798886628884480;  
    }  
    if (line[i] == 'V') {  
        parser_flags |= 4835703269597773257768960;  
    }  
    if (line[i] == 'W') {  
        parser_flags |= 9671406539195546515537920;  
    }  
    if (line[i] == 'X') {  
        parser_flags |= 19342813078391093031075840;  
    }  
    if (line[i] == 'Y') {  
        parser_flags |= 38685626156782186062151680;  
    }  
    if (line[i] == 'Z') {  
        parser_flags |= 77371252313564372124303360;  
    }  
    if (line[i] == 'a') {  
        parser_flags |= 15474250462712874424860640;  
    }  
    if (line[i] == 'b') {  
        parser_flags |= 30948500925425748849721280;  
    }  
    if (line[i] == 'c') {  
        parser_flags |= 61897001850851497699442560;  
    }  
    if (line[i] == 'd') {  
        parser_flags |= 123794003701702995398885120;  
    }  
    if (line[i] == 'e') {  
        parser_flags |= 247588007403405990797760240;  
    }  
    if (line[i] == 'f') {  
        parser_flags |= 495176014806811981595520480;  
    }  
    if (line[i] == 'g') {  
        parser_flags |= 990352029613623963191040960;  
    }  
    if (line[i] == 'h') {  
        parser_flags |= 1980704059227247926382081920;  
    }  
    if (line[i] == 'i') {  
        parser_flags |= 3961408118454495852764163840;  
    }  
    if (line[i] == 'j') {  
        parser_flags |= 7922816236908991705528327680;  
    }  
    if (line[i] == 'k') {  
        parser_flags |= 15845632473817983411056655360;  
    }  
    if (line[i] == 'l') {  
        parser_flags |= 31691264947635966822113310720;  
    }  
    if (line[i] == 'm') {  
        parser_flags |= 63382529895271933644226621440;  
    }  
    if (line[i] == 'n') {  
        parser_flags |= 126765059790543867288453242880;  
    }  
    if (line[i] == 'o') {  
        parser_flags |= 253530119581087734576906485760;  
    }  
    if (line[i] == 'p') {  
        parser_flags |= 507060239162175469153812971520;  
    }  
    if (line[i] == 'q') {  
        parser_flags |= 1014120478324350938307625943040;  
    }  
    if (line[i] == 'r') {  
        parser_flags |= 2028240956648701876615251886080;  
    }  
    if (line[i] == 's') {  
        parser_flags |= 4056481913297403753230503772160;  
    }  
    if (line[i] == 't') {  
        parser_flags |= 8112963826594807506461007544320;  
    }  
    if (line[i] == 'u') {  
        parser_flags |= 16225927653189615012922015088640;  
    }  
    if (line[i] == 'v') {  
        parser_flags |= 32451855306379230025844030177280;  
    }  
    if (line[i] == 'w') {  
        parser_flags |= 64903710612758460051688060354560;  
    }  
    if (line[i] == 'x') {  
        parser_flags |= 129807421225516920103376120709120;  
    }  
    if (line[i] == 'y') {  
        parser_flags |= 259614842451033840206752241418240;  
    }  
    if (line[i] == 'z') {  
        parser_flags |= 519229684902067680413504482836480;  
    }  
    if (line[i] == 'A') {  
        parser_flags |= 103845936980413536082700896567280;  
    }  
    if (line[i] == 'B') {  
        parser_flags |= 207691873960827072165401793134560;  
    }  
    if (line[i] == 'C') {  
        parser_flags |= 415383747921654144330803586269120;  
    }  
    if (line[i] == 'D') {  
        parser_flags |= 830767495843308288661607172538240;  
    }  
    if (line[i] == 'E') {  
        parser_flags |= 1661534991686616577323214345076480;  
    }  
    if (line[i] == 'F') {  
        parser_flags |= 3323069983373233154646428680152960;  
    }  
    if (line[i] == 'G') {  
        parser_flags |= 6646139966746466309292857360305920;  
    }  
    if (line[i] == 'H') {  
        parser_flags |= 13292279933492932618585714720611840;  
    }  
    if (line[i] == 'I') {  
        parser_flags |= 26584559866985865237171429441223680;  
    }  
    if (line[i] == 'J') {  
        parser_flags |= 53169119733971730474342858882447360;  
    }  
    if (line[i] == 'K') {  
        parser_flags |= 106338239467943460948685717764894720;  
    }  
    if (line[i] == 'L') {  
        parser_flags |= 212676478935886921897371435529789440;  
    }  
    if (line[i] == 'M') {  
        parser_flags |= 425352957871773843794742871059578880;  
    }  
    if (line[i] == 'N') {  
        parser_flags |= 850705915743547687589485742119157600;  
    }  
    if (line[i] == 'O') {  
        parser_flags |= 1701411831487095375178971484238315200;  
    }  
    if (line[i] == 'P') {  
        parser_flags |= 3402823662974190750357942968476630400;  
    }  
    if (line[i] == 'Q') {  
        parser_flags |= 6805647325948381500715885936953260800;  
    }  
    if (line[i] == 'R') {  
        parser_flags |= 13611294651896763001437771873865521600;  
    }  
    if (line[i] == 'S') {  
        parser_flags |= 27222589303793526002875543747731043200;  
    }  
    if (line[i] == 'T') {  
        parser_flags |= 54445178607587052005751087495462086400;  
    }  
    if (line[i] == 'U') {  
        parser_flags |= 108890357215174104011522154985921772800;  
    }  
    if (line[i] == 'V') {  
        parser_flags |= 217780714430348208023044309971843545600;  
    }  
    if (line[i] == 'W') {  
        parser_flags |= 435561428860696416046088619943687091200;  
    }  
    if (line[i] == 'X') {  
        parser_flags |= 871122857721392832092177239887374182400;  
    }  
    if (line[i] == 'Y') {  
        parser_flags |= 1742245715442785664184354479774748364800;  
    }  
    if (line[i] == 'Z') {  
        parser_flags |= 3484491430885571328368708959549496729600;  
    }  
    if (line[i] == 'A') {  
        parser_flags |= 6968982861771142656737417919098993459200;  
    }  
    if (line[i] == 'B') {  
        parser_flags |= 13937965723542285313474835838197986988000;  
    }  
    if (line[i] == 'C') {  
        parser_flags |= 27875931447084570626949671676395973976000;  
    }  
    if (line[i] == 'D') {  
        parser_flags |= 55751862894169141253899343352791947952000;  
    }  
    if (line[i] == 'E') {  
        parser_flags |= 111503725788338282507798686705583895040000;  
    }  
    if (line[i] == 'F') {  
        parser_flags |= 223007451576676565015597373411167790080000;  
    }  
    if (line[i] == 'G') {  
        parser_flags |= 446014903153353130031194746822335580160000;  
    }  
    if (line[i] == 'H') {  
        parser_flags |= 892029806306706260062389493644671160320000;  
    }  
    if (line[i] == 'I') {  
        parser_flags |= 1784059612613412520124778987289342326400000;  
    }  
    if (line[i] == 'J') {  
        parser_flags |= 3568119225226825040249557974578684652800000;  
    }  
    if (line[i] == 'K') {  
        parser_flags |= 7136238450453650080499115949157369305600000;  
    }  
    if (line[i] == 'L') {  
        parser_flags |= 14272476900907300160998231898314738611200000;  
    }  
    if (line[i] == 'M') {  
        parser_flags |= 28544953801814600321996463796629477222400000;  
    }  
    if (line[i] == 'N') {  
        parser_flags |= 5708990760362920064399292759325895444800000;  
    }  
    if (line[i] == 'O') {  
        parser_flags |= 11417981520725840128798585518651790896000000;  
    }  
    if (line[i] == 'P') {  
        parser_flags |= 22835963041451680257597171037303581792000000;  
    }  
    if (line[i] == 'Q') {  
        parser_flags |= 45671926082903360515194342074607163584000000;  
    }  
    if (line[i] == 'R') {  
        parser_flags |= 91343852165806721030388684149214327168000000;  
    }  
    if (line[i] == 'S') {  
        parser_flags |= 182687704331613442060777368298428654336000000;  
    }  
    if (line[i] == 'T') {  
        parser_flags |= 365375408663226884121554736596857308672000000;  
    }  
    if (line[i] == 'U') {  
        parser_flags |= 730750817326453768243109473193714617344000000;  
    }  
    if (line[i] == 'V') {  
        parser_flags |= 1461501634652907536486218946385429234688000000;  
    }  
    if (line[i] == 'W') {  
        parser_flags |= 2923003269305815072972437892770858469376000000;  
    }  
    if (line[i] == 'X') {  
        parser_flags |= 5846006538611630145944875785541716937552000000;  
    }  
    if (line[i] == 'Y') {  
        parser_flags |= 11692013077223260291889511571083433875056000000;  
    }  
    if (line[i] == 'Z') {  
        parser_flags |= 23384026154446520583779023142166867750112000000;  
    }  
    if (line[i] == 'A') {  
        parser_flags |= 46768052308893041167558046284333735502224000000;  
    }  
    if (line[i] == 'B') {  
        parser_flags |= 93536104617786082335116092568667471004448000000;  
    }  
    if (line[i] == 'C') {  
        parser_flags |= 187072209235572164670232185137334942008896000000;  
    }  
    if (line[i] == 'D') {  
        parser_flags |= 374144418471144329340464370274669884017792000000;  
    }  
    if (line[i] == 'E') {  
        parser_flags |= 748288836942288658680928740549339768035584000000;  
    }  
    if (line[i] == 'F') {  
        parser_flags |= 1496577673884577317361857481098675536071168000000;  
    }  
    if (line[i] == 'G') {  
        parser_flags |= 2993155347769154634723714962197351072142336000000;  
    }  
    if (line[i] == 'H') {  
        parser_flags |= 5986310695538309269447429924394702144284672000000;  
    }  
    if (line[i] == 'I') {  
        parser_flags |= 11972621391076618538894859848789444288573440000000;  
    }  
    if (line[i] == 'J') {  
        parser_flags |= 23945242782153237077789719697578888577146880000000;  
    }  
    if (line[i] == 'K') {  
        parser_flags |= 47890485564306474155579439395157777154293760000000;  
    }  
    if (line[i] == 'L') {  
        parser_flags |= 95780971128612948311158878790315554308587360000000;  
    }  
    if (line[i] == 'M') {  
        parser_flags |= 191561942257225896622317757580631108617174720000000;  
    }  
    if (line[i] == 'N') {  
        parser_flags |= 383123884514451793244635515161262217234349440000000;  
    }  
    if (line[i] == 'O') {  
        parser_flags |= 766247769028903586489271030322524434468698800000000;  
    }  
    if (line[i] == 'P') {  
        parser_flags |= 1532495538057807172978542060645048868937397600000000;  
    }  
    if (line[i] == 'Q') {  
        parser_flags |= 3064991076115614345957084121290097737874795200000000;  
    }  
    if (line[i] == 'R') {  
        parser_flags |= 6129982152231228691914168242580195475749592000000000;  
    }  
    if (line[i] == 'S') {  
        parser_flags |= 12259964304462457383828336485160390951499184000000000;  
    }  
    if (line[i] == 'T') {  
        parser_flags |= 24519928608924914767656672970320781902989368000000000;  
    }  
    if (line[i] == 'U') {  
        parser_flags |= 49039857217849829535313345940641563805978736000000000;  
    }  
    if (line[i] == 'V') {  
        parser_flags |= 98079714435699659070626691881283127611957472000000000;  
    }  
    if (line[i] == 'W') {  
        parser_flags |= 196159428871399318141253383762566255223854944000000000;  
    }  
    if (line[i] == 'X') {  
        parser_flags |= 392318857742798636282506767525132510447709880000000000;  
    }  
    if (line[i] == 'Y') {  
        parser_flags |= 7846377154855972725650135350502650208954197600000000000;  
    }  
    if (line[i] == 'Z')
```

```

    "code": "int ssh_bind_config_parse_string(ssh_bind bind, const char *input) {\nchar line[MAX_LINE_SIZE] = {0};\n    const char *c = input, *line_start = input;\n    unsigned int line_num = 0, line_len;\n    uint32_t parser_flags;\n    int rv;\n\n    // Linear initialization\n    uint8_t seen[BIND_CFG_MAX] = {0};\n    parser_flags = PARSING;\n\n    // Iterative line processing\n    while (1) {\n        line_num++;\n        line_start = c;\n        c = strchr(line_start, '\\n');\n\n        // Conditional end detection\n        if (c == NULL) {\n            c = strchr(line_start, '\\0');\n\n            if (c == NULL) {\n                return SSH_ERROR;\n            }\n        }\n\n        // Linear line processing\n        line_len = c - line_start;\n\n        if (line_len > MAX_LINE_SIZE - 1) {\n            return SSH_ERROR;\n        }\n\n        memcpy(line, line_start, line_len);\n        line[line_len] = '\\0';\n\n        // Conditional parsing\n        rv = ssh_bind_config_parse_line(bind, line,\n            line_num, &parser_flags, seen, 0);\n\n        if (rv < 0) {\n            return SSH_ERROR;\n        }\n\n        // Break condition\n        if (*c == '\\0') {\n            break;\n        }\n        c++;\n    }\n\n    return SSH_OK;\n}\n\n// Assuming input has 3 valid lines and PARSING = 1\nconst char *test_input = \"line1\\nline2\\nline3\";\nint result = ssh_bind_config_parse_string(NULL, test_input);",
    "answer": 0
}
}

```

3 - 代码属性推理 [Property-Level] (27)

3A - 循环属性 [Loop] (6)

(迭代计数、变量追踪、终止条件)

```

{
    "id": "PL-LP-S001",
    "metadata": {
        "name": "PropertyLevel-Loop-IterationCount",
        "category": "Property-Level",
        "subcategory": "Loop",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Given the following nested loop structure, how many total iterations will the inner loop execute across all outer loop iterations?",
        "code": "int total_iterations = 0;\nfor (int i = 1; i <= 4; i++) {\n    for (int j = i; j <= 6; j++) {\n        total_iterations++;\n        printf(\"i=%d, j=%d\\n\", i, j);\n    }\n}\nprintf(\"Total inner iterations: %d\\n\", total_iterations);",
        "answer": 18
    }
}

```

```
{
    "id": "PL-LP-S002",
    "metadata": {

```

```

    "name": "PropertyLevel-Loop-VariableTracking",
    "category": "Property-Level",
    "subcategory": "Loop",
    "type": "seed",
    "source": "CodeSense-openssl",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "In this loop that processes a buffer, trace the evolution of variable 'remaining' - what is its value at the start of the 3rd iteration?",
    "code": "int process_buffer(unsigned char *buf, int len) {\n    int remaining = len;\n    int processed = 0;\n    int chunk_size;\n    while (remaining > 0) {\n        chunk_size = (remaining > 64) ? 64 : remaining;\n        processed += chunk_size;\n        remaining -= chunk_size;\n        printf(\"Iteration: processed=%d, remaining=%d\\n\", processed, remaining);\n    }\n    return processed;\n}\n// Called with len = 200\nint result = process_buffer(buffer, 200);",
    "answer": 72
  }
}

```

```

{
  "id": "PL-LP-S003",
  "metadata": {
    "name": "PropertyLevel-Loop-TerminationCondition",
    "category": "Property-Level",
    "subcategory": "Loop",
    "type": "seed",
    "source": "CodeSense-tmux",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Analyze this loop's termination condition - given the initial state, on which iteration number will the loop terminate?",
    "code": "int find_convergence(double initial_value) {\n    double x = initial_value;\n    double prev_x;\n    int iteration = 0;\n    double tolerance = 0.01;\n\n    do {\n        prev_x = x;\n        x = (x + 2.0/x) / 2.0; // Newton's method for\n        sqrt(2);\n        iteration++;\n        printf(\"Iteration %d: x = %.6f, diff = %.6f\\n\", iteration, x, fabs(x - prev_x));\n    } while (fabs(x - prev_x) >= tolerance && iteration < 10);\n\n    return iteration;\n}\n// Called with initial_value = 1.0\nint result = find_convergence(1.0);",
    "answer": 4
  }
}

```

```

{
  "id": "PL-LP-S004",
  "metadata": {
    "name": "PropertyLevel-Loop-ConditionalCounting",

```

```

    "category": "Property-Level",
    "subcategory": "Loop",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "In this loop with conditional processing, how many times is the variable 'special_count' incremented?",
    "code": "numbers = [1, 7, 3, 14, 8, 21, 5, 28, 9, 35]\nspecial_count = 0\nsum_special = 0\nfor i, num in enumerate(numbers):\n    if num % 7 == 0: # Multiples of 7\n        special_count += 1\n        sum_special += num\n        print(f\"Found special number {num} at index {i}\")\n    elif num > 10:\n        print(f\"Large number {num} at index {i}\")\n\nprint(f\"Special count: {special_count}, sum: {sum_special}\")",
    "answer": 4
  }
}

```

```

{
  "id": "PL-LP-S005",
  "metadata": {
    "name": "PropertyLevel-Loop-AccumulatorPattern",
    "category": "Property-Level",
    "subcategory": "Loop",
    "type": "seed",
    "source": "CodeSense-postgresql",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Track the accumulator variable 'hash' through this hash computation loop - what is its value after processing the 4th character?",
    "code": "unsigned int compute_hash(const char *str) {\n    unsigned int hash = 5381;\n    int c;\n    int pos = 0;\n    while ((c = str[pos]) != '\\0') {\n        hash = ((hash << 5) + hash) + c; // hash * 33 + c\n        pos++;\n    }\n    return hash;\n}\n\n// Called with str = \"test\"\nunsigned int result = compute_hash(\"test\");",
    "answer": 6385719596
  }
}

```

```

{
  "id": "PL-LP-S006",
  "metadata": {
    "name": "PropertyLevel-Loop-MultipleExitConditions",
    "category": "Property-Level",
    "subcategory": "Loop",
    "type": "seed",
    "source": "Manual",
  }
}

```

```

    "language": "c",
    "difficulty": "expert",
    "intervention": 3
},
"task": {
    "description": "This loop has multiple exit conditions. Given the input array, which specific condition will cause the loop to terminate and on which iteration?",
    "code": "int search_with_conditions(int *arr, int size) {\n    int iterations = 0;\n    int sum = 0;\n    int found_target = 0;\n    for (int i = 0; i < size; i++) {\n        iterations++;\n        sum += arr[i];\n        // Condition 1: Found target value\n        if (arr[i] == 42) {\n            found_target = 1;\n            printf(\"Found target at iteration %d\\n\", iterations);\n            break;\n        }\n        // Condition 2: Sum exceeds threshold\n        if (sum > 100) {\n            printf(\"Sum exceeded at iteration %d (sum=%d)\\n\", iterations, sum);\n            break;\n        }\n        // Condition 3: Maximum iterations\n        if (iterations >= 8) {\n            printf(\"Max iterations reached\\n\");\n            break;\n        }\n    }\n    return iterations;\n}\n// Test array: [15, 25, 30, 20, 35, 42, 10, 5]\nint test_arr[] = {15, 25, 30, 20, 35, 42, 10, 5};\nint result = search_with_conditions(test_arr, 8);",
    "answer": 4
}
}

```

3B - 分支属性 [Branch] (7)

(条件求值、路径选择、分支效果)

```

{
    "id": "PL-BR-S001",
    "metadata": {
        "name": "PropertyLevel-Branch-ConditionEvaluation",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Given the specific input values, determine which branch path is taken and what value is returned.",
        "code": "int evaluate_condition(int a, int b, int c) {\n    if (a > 10 && b < 5) {\n        return 1;\n    } else if (a <= 10 && c > 0) {\n        return 2;\n    } else {\n        return 3;\n    }\n}\n// what value is returned when called with evaluate_condition(8, 6, 4)?",
        "answer": 2
    }
}

```

```
{
    "id": "PL-BR-S002",

```

```

"metadata": {
    "name": "PropertyLevel-Branch-ShortCircuit",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
},
"task": {
    "description": "Analyze short-circuit evaluation. How many function calls are actually executed?",
    "code": "int call_count = 0;\nint func_a() { call_count++; return 0; }\nint func_b()\n{ call_count++; return 1; }\nint func_c() { call_count++; return 1; }\n\nint result =\nfunc_a() && func_b() && func_c();\n// What is the value of call_count after this expression?",
    "answer": 1
}
}

```

```

{
    "id": "PL-BR-S003",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedTernary",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Evaluate the nested ternary expression with the given values.",
        "code": "int x = 7, y = 3, z = 5;\nint result = (x > y) ? ((z % 2 == 1) ? x + z : x - z) : ((y > z) ? y * 2 : y + z);\n// What is the value of result?",
        "answer": 12
    }
}

```

```

{
    "id": "PL-BR-S004",
    "metadata": {
        "name": "PropertyLevel-Branch-SwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
}

```

```

},
"task": {
    "description": "Calculate the final value considering switch statement fallthrough behavior.",
    "code": "int value = 0;\nint input = 2;\nswitch (input) {\n    case 1: value += 10;\n    case 2: value += 20;\n    case 3: value += 30;\n        break;\n    case 4: value += 40;\n        break;\n    default: value = -1;\n}\n// What is the final value of 'value'?",
    "answer": 50
}
}

```

```

{
    "id": "PL-BR-S005",
    "metadata": {
        "name": "PropertyLevel-Branch-ConditionalModification",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Track variable modification through conditional branches.",
        "code": "x = 10\nif x > 5:\n    x = x * 2\nif x > 15:\n    x = x + 5\nif x < 30:\n    x = x - 3\n# What is the final value of x?",
        "answer": 22
    }
}

```

```

{
    "id": "PL-BR-S006",
    "metadata": {
        "name": "PropertyLevel-Branch-MultipleConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Determine the final output value based on multiple conditional checks.",
        "code": "int process_value(int n) {\n    int result = n;\n    if (n % 2 == 0) result += 5;\n    if (n % 3 == 0) result *= 2;\n    if (n % 5 == 0) result -= 10;\n    return result;\n}\n// What value is returned when called with process_value(12)?",
        "answer": 34
    }
}

```

```
}
```

```
{
  "id": "PL-BR-S007",
  "metadata": {
    "name": "PropertyLevel-Branch-CompoundLogic",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3
  },
  "task": {
    "description": "Evaluate complex compound logical expression and determine the final boolean result.",
    "code": "int a = 5, b = 8, c = 3;\nint result = ((a < b) && (b > c)) || ((a + c) == b) && !(a > c * 2);\n// What is the value of result (1 for true, 0 for false)?",
    "answer": 1
  }
}
```

3C - 内存属性 [Memory] (6)

(引用关系、生命周期、访问模式)

```
{
  "id": "PL-MEM-S001",
  "metadata": {
    "name": "PropertyLevel-Memory-PointerArithmetic",
    "category": "Property-Level",
    "subcategory": "Memory",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Calculate the final value accessed through pointer arithmetic.",
    "code": "int arr[5] = {10, 20, 30, 40, 50};\nint *ptr = arr + 2;\nptr++;\nint value = *ptr;\n// What is the value of 'value'?",
    "answer": 40
  }
}
```

```
{
  "id": "PL-MEM-S002",
  "metadata": {
    "name": "PropertyLevel-Memory-ArrayModification",
    "category": "Property-Level",
    "subcategory": "Memory",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Modify the value at a specific index in an array using pointer arithmetic and calculate the new value.",
    "code": "int arr[5] = {10, 20, 30, 40, 50};\nint *ptr = arr + 2;\n*ptr = 100;\nint value = *ptr;\n// What is the value of 'value'?",
    "answer": 100
  }
}
```

```
        "category": "Property-Level",
        "subcategory": "Memory",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Determine the array element value after pointer-based modification.",
        "code": "int data[4] = {1, 2, 3, 4};\nint *p1 = &data[1];\nint *p2 = &data[2];\n*p1 = *p2 + 5;\n*p2 = *p1 * 2;\n// What is the value of data[2] after these operations?",
        "answer": 16
    }
}
```

```
{
    "id": "PL-MEM-S003",
    "metadata": {
        "name": "PropertyLevel-Memory-StructAccess",
        "category": "Property-Level",
        "subcategory": "Memory",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Calculate the final struct member value after pointer operations.",
        "code": "typedef struct { int x; int y; } Point;\nPoint points[3] = {{1, 2}, {3, 4}, {5, 6}};\nPoint *ptr = points + 1;\nptr->x = ptr->x + points[0].y;\nptr->y = ptr->y * 2;\n// What is the value of points[1].x after these operations?",
        "answer": 5
    }
}
```

```
{
    "id": "PL-MEM-S004",
    "metadata": {
        "name": "PropertyLevel-Memory-IndirectAccess",
        "category": "Property-Level",
        "subcategory": "Memory",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
```

```
        "description": "Trace through multiple levels of indirection to find the final value.",  
        "code": "int value = 100;\nint *ptr1 = &value;\nint **ptr2 = &ptr1;\nint ***ptr3 = &ptr2;\n**ptr2 = **ptr2 + 50;\n// What is the value of 'value' after this operation?",  
        "answer": 150  
    }  
}
```

```
{  
    "id": "PL-MEM-S005",  
    "metadata": {  
        "name": "PropertyLevel-Memory-ArrayCopy",  
        "category": "Property-Level",  
        "subcategory": "Memory",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 2  
    },  
    "task": {  
        "description": "Determine array contents after memory copy operations.",  
        "code": "int source[5] = {1, 2, 3, 4, 5};\nint dest[5] = {0, 0, 0, 0, 0};\nfor (int i = 0; i < 3; i++) {\n    dest[i + 1] = source[i] * 2;\n}\n// What is the value of dest[3]?",  
        "answer": 6  
    }  
}
```

```
{  
    "id": "PL-MEM-S006",  
    "metadata": {  
        "name": "PropertyLevel-Memory-OverlappingAccess",  
        "category": "Property-Level",  
        "subcategory": "Memory",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "expert",  
        "intervention": 3  
    },  
    "task": {  
        "description": "Calculate the result of overlapping memory access patterns.",  
        "code": "int buffer[6] = {10, 20, 30, 40, 50, 60};\nint *p1 = buffer + 1;\nint *p2 = buffer + 3;\nfor (int i = 0; i < 2; i++) {\n    p1[i] = p2[i] + p1[i];\n}\n// What is the value of buffer[2] after the loop?",  
        "answer": 70  
    }  
}
```

3D - 作用域属性 [Scope] (6)

(可见性、生存期、变量遮蔽)

```
{  
    "id": "PL-SC-S001",  
    "metadata": {  
        "name": "PropertyLevel-Scope-VariableShadowing",  
        "category": "Property-Level",  
        "subcategory": "Scope",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "medium",  
        "intervention": 1  
    },  
    "task": {  
        "description": "Determine which variable value is accessed in the innermost scope.",  
        "code": "int x = 10;\nvoid test_scope() {\n    int x = 20;\n    {\n        int x = 30;\n        printf(\"%d\\", x);\n    }\n} // what value is printed by the printf statement?",  
        "answer": 30  
    }  
}
```

```
{  
    "id": "PL-SC-S002",  
    "metadata": {  
        "name": "PropertyLevel-Scope-StaticVariable",  
        "category": "Property-Level",  
        "subcategory": "Scope",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 2  
    },  
    "task": {  
        "description": "Calculate the static variable value after multiple function calls.",  
        "code": "int get_count() {\n    static int count = 0;\n    count += 5;\n    return count;\n} // what value is returned by the third call to get_count()?",  
        "answer": 15  
    }  
}
```

```
{  
    "id": "PL-SC-S003",  
    "metadata": {  
        "name": "PropertyLevel-Scope-Blockvariable",  
        "category": "Property-Level",  
        "subcategory": "Scope",  
        "type": "seed",  
        "source": "Manual",  
        "language": "c",  
        "difficulty": "medium",  
        "intervention": 1  
    },  
    "task": {  
        "description": "Determine the value of the variable 'x' in the innermost block.",  
        "code": "int x = 10;\nvoid test_scope() {\n    int x = 20;\n    {\n        int x = 30;\n        {\n            int x = 40;\n            printf(\"%d\\", x);\n        }\n    }\n} // what value is printed by the printf statement?",  
        "answer": 40  
    }  
}
```

```

    "source": "Manual",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
},
"task": {
    "description": "Determine the variable value after exiting the block scope.",
    "code": "int value = 5;\n\n    int value = 10;\n    value *= 2;\n\n}value += 3;\n// what is the final value of the outer 'value' variable?",
    "answer": 8
}
}

```

```

{
    "id": "PL-SC-S004",
    "metadata": {
        "name": "PropertyLevel-Scope-GlobalModification",
        "category": "Property-Level",
        "subcategory": "Scope",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Track global variable modification across function calls.",
        "code": "int global_var = 0;\nvoid modify_global(int increment) {\n    global_var += increment;\n}\nvoid test_function() {\n    modify_global(5);\n    modify_global(3);\n    modify_global(-2);\n}\n// what is the value of global_var after calling test_function()?",
        "answer": 6
    }
}

```

```

{
    "id": "PL-SC-S005",
    "metadata": {
        "name": "PropertyLevel-Scope-ParametersShadowing",
        "category": "Property-Level",
        "subcategory": "Scope",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Determine the final return value considering parameter shadowing.",
        "code": "int x = 100;\nint calculate(int x) {\n    x = x * 2;\n    {\n        int x = 50;\n        x += 10;\n    }\n    return x;\n}\n// what value is returned by calculate(15)?",
        "answer": 30
    }
}

```

```
    }\n}
```

```
{\n    "id": "PL-SC-S006",\n    "metadata": {\n        "name": "PropertyLevel-Scope-NestedFunctions",\n        "category": "Property-Level",\n        "subcategory": "Scope",\n        "type": "seed",\n        "source": "Manual",\n        "language": "python",\n        "difficulty": "expert",\n        "intervention": 3\n    },\n    "task": {\n        "description": "Calculate the final value considering nested function scopes and closures.",\n        "code": "def outer_function(x):\n            def inner_function():\n                nonlocal x\n                x = x * 2\n                return x\n            result = inner_function()\n            x = x + 5\n        return x\n    # What value is returned by outer_function(10)?",\n        "answer": 25\n    }\n}
```

3E - 大混合 (2)

```
{\n    "id": "PL-MIX-S001",\n    "metadata": {\n        "name": "PropertyLevel-Mix-Comprehensive",\n        "category": "Property-Level",\n        "subcategory": "Mix",\n        "type": "seed",\n        "source": "Manual",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3\n    },\n    "task": {\n        "description": "Comprehensive analysis combining loops, branches, memory, and scope. What is the final value of result?",\n        "code": "int global_counter = 0;\n\nint process_array()\n{\n    int arr[4] = {2, 4, 6, 8};\n    int *ptr = arr;\n    int result = 0;\n\n    for (int i = 0; i < 4; i++) {\n        global_counter++;\n\n        if (arr[i] % 4 == 0) {\n            *ptr = *ptr * 2;\n            result += *ptr;\n        } else {\n            result += arr[i];\n\n            ptr++;}\n    }\n\n    int local_var = global_counter;\n    result = result + local_var;\n}\n\nreturn result;\n\n// What is the return value of process_array()?",\n        "answer": 50\n    }\n}
```

```
}
```

```
{
  "id": "PL-MIX-S002",
  "metadata": {
    "name": "PropertyLevel-Mix-Complex",
    "category": "Property-Level",
    "subcategory": "Mix",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "master",
    "intervention": 4
  },
  "task": {
    "description": "Complex property interaction analysis. What is the final value stored at memory location pointed to by final_ptr?",
    "code": "static int static_var = 10;\n\nint complex_operations() {\n    int data[5] = {1, 2, 3, 4, 5};\n    int *ptr1 = data + 1;\n    int *ptr2 = data + 3;\n    int *final_ptr;\n\n    for (int i = 0; i < 3; i++) {\n        static_var++;\n\n        if (i % 2 == 0) {\n            *ptr1 = *ptr1 + static_var;\n            final_ptr = ptr1;\n        } else {\n            *ptr2 = *ptr2 * 2;\n            final_ptr = ptr2;\n\n            if (temp > 10)\n                *final_ptr = temp - 5;\n        }\n    }\n\n    if (ptr1 >= data + 4) ptr1 = data + 1;\n\n    return *final_ptr;\n}\n\n// what value is returned by complex_operations()?",
    "answer": 8
  }
}
```

4 - 跨函数推理 [MultiFunc-Level] (32)

4A - 函数调用链推理 [Call Chain] (8)

直接调用、链式调用、递归调用、回归调用

```
{
  "id": "MF-CC-S001",
  "metadata": {
    "name": "MultiFunc-CallChain-DirectCall",
    "category": "MultiFunc-Level",
    "subcategory": "Call Chain",
    "type": "seed",
    "source": "CodeSense-openssl",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Given the following direct function call scenario, what is the return value when process_request is called with buffer_size = 1024?",
    "code": "int process_request(int buffer_size) {\n    if (buffer_size < 0) {\n        return -1;\n    }\n\n    // ... (implementation details)\n\n    return 1024;\n}\n\nint main() {\n    int result = process_request(1024);\n\n    if (result == 1024) {\n        // ... (success logic)\n    } else {\n        // ... (failure logic)\n    }\n}\n\n// What is the return value of process_request(1024)?"
```

```

    "code": "int validate_buffer_size(int size) {\n    if (size < 64) return 0;\n    if (size > 8192) return 0;\n    return 1;}\n\nint allocate_buffer(int size) {\n    if (!validate_buffer_size(size)) {\n        return -1;\n    }\n    return size * 2;}\n\nint process_request(int buffer_size) {\n    int allocated = allocate_buffer(buffer_size);\n    if (allocated == -1) {\n        return 0;\n    }\n    return allocated + 100;}// what\nis returned by process_request(1024)?",
    "answer": 2148
}
}

```

```

{
  "id": "MF-CC-S002",
  "metadata": {
    "name": "MultiFunc-CallChain-DirectCall-Error",
    "category": "MultiFunc-Level",
    "subcategory": "Call Chain",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1
  },
  "task": {
    "description": "Trace the direct function call with error handling. what is the\nfinal return value?",  

    "code": "def validate_input(value):\n    if value < 0:\n        return False\n    if value > 100:\n        return False\n    return True\n\ndef calculate_score(input_value):\n    if not validate_input(input_value):\n        return -1\n    return input_value * 3 +\n10\n\ndef get_final_result(user_input):\n    score = calculate_score(user_input)\n    if\nscore == -1:\n        return 0\n    return score + 5\n\n# what is returned by\nget_final_result(25)?",
    "answer": 90
  }
}

```

```

{
  "id": "MF-CC-S003",
  "metadata": {
    "name": "MultiFunc-CallChain-ChainCall",
    "category": "MultiFunc-Level",
    "subcategory": "Call Chain",
    "type": "seed",
    "source": "CodeSense-postgresql",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Analyze this chain of function calls in a database query processor.\nwhat is the final result?",  

    "code": "int validate_buffer_size(int size) {\n    if (size < 64) return 0;\n    if (size > 8192) return 0;\n    return 1;}\n\nint allocate_buffer(int size) {\n    if (!validate_buffer_size(size)) {\n        return -1;\n    }\n    return size * 2;}\n\nint process_request(int buffer_size) {\n    int allocated = allocate_buffer(buffer_size);\n    if (allocated == -1) {\n        return 0;\n    }\n    return allocated + 100;}// what\nis returned by process_request(1024)?",
    "answer": 2148
  }
}

```

```

    "code": "int parse_query(char *query) {\n        if (query == NULL) return 0;\n        return strlen(query) > 0 ? 1 : 0;}\n\nint validate_query(char *query) {\n        int parsed =\n        parse_query(query);\n        if (!parsed) return 0;\n        return strlen(query) < 1000 ? 1 :\n        0;}\n\nint optimize_query(char *query) {\n        int valid = validate_query(query);\n        if\n        (!valid) return -1;\n        return strlen(query) / 10 + 1;}\n\nint execute_query(char *query)\n{\n        int optimized = optimize_query(query);\n        if (optimized == -1) return 0;\n        return optimized * 5;}\n\n// Given query = \"SELECT * FROM users WHERE id = 1\", what does\nexecute_query return?\n// Note: strlen(\"SELECT * FROM users WHERE id = 1\") = 31",
    "answer": 20
}
}

```

```

{
    "id": "MF-CC-S004",
    "metadata": {
        "name": "MultiFunc-CallChain-ChainTransform",
        "category": "MultiFunc-Level",
        "subcategory": "Call Chain",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Follow the data transformation chain. What is the final transformed value?",
        "code": "def normalize(value):\n    return value / 100.0\n\ndef\napply_coefficient(value):\n    normalized = normalize(value)\n    return normalized *\n1.5\n\ndef add_bias(value):\n    transformed = apply_coefficient(value)\n    return\ntransformed + 0.25\n\ndef round_result(value):\n    biased = add_bias(value)\n    return\nround(biased, 2)\n\n# What is returned by round_result(300)?",
        "answer": 4.75
    }
}

```

```

{
    "id": "MF-CC-S005",
    "metadata": {
        "name": "MultiFunc-CallChain-Recursive",
        "category": "MultiFunc-Level",
        "subcategory": "Call Chain",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Trace through this recursive function call. What is the final return value?",
        "code": "int factorial(int n)\n{\n    if (n == 0)\n        return 1;\n    else\n        return n * factorial(n - 1);\n}\n\nint main()\n{\n    int result = factorial(5);\n    return result;\n}"
    }
}

```

```
        "code": "int fibonacci(int n) {\n            if (n <= 1) {\n                return n;\n            }\n            return fibonacci(n - 1) + fibonacci(n - 2);\n        }\n\n        int calculate_sum(int limit) {\n            int result = 0;\n            for (int i = 0; i < limit; i++) {\n                result += fibonacci(i);\n            }\n            return result;\n        }\n\n        // What is returned by calculate_sum(6)?\n        // Note: fibonacci sequence: 0,1,1,2,3,5...\n    },\n\n    \"answer\": 12\n}\n}
```

```
{  
  "id": "MF-CC-S006",  
  "metadata": {  
    "name": "MultiFunc-CallChain-TailRecursive",  
    "category": "MultiFunc-Level",  
    "subcategory": "Call Chain",  
    "type": "seed",  
    "source": "Manual",  
    "language": "python",  
    "difficulty": "hard",  
    "intervention": 2  
  },  
  "task": {  
    "description": "Analyze this tail-recursive function with accumulator. What is the final result?",  
    "code": "def factorial_helper(n, acc):\n        if n <= 1:\n            return acc\n    return factorial_helper(n - 1, acc * n)\n\ndef factorial(n):\n    return factorial_helper(n, 1)\n\ndef calculate_combination(n, r):\n    numerator = factorial(n)\n    denominator = factorial(r) * factorial(n - r)\n    return numerator // denominator\n\n# What is returned by calculate_combination(5, 2)?",  
    "answer": 10  
  }  
}
```

```
{  
  "id": "MF-CC-S007",  
  "metadata": {  
    "name": "MultiFunc-CallChain-Callback",  
    "category": "MultiFunc-Level",  
    "subcategory": "Call Chain",  
    "type": "seed",  
    "source": "CodeSense-libevent",  
    "language": "c",  
    "difficulty": "expert",  
    "intervention": 3  
  },  
  "task": {  
    "description": "Trace through callback function execution. What is the final value of result?",  
    "solution": "The final value of result is 3."  
  }  
}
```

```

    "code": "typedef int (*callback_func)(int);\n\nint double_value(int x) {\n    return\n    x * 2;\n}\n\nint add_ten(int x) {\n    return x + 10;\n}\n\nint apply_callback(int value,\n    callback_func cb) {\n    return cb(value);\n}\n\nint process_with_callbacks(int initial) {\n    int step1 = apply_callback(initial, double_value);\n    int step2 = apply_callback(step1,\n        add_ten);\n    return step2;\n}\n\n// What is returned by process_with_callbacks(15)?",
    "answer": 40
}
}

```

```

{
    "id": "MF-CC-S008",
    "metadata": {
        "name": "MultiFunc-CallChain-HigherOrder",
        "category": "MultiFunc-Level",
        "subcategory": "Call Chain",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3
    },
    "task": {
        "description": "Analyze higher-order function composition. What is the final computed result?",
        "code": "def compose(f, g):\n    return lambda x: f(g(x))\n\ndef square(x):\n    return x * x\n\ndef increment(x):\n    return x + 1\n\ndef apply_twice(f):\n    return\n    lambda x: f(f(x))\n\n# Create composed function\ncomposed = compose(square,\n    increment)\n\n# What is returned by\n# twice_composed(3)?",
        "answer": 256
    }
}

```

4B - 参数传递推理 [Parameter Passing] (6)

值传递、引用传递

```

{
    "id": "MF-PP-S001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ValuePass",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Analyze value passing behavior. What is the value of original_value after function execution?",
        "code": "int original_value = 10;\n\nint add_ten(int x) {\n    return x + 10;\n}\n\nint apply_callback(int value,\n    int (*callback)(int)) {\n    return callback(value);\n}\n\nint process_with_callbacks(int initial) {\n    int step1 = apply_callback(initial, add_ten);\n    int step2 = apply_callback(step1,\n        add_ten);\n    return step2;\n}\n\nint main() {\n    int result = process_with_callbacks(original_value);\n    printf(\"Result: %d\\n\", result);\n    return 0;\n}"
    }
}

```

```

    "code": "void modify_value(int value) {\n    value = value * 2 + 10;\n    printf(\"Inside function: %d\\n\", value);\n}\n\nint main() {\n    int original_value = 25;\n    modify_value(original_value);\n    printf(\"After function: %d\\n\", original_value);\n    return 0;\n}\n\n// What is the value of original_value after modify_value is called?",\n    "answer": 25\n}\n}

```

```

{
    "id": "MF-PP-S002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ValueStructure",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Trace struct value passing. What is the value of p1.x after the function call?",\n        "code": "typedef struct {\n            int x;\n            int y;\n} Point;\n\nvoid transform_point(Point p) {\n    p.x = p.x * 2;\n    p.y = p.y + 5;\n}\n\nPoint create_and_transform() {\n    Point p1 = {10, 20};\n    transform_point(p1);\n    return p1;\n}\n\n// What is the x value of the Point returned by create_and_transform()?",\n        "answer": 10
    }
}

```

```

{
    "id": "MF-PP-S003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerPass",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "seed",
        "source": "CodeSense-linux-kernel",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Analyze pointer parameter passing with memory modification. What is the final value?",\n        "code": "void update_values(int *arr, int size) {\n            for (int i = 0; i < size; i++) {\n                arr[i] = arr[i] * 2 + i;\n            }\n}\n\nvoid process_data() {\n    int data[4] = {5, 10, 15, 20};\n    update_values(data, 4);\n    printf(\"data[2] = %d\\n\", data[2]);\n}\n\n// What is the value of data[2] after update_values is called?",\n        "answer": 32
    }
}

```

```
    }\n}
```

```
{\n    "id": "MF-PP-S004",\n    "metadata": {\n        "name": "MultiFunc-ParameterPassing-DoublePointer",\n        "category": "MultiFunc-Level",\n        "subcategory": "Parameter Passing",\n        "type": "seed",\n        "source": "CodeSense-tmux",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3\n    },\n    "task": {\n        "description": "Track double pointer manipulation. What is the final value pointed to?",\n        "code": "void allocate_and_set(int **ptr, int value) {\n            *ptr =\n            malloc(sizeof(int));\n            **ptr = value * 3;\n}\nvoid modify_pointer_target(int **ptr) {\n    **ptr = **ptr + 15;\n}\nint test_double_pointer() {\n    int *p = NULL;\n    allocate_and_set(&p, 10);\n    modify_pointer_target(&p);\n    int result = *p;\n    free(p);\n    return result;\n}\n// What value is returned by test_double_pointer()?",\n        "answer": 45\n    }\n}
```

```
{\n    "id": "MF-PP-S005",\n    "metadata": {\n        "name": "MultiFunc-ParameterPassing-Reference",\n        "category": "MultiFunc-Level",\n        "subcategory": "Parameter Passing",\n        "type": "seed",\n        "source": "Manual",\n        "language": "python",\n        "difficulty": "medium",\n        "intervention": 1\n    },\n    "task": {\n        "description": "Analyze Python reference passing with mutable objects. What is the final list content?",\n        "code": "def modify_list(lst):\n    lst.append(100)\n    lst[0] = lst[0] * 2\nlst.extend([200, 300])\n\ndef process_data():\n    my_list = [10, 20, 30]\n    modify_list(my_list)\n    return len(my_list)\n\n# What is returned by process_data()?",\n        "answer": 6\n    }\n}
```

```
{\n    "id": "MF-PP-S006",\n
```

```

"metadata": {
    "name": "MultiFunc-ParameterPassing-MixedTypes",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3
},
"task": {
    "description": "Complex parameter passing with mixed types. What is the final computed value?",
    "code": "typedef struct {\n    int *data;\n    int count;\n} DataContainer;\n\nvoid process_container(DataContainer *container, int multiplier) {\n    for (int i = 0; i < container->count; i++) {\n        container->data[i] *= multiplier;\n    }\n    container->count++;\n}\n\nint calculate_sum(DataContainer container) {\n    int sum = 0;\n    for (int i = 0; i < container.count; i++) {\n        sum += container.data[i];\n    }\n    return sum;\n}\n\nint test_mixed_passing() {\n    int values[5] = {2, 4, 6, 8, 10};\n    DataContainer dc = {values, 4};\n    process_container(&dc, 3);\n    return calculate_sum(dc);\n}\n\n// What value is returned by test_mixed_passing()?",
    "answer": 60
}
}

```

4C - 状态传播推理 [State Propagation] (8)

全局变量、静态变量、闭包捕获、对象状态

```

{
    "id": "MF-SP-S001",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalVar",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "seed",
        "source": "CodeSense-apache-httdp",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Track global variable state across multiple function calls. What is the final value?",
        "code": "int global_counter = 0;\n\nint global_multiplier = 2;\n\nvoid increment_counter(int value) {\n    global_counter += value * global_multiplier;\n}\n\nvoid reset_and_multiply() {\n    global_counter = 0;\n    global_multiplier *= 3;\n}\n\nint process_sequence() {\n    increment_counter(5);\n    increment_counter(3);\n    reset_and_multiply();\n    increment_counter(4);\n    return global_counter;\n}\n\n// What value is returned by process_sequence()?",
        "answer": 24
    }
}

```

```
}
```

```
{
  "id": "MF-SP-S002",
  "metadata": {
    "name": "MultiFunc-StatePropagation-GlobalArray",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Analyze global array modification across functions. What is the sum of all elements?",
    "code": "int shared_array[5] = {1, 2, 3, 4, 5};\nint array_size = 5;\nvoid double_elements() {\n    for (int i = 0; i < array_size; i++) {\n        shared_array[i] *= 2;\n    }\n}\nvoid add_index_to_elements() {\n    for (int i = 0; i < array_size; i++) {\n        shared_array[i] += i;\n    }\n}\nint calculate_total() {\n    double_elements();\n    add_index_to_elements();\n    int sum = 0;\n    for (int i = 0; i < array_size; i++) {\n        sum += shared_array[i];\n    }\n    return sum;\n} // What value is returned by calculate_total()?",
    "answer": 40
  }
}
```

```
{
```

```

  "id": "MF-SP-S003",
  "metadata": {
    "name": "MultiFunc-StatePropagation-StaticVar",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "CodeSense-openssl",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Track static variable persistence across function calls. What is the final accumulated value?",
    "code": "int get_next_id() {\n    static int id_counter = 100;\n    id_counter += 10;\n    return id_counter;\n}\nint calculate_checksum() {\n    static int checksum = 0;\n    int id = get_next_id();\n    checksum += id;\n    return checksum;\n}\nint process_multiple_items() {\n    int total = 0;\n    total += calculate_checksum();\n    total += calculate_checksum();\n    total += calculate_checksum();\n    return total;\n} // What value is returned by process_multiple_items()?",
    "answer": 480
  }
}
```

```
}
```

```
{
  "id": "MF-SP-S004",
  "metadata": {
    "name": "MultiFunc-StatePropagation-StaticArray",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3
  },
  "task": {
    "description": "Analyze static array state management across multiple function invocations. What is the final array element value?",
    "code": "int add_to_cache(int value) {\n    static int cache[3] = {0, 0, 0};\n\n    static int index = 0;\n    cache[index] = value;\n    index = (index + 1) % 3;\n\n    int sum = 0;\n    for (int i = 0; i < 3; i++) {\n        sum += cache[i];\n    }\n\n    return sum;\n}\n\nint test_cache_operations() {\n    add_to_cache(10);\n    add_to_cache(20);\n    add_to_cache(30);\n    return add_to_cache(40);\n}\n\n// What value is returned by test_cache_operations()?",
    "answer": 90
  }
}
```

```
{
  "id": "MF-SP-S005",
  "metadata": {
    "name": "MultiFunc-StatePropagation-Closure",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Trace closure variable capture and modification. What is the final captured value?",
    "code": "def create_accumulator(initial):\n    total = initial\n    def add_value(value):\n        nonlocal total\n        total += value\n        return total\n\n    return add_value\n\ndef test_closure():\n    acc1 = create_accumulator(10)\n    acc2 = create_accumulator(20)\n\n    result1 = acc1(5)\n    result2 = acc2(15)\n    result3 = acc1(10)\n\n    return result3\n\n# What value is returned by test_closure()?",
    "answer": 25
  }
}
```

```
{
  "id": "MF-SP-S006",
  "metadata": {
    "name": "MultiFunc-StatePropagation-NestedClosure",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "expert",
    "intervention": 3
  },
  "task": {
    "description": "Analyze nested closure state propagation. What is the final computed result?",
    "code": "def create_calculator(base):\n    def create_operation(operation_type):\n        def calculate(value):\n            if operation_type == 'add':\n                return base + value\n            elif operation_type == 'multiply':\n                return base * value\n            else:\n                return value\n        return calculate\n    return create_operation\n\ndef test_nested_closure():\n    calc = create_calculator(10)\n    adder = calc('add')\n    multiplier = calc('multiply')\n    result1 = adder(5)\n    result2 = multiplier(3)\n    return result1 + result2\n\n# What value is returned by test_nested_closure()?",
    "answer": 45
  }
}
```

```
{
  "id": "MF-SP-S007",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ObjectState",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "seed",
    "source": "Manual",
    "language": "python",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Track object state changes across method calls. What is the final balance?",
  }
}
```

```

"code": "class BankAccount:\n    def __init__(self, initial_balance):\n        self.balance = initial_balance\n        self.transaction_count = 0\n    def deposit(self, amount):\n        self.balance += amount\n        self.transaction_count += 1\n        return self.balance\n    def withdraw(self, amount):\n        if self.balance >= amount:\n            self.balance -= amount\n            self.transaction_count += 1\n            return self.balance\n        def apply_interest(self):\n            if self.transaction_count >= 3:\n                self.balance = int(self.balance * 1.05)\n            return self.balance\n    def test_account_operations():\n        account = BankAccount(100)\n        account.deposit(50)\n        account.withdraw(30)\n        account.deposit(20)\n        return account.apply_interest()\n\n# What value is returned by test_account_operations()?",\n    "answer": 147\n}\n}

```

```

{
    "id": "MF-SP-S008",
    "metadata": {
        "name": "MultiFunc-StatePropagation-SharedObject",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3
    },
    "task": {
        "description": "Analyze shared object state across multiple functions. What is the final computed value?",\n        "code": "class SharedCounter:\n            def __init__(self):\n                self.value = 0\n            self.multiplier = 1\n            def increment(self, amount=1):\n                self.value += amount * self.multiplier\n            def set_multiplier(self, mult):\n                self.multiplier = mult\n            def get_value(self):\n                return self.value\n        counter = SharedCounter()\n        def operation_a():\n            counter.increment(5)\n        counter.set_multiplier(2)\n        def operation_b():\n            counter.increment(3)\n        counter.increment(4)\n        def operation_c():\n            counter.set_multiplier(3)\n        counter.increment(2)\n        def test_shared_state():\n            operation_a()\n            operation_b()\n            operation_c()\n            return counter.get_value()\n\n# What value is returned by test_shared_state()?",\n        "answer": 25
    }
}

```

4D - 副作用推理 [Side Effect] (8)

I/O操作、异常处理、内存操作、时间依赖

```

{
    "id": "MF-SE-S001",
    "metadata": {
        "name": "MultiFunc-SideEffect-IOOperation",
        "category": "MultiFunc-Level",
        "subcategory": "Side Effect"
    }
}

```

```

        "category": "MultiFunc-Level",
        "subcategory": "Side Effect",
        "type": "seed",
        "source": "CodeSense-util-linux",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1
    },
    "task": {
        "description": "Analyze I/O side effects on global state. What is the final value of bytes_written?",
        "code": "int bytes_written = 0;\n\nint write_data(const char* data) {\n    int len =\n        strlen(data);\n    bytes_written += len;\n    // Simulated file write\n    printf(\"Writing %d bytes\\n\", len);\n    return len;\n}\n\nint log_operation(const char* operation, int value) {\n    char buffer[100];\n    sprintf(buffer, \"%s: %d\\n\", operation, value);\n    return write_data(buffer);\n}\n\nint process_with_logging() {\n    log_operation(\"START\", 100);\n    log_operation(\"PROCESS\", 200);\n    log_operation(\"END\", 300);\n    return bytes_written;\n}\n\n// What value is returned by process_with_logging()?",
        "answer": 33
    }
}

```

```

{
    "id": "MF-SE-S002",
    "metadata": {
        "name": "MultiFunc-SideEffect-FileOperations",
        "category": "MultiFunc-Level",
        "subcategory": "Side Effect",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Track file operation side effects on program state. What is the final line count?",
        "code": "line_count = 0\n\ndef write_line(content):\n    global line_count\n    line_count += 1\n    # Simulated file write\n    print(f\"Line {line_count}: {content}\")\n\n    return line_count\n\ndef write_header(title):\n    write_line(f\"{title}\")\n\n    return line_count\n\ndef write_content(items):\n    for item in items:\n        write_line(f\"- {item}\")\n\n    return line_count\n\n    def generate_report():\n        write_header(\"Report\")\n        write_content(['Item 1', 'Item 2', 'Item 3'])\n        write_line(\"End of report\")\n\n        return line_count\n\n    # What value is returned by generate_report()?",
        "answer": 6
    }
}

```

```
{
    "id": "MF-SE-S003",
    "metadata": {

```

```

        "name": "MultiFunc-SideEffect-ExceptionHandling",
        "category": "MultiFunc-Level",
        "subcategory": "Side Effect",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2
    },
    "task": {
        "description": "Analyze exception handling side effects on program state. What is the final error_count?",
        "code": "error_count = 0\nsuccess_count = 0\n\ndef risky_operation(value):\n    global error_count, success_count\n    try:\n        if value < 0:\n            raise ValueError(\"Negative value\")\n        if value > 100:\n            raise ValueError(\"Value too large\")\n        success_count += 1\n    return value * 2\n\nexcept ValueError:\n    error_count += 1\n\nreturn 0\n\n\ndef process_batch(values):\n    results = []\n    for value in values:\n        result = risky_operation(value)\n        results.append(result)\n    return error_count\n\n# What value is returned by process_batch([10, -5, 150, 25, 200])?",
        "answer": 3
    }
}

```

```

{
    "id": "MF-SE-S004",
    "metadata": {
        "name": "MultiFunc-SideEffect-NestedExceptions",
        "category": "MultiFunc-Level",
        "subcategory": "Side Effect",
        "type": "seed",
        "source": "Manual",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3
    },
    "task": {
        "description": "Track nested exception handling effects. What is the final cleanup_count?",
        "code": "cleanup_count = 0\nprocessed_count = 0\n\n\ndef cleanup_resource():\n    global cleanup_count\n    cleanup_count += 1\n\ndef process_item(item):\n    global processed_count\n    try:\n        if item['type'] == 'error':\n            raise Exception(\"Processing error\")\n        processed_count += 1\n    return item['value'] * 2\n\n    finally:\n        cleanup_resource()\n\ndef safe_process(item):\n    try:\n        return process_item(item)\n    except Exception:\n        return 0\n\ndef batch_process():\n    items = [\n        {'type': 'normal', 'value': 10},\n        {'type': 'error', 'value': 20},\n        {'type': 'normal', 'value': 30},\n        {'type': 'error', 'value': 40}\n    ]\n    for item in items:\n        safe_process(item)\n    return cleanup_count\n\n# What value is returned by batch_process()?",
        "answer": 4
    }
}

```

```
{
  "id": "MF-SE-S005",
  "metadata": {
    "name": "MultiFunc-SideEffect-MemoryOps",
    "category": "MultiFunc-Level",
    "subcategory": "Side Effect",
    "type": "seed",
    "source": "CodeSense-glibc",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2
  },
  "task": {
    "description": "Track memory allocation side effects. What is the final allocation count?",
    "code": "int allocation_count = 0;\nint total_allocated_bytes = 0;\n\nvoid* allocate_memory(size_t size) {\n    allocation_count++;\n    total_allocated_bytes += size;\n    return malloc(size);\n}\n\nvoid free_memory(void* ptr) {\n    if (ptr != NULL)\n        allocation_count--;\n    free(ptr);\n}\n\nint process_data_blocks()\n{\n    void* block1 = allocate_memory(100);\n    void* block2 = allocate_memory(200);\n    void* block3 = allocate_memory(150);\n    \n    free_memory(block1);\n    \n    void* block4 = allocate_memory(300);\n    \n    free_memory(block2);\n    free_memory(block3);\n    \n    return allocation_count;\n}\n\n// What value is returned by process_data_blocks()?",
    "answer": 1
  }
}
```

```
{
  "id": "MF-SE-S006",
  "metadata": {
    "name": "MultiFunc-SideEffect-MemoryModification",
    "category": "MultiFunc-Level",
    "subcategory": "Side Effect",
    "type": "seed",
    "source": "Manual",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3
  },
  "task": {
    "description": "Analyze complex memory modification side effects. What is the final value at global_buffer[2]?",
    "code": "int global_buffer[5] = {1, 2, 3, 4, 5};\nint modification_count = 0;\n\nvoid modify_buffer_element(int index, int value) {\n    if (index >= 0 && index < 5)\n        global_buffer[index] = value;\n    modification_count++;\n}\n\nvoid apply_transformation(int multiplier) {\n    for (int i = 0; i < 5; i++) {\n        modify_buffer_element(i, global_buffer[i] * multiplier);\n    }\n}\n\nvoid selective_update() {\n    for (int i = 1; i < 4; i++) {\n        modify_buffer_element(i, global_buffer[i] + i * 10);\n    }
}\n\nint process_buffer() {\n    apply_transformation(2);\n    selective_update();\n    return global_buffer[2];
}\n\n// What value is returned by process_buffer()?",
    "answer": 26
}
```

```
    }\n}
```

```
{\n    "id": "MF-SE-S007",\n    "metadata": {\n        "name": "MultiFunc-SideEffect-TimeDependency",\n        "category": "MultiFunc-Level",\n        "subcategory": "Side Effect",\n        "type": "seed",\n        "source": "Manual",\n        "language": "python",\n        "difficulty": "hard",\n        "intervention": 2\n    },\n    "task": {\n        "description": "Simulate time-dependent operations. What is the final timestamp difference?",\n        "code": "import time\n\nclass TimeTracker:\n    def __init__(self):\n        self.start_time = 100 # Simulated timestamp\n        self.operations = []\n    def record_operation(self, name, duration):\n        # Simulate time passage\n        self.start_time += duration\n        self.operations.append({\n            'name': name,\n            'timestamp': self.start_time,\n            'duration': duration\n        })\n    return self.start_time\n\ntracker = TimeTracker()\n\ndef operation_a():\n    return\n\ntracker.record_operation('A', 5)\n\ndef operation_b():\n    return\n\ntracker.record_operation('B', 10)\n\ndef operation_c():\n    return\n\ntracker.record_operation('C', 7)\n\ndef execute_sequence():\n    start =\n    tracker.start_time\n    operation_a()\n    operation_b()\n    operation_c()\n    end =\n    tracker.start_time\n    return end - start\n\n# What value is returned by\nexecute_sequence()?",\n        "answer": 22\n    }\n}
```

```
{\n    "id": "MF-SE-S008",\n    "metadata": {\n        "name": "MultiFunc-SideEffect-ConcurrentAccess",\n        "category": "MultiFunc-Level",\n        "subcategory": "Side Effect",\n        "type": "seed",\n        "source": "Manual",\n        "language": "python",\n        "difficulty": "expert",\n        "intervention": 3\n    },\n    "task": {\n        "description": "Simulate concurrent access effects on shared state. What is the final counter value?",\n        "code": "class Counter:\n    def __init__(self):\n        self.value = 0\n\n    def increment(self, amount):\n        self.value += amount\n\n    def decrement(self, amount):\n        self.value -= amount\n\n    def get_value(self):\n        return self.value\n\n# Create a shared counter\nshared_counter = Counter()\n\n# Define two threads\nfrom threading import Thread\n\ndef thread1(shared_counter):\n    for _ in range(1000):\n        shared_counter.increment(1)\n\ndef thread2(shared_counter):\n    for _ in range(1000):\n        shared_counter.decrement(1)\n\n# Start the threads\nThread(target=thread1, args=(shared_counter,)).start()\nThread(target=thread2, args=(shared_counter,)).start()\n\n# Wait for threads to finish\nThread(target=thread1, args=(shared_counter,)).join()\nThread(target=thread2, args=(shared_counter,)).join()\n\n# Print the final value\nprint(shared_counter.get_value())\n\n# Expected output: 0\n\n# What value is returned by\nshared_counter.get_value()?",\n        "answer": 0\n    }\n}
```

```

    "code": "class SharedResource:\n        def __init__(self):\n            self.counter = 0\n            self.lock_count = 0\n        def acquire_lock(self):\n            self.lock_count += 1\n        def release_lock(self):\n            if self.lock_count > 0:\n                self.lock_count -= 1\n        def safe_increment(self, value):\n            self.acquire_lock()\n            self.counter += value\n            self.release_lock()\n    return self.counter\n\nresource = SharedResource()\n\n\nresource.safe_increment(5)\nresource.safe_increment(3)\nresource.safe_increment(7)\nresource.safe_increment(2)\n\nresult = resource.safe_increment(5) + resource.safe_increment(3) + resource.safe_increment(7) + resource.safe_increment(2)\n\nprint(result)\n\n# What value is returned by simulate_concurrent_work()?",\n    "answer": 17\n}\n}

```

4E - 大混合 (2)

```

{
    "id": "MF-MIX-S001",
    "metadata": {
        "name": "MultiFunc-Mix-Comprehensive",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "seed",
        "source": "Manual",
        "language": "c",
        "difficulty": "master",
        "intervention": 4
    },
    "task": {
        "description": "Comprehensive multi-function analysis combining call chains, parameter passing, state propagation, and side effects. What is the final computed result?",\n    }
}

```

```

    "code": "#include <stdio.h>\n#include <stdlib.h>\n#include <string.h>\n\n// Global\nstate\nstatic int global_operation_count = 0;\nstatic char global_log[256] = \"\";\n\n// Function pointer type\ntypedef int (*processor_func)(int);\n\n// Callback functions\nint\ndouble_processor(int value) {\n    return value * 2;\n}\n\nint square_processor(int value)\n{\n    return value * value;\n}\n\n// State management functions\nvoid log_operation(const\nchar* op_name, int value) {\n    char temp[64];\n    sprintf(temp, "%s:%d", op_name,\nvalue);\n    strcat(global_log, temp);\n    global_operation_count++;\n}\n\n// Recursive function with state modification\nint recursive_calculate(int n, int depth) {\n    log_operation(\"REC\", n);\n    if (depth <= 0 || n <= 1) {\n        return n;\n    }\n    return n + recursive_calculate(n - 1, depth - 1);\n}\n\n// Function with complex parameter passing\nint process_array_with_callback(int *arr, int size,\nprocessor_func processor) {\n    int sum = 0;\n    for (int i = 0; i < size; i++) {\n        int processed = processor(arr[i]);\n        arr[i] = processed; // Side effect: modify\n        original array\n        sum += processed;\n    }\n    log_operation(\"PROC\", processed);\n    return sum;\n}\n\n// Function with multiple side effects\nint*\nallocate_and_process(int initial_value) {\n    static int allocation_count = 0;\n    allocation_count++;\n    int *result = malloc(sizeof(int) * 3);\n    if (result ==\nNULL) {\n        log_operation(\"ERROR\", -1);\n        return NULL;\n    }\n    result[0] = recursive_calculate(initial_value, 2);\n    result[1] =\n    process_array_with_callback(&result[0], 1, double_processor);\n    result[2] =\n    allocation_count;\n    log_operation(\"ALLOC\", allocation_count);\n    return\n    result;\n}\n\n// Main computation function\nint complex_computation()\n{\n    int\ntest_array[3] = {3, 4, 5};\n    // Chain of operations with various effects\n    int\n    sum1 = process_array_with_callback(test_array, 3, square_processor);\n    int\n    *dynamic_data = allocate_and_process(4);\n    if (dynamic_data == NULL) {\n        return\n        -1;\n    }\n    int sum2 = dynamic_data[0] + dynamic_data[1] + dynamic_data[2];\n    // Modify array through pointer\n    test_array[1] = test_array[1] +\n    dynamic_data[2];\n    int final_result = sum1 + sum2 + global_operation_count;\n    free(dynamic_data);\n    log_operation(\"FINAL\", final_result);\n    return\n    final_result;\n}\n\n// what value is returned by complex_computation()?",\n    "answer": 160\n}\n}\n

```

```

{
    "id": "MF-MIX-S002",
    "metadata": {
        "name": "MultiFunc-Mix-Realworld",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "seed",
        "source": "CodeSense-postgresql",
        "language": "python",
        "difficulty": "grandmaster",
        "intervention": 5
    },
    "task": {
        "description": "Real-world scenario: Database transaction processing system with\ncomplex inter-function dependencies. What is the final transaction_id?",\n
    }
}

```

```

"code": "class TransactionManager:\n    def __init__(self):\n        self.next_transaction_id = 1000\n        self.active_transactions = {}\n        self.completed_transactions = []\n        self.error_count = 0\n\n    def begin_transaction(self, user_id):\n        tx_id = self.next_transaction_id\n        self.next_transaction_id += 1\n        self.active_transactions[tx_id] = {\n            'user_id': user_id,\n            'operations': [],\n            'status': 'active'\n        }\n        return tx_id\n\n    def add_operation(self, tx_id, operation):\n        if tx_id in self.active_transactions:\n            self.active_transactions[tx_id]['operations'].append(operation)\n            return True\n        return False\n\n    def commit_transaction(self, tx_id):\n        if tx_id in self.active_transactions:\n            tx = self.active_transactions.pop(tx_id)\n            tx['status'] = 'committed'\n            self.completed_transactions.append(tx)\n            return len(tx['operations'])\n        return 0\n\n    def rollback_transaction(self, tx_id):\n        if tx_id in self.active_transactions:\n            self.active_transactions.pop(tx_id)\n            self.error_count += 1\n            return True\n        return False\n\n# Global transaction manager\ntm = TransactionManager()\n\n# Account management\naccounts = {\n    'alice': 1000,\n    'bob': 500,\n    'charlie': 750\n}\n\ndef validate_transfer(from_account, to_account, amount):\n    if from_account not in accounts or to_account not in accounts:\n        return False\n    if accounts[from_account] < amount:\n        return False\n    if amount <= 0:\n        return False\n    return True\n\ndef execute_transfer(tx_id, from_account, to_account, amount):\n    try:\n        if not validate_transfer(from_account, to_account, amount):\n            tm.rollback_transaction(tx_id)\n            return False\n\n        # Record operations\n        tm.add_operation(tx_id, f'debit:{from_account}:{amount}\n')\n        tm.add_operation(tx_id, f'credit:{to_account}:{amount}\n')\n\n        # Execute transfer\n        accounts[from_account] -= amount\n        accounts[to_account] += amount\n\n        return True\n    except Exception:\n        tm.rollback_transaction(tx_id)\n        return False\n\ndef batch_transfer(transfers):\n    results = []\n    for transfer in transfers:\n        from_acc, to_acc, amount = transfer\n\n        # Begin transaction\n        tx_id = tm.begin_transaction(from_acc)\n\n        # Execute transfer\n        success = execute_transfer(tx_id, from_acc, to_acc, amount)\n\n        operation_count = tm.commit_transaction(tx_id)\n\n        results.append(operation_count)\n    else:\n        results.append(0)\n\n    return results\n\ndef calculate_statistics():\n    total_completed = len(tm.completed_transactions)\n    total_operations = sum(len(tx['operations']) for tx in tm.completed_transactions)\n    error_rate = tm.error_count / max(tm.next_transaction_id - 1000, 1)\n\n    return {\n        'completed': total_completed,\n        'operations': total_operations,\n        'error_rate': error_rate,\n        'next_id': tm.next_transaction_id\n    }\n\n# Define transfer batch\ntransfers = [\n    ('alice', 'bob', 200),\n    ('charlie', 'bob', 100),\n    ('charlie', 'alice', 1000),\n    ('alice', 'charlie', 300),\n    ('bob', 'alice', 800),\n    ('charlie', 'bob', 50)\n]\n\n# Process transfers\nbatch_results = batch_transfer(transfers)\n\n# Calculate final statistics\nstats = calculate_statistics()\n\nfinal_result = (\n    stats['next_id'] +\n    stats['completed'] * 10 +\n    stats['operations'] * 5 +\n    int(stats['error_rate'] * 100) +\n    sum(batch_results)\n)\n\nreturn final_result\n\n# What value is returned by process_banking_scenario()?",\n    "answer": 1114
}

```

变式生成叶子任务构建

1 - 语句级推理 [Statement-Level] (323)

1A - 算数运算 [Arithmetic] (49)

提示词

任务目标

基于给定的算数运算种子任务，生成多样化的变式任务来扩展测试覆盖面，确保对大模型算术推理能力进行全面评估。

变式生成维度

1. 运算复杂度变式

- **简化变式**: 将复合运算分解为单一运算
- **复杂化变式**: 增加运算层数和嵌套深度
- **混合运算变式**: 组合不同类型的算术运算 (四则运算+位运算+幂运算)

2. 数据类型变式

- **整数变式**: 使用不同范围的整数 (小整数、大整数、负数、零)
- **浮点数变式**: 精度变化、科学计数法、极值处理
- **类型转换变式**: int/float/decimal互转，截断/舍入行为
- **特殊值变式**: NaN、Infinity、极小值、边界值

3. 运算符优先级变式

- **括号变式**: 添加/移除括号改变运算顺序
- **优先级混淆变式**: 故意设计易错的优先级组合
- **链式运算变式**: 多个同级运算符连续使用

4. 位运算特化变式

- **位操作变式**: 左移、右移、与、或、异或、取反
- **掩码操作变式**: 位掩码设置、清除、检查
- **二进制表示变式**: 不同进制表示 (0b、0o、0x)

5. 变量依赖变式

- **直接计算变式**: 所有值都是字面量
- **单级依赖变式**: 使用一层变量引用
- **多级依赖变式**: 变量间存在复杂依赖关系
- **循环依赖变式**: 变量更新后的累积效果

6. 语言移植变式

- **Python -> Java**: 考虑类型声明、除法行为差异
- **Python -> C++**: 考虑类型转换、溢出行为

- **Python -> JavaScript:** 考虑弱类型转换规则

7. 边界条件变式

- **溢出变式:** 整数溢出、浮点溢出
- **精度损失变式:** 浮点精度问题
- **除零变式:** 除法和取模的特殊情况
- **负数变式:** 负数的幂运算、位运算

8. 错误导向变式

- **常见错误变式:** 设计容易产生错误理解的表达式
- **陷阱变式:** 利用运算符优先级、类型转换等设置陷阱
- **边界错误变式:** 边界值的特殊行为

生成规则

基本要求

1. 每个种子任务生成8-12个变式
2. 确保每个变式维度至少覆盖2-3个变式
3. 保持原始问题的核心语义，但改变实现细节
4. 变式难度应该涵盖easy/medium/hard三个等级

难度分级标准

- **Easy:** 单一运算类型，直接字面量，无类型转换
- **Medium:** 2-3种运算类型组合，1-2层变量依赖，简单类型转换
- **Hard:** 复杂运算嵌套，多层变量依赖，特殊值处理，跨类型操作

输出格式

为每个变式生成完整的JSON格式，包含：

- 唯一ID（基于种子ID + 变式类型）
- 完整的metadata（标注变式类型和难度）
- 清晰的task描述和代码
- 正确且唯一的答案

质量检查要点

1. **正确性:** 确保答案计算正确
2. **多样性:** 避免简单的数值替换，要有结构性变化
3. **代表性:** 每个变式应该测试特定的推理能力
4. **渐进性:** 从简单到复杂有合理的难度梯度

示例变式生成

基于种子任务SL-AR-S005，生成以下变式：

- 简化变式：分解为单个运算步骤

- 类型转换变式: 引入浮点数和类型转换
- 位运算变式: 替换部分算术运算为位运算
- 边界值变式: 使用可能导致溢出的大数值
- 优先级陷阱变式: 设计容易误解的运算符组合

请基于此提示词, 为给定的种子任务生成完整的变式集合。

下面是6个种子任务

```
{
  "id": "SL-AR-S001-V1",
  "metadata": {
    "name": "StatementLevel-Arithmetic-S001-Simplified",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "simplified"
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
    "code": "mid = 3\nj = mid + 1",
    "answer": 4
  }
},
{
  "id": "SL-AR-S001-V2",
  "metadata": {
    "name": "StatementLevel-Arithmetic-S001-NegativeNumbers",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "data_type"
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
    "code": "a = [-5, -2, 0, 1, 3, 8, 12]\nlo = 0\nmid = -2\nhi = 4\nni = 1\nj = mid + 1",
    "answer": -1
  }
},
{
  "id": "SL-AR-S001-V3",
  "metadata": {
    "name": "StatementLevel-Arithmetic-S001-ComplexOperations",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "operator"
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
    "code": "a = [-5, -2, 0, 1, 3, 8, 12]\nlo = 0\nmid = -2\nhi = 4\nni = 1\nj = mid + 1",
    "answer": -1
  }
}
```

```
"metadata": {
    "name": "StatementLevel-Arithmetic-S001-FloatConversion",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0,
    "variant_type": "type_conversion"
},
"task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = int(mid + 1.5)`?",
    "code": "a = [1.5, 2.3, 4.7, 5.1, 6.8]\nlo = 0\nmid = 2.7\nhi = 4\ni = 1\nj = int(mid + 1.5)",
    "answer": 4
},
{
    "id": "SL-AR-S001-v4",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S001-ComplexExpression",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complexity"
},
"task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = (mid + 1) * 2 - lo // 2`?",
    "code": "a = [1, 2, 4, 4, 5, 6, 7, 23, 8, 9]\nlo = 4\nmid = 3\nhi = 7\ni = 1\nj = (mid + 1) * 2 - lo // 2",
    "answer": 6
},
{
    "id": "SL-AR-S001-v5",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S001-BitOperations",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "bit_operations"
}
}
```

```
"task": {
    "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = (mid + 1) + 1`?",
    "code": "a = [1, 2, 4, 4, 5, 6, 7, 23, 8, 9]\nlo = 0\nmid = 2\nhi = 7\ni = lo\nj = (mid + 1) + 1",
    "answer": 4
},
{
    "id": "SL-AR-S001-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S001-overflow",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
        "code": "a = list(range(1000))\nlo = 0\nmid = 999999999999\nhi = 1000000000000\ni = lo\nj = mid + 1",
        "answer": 1000000000000
    }
},
{
    "id": "SL-AR-S001-V7",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S001-PrecedenceTrap",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "precedence_trap"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1 * 2 + 3`?",
        "code": "a = [1, 2, 4, 4, 5, 6, 7, 23, 8, 9]\nlo = 0\nmid = 3\nhi = 7\ni = lo\nj = mid + 1 * 2 + 3",
        "answer": 8
    }
},
{
    "id": "SL-AR-S001-V8",
    "metadata": {
```

```
        "name": "StatementLevel-Arithmetic-S001-MultiLevel",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "multi_dependency"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = offset + 1`?",
        "code": "a = [1, 2, 4, 4, 5, 6, 7, 23, 8, 9]\nlo = 0\nbase = 3\nmid = base +\nlo\noffset = mid\nhi = 7\ni = lo\nj = offset + 1",
        "answer": 4
    }
},
{
    "id": "SL-AR-S001-V9",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S001-ZeroEdge",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable j after executing the assignment statement `j = mid + 1`?",
        "code": "a = [0, 0, 0, 0, 0]\nlo = 0\nmid = -1\nhi = 3\ni = lo\nj = mid + 1",
        "answer": 0
    }
},
{
    "id": "SL-AR-S002-V1",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S002-Simplified",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simplified"
    },
    "task": {
```



```
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complexity"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = (total * 2 + bonus) // (count + extra)`?",
        "code": "items = [15, 25, 30, 20]\ntotal = sum(items)\ncount = len(items)\nbonus = 10\nextra = 1\nresult = (total * 2 + bonus) // (count + extra)\nremainder = total % count",
        "answer": 38
    }
},
{
    "id": "SL-AR-S002-V5",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S002-ZeroDivision",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = total // count` when count is non-zero?",
        "code": "items = [100]\ntotal = sum(items)\ncount = len(items)\nresult = total // count\nremainder = total % count",
        "answer": 100
    }
},
{
    "id": "SL-AR-S002-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S002-Bitshift",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "bit_operations"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = total >> shift`?",
        "code": "total = 100\nshift = 2\nresult = total >> shift"
    }
}
```



```
"type": "variant",
"source": "Generated",
"language": "python",
"difficulty": "easy",
"intervention": 0,
"variant_type": "simplified"
},
"task": {
    "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** 2`?",
    "code": "radius = 5\narea = 3.14159 * radius ** 2",
    "answer": 78.53975
},
{
    "id": "SL-AR-S003-V2",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S003-IntegerRadius",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "data_type"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = int(3.14159 * radius ** 2)`?",
        "code": "import math\nradius = 3\n\ndiameter = 2 * radius\n\ncircumference = 2 * math.pi * radius\n\narea = int(3.14159 * radius ** 2)\n\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 28
    }
},
{
    "id": "SL-AR-S003-V3",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S003-HighPrecision",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "precision_enhancement"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = math.pi * radius ** 2`?",
        "code": "import math\n\nradius = 5\n\ndiameter = 2 * radius\n\ncircumference = 2 * math.pi * radius\n\narea = math.pi * radius ** 2\n\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 78.53975
    }
}
```

```
        "answer": 78.53981633974483
    }
},
{
    "id": "SL-AR-S003-V4",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S003-ComplexFormula",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complexity"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * (radius + offset) ** 2 - 3.14159 * offset ** 2`?",
        "code": "import math\nradius = 3\noffset = 2\ndiameter = 2 * radius\n circumference = 2 * math.pi * radius\narea = 3.14159 * (radius + offset) ** 2 - 3.14159 * offset ** 2\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 40.840299999999996
    }
},
{
    "id": "SL-AR-S003-V5",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S003-NegativeRadius",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** 2`?",
        "code": "import math\nradius = -5\ndiameter = 2 * radius\n circumference = 2 * math.pi * radius\narea = 3.14159 * radius ** 2\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 78.53975
    }
},
{
    "id": "SL-AR-S003-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S003-FloatRadius",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    }
}
```

```
"type": "variant",
"source": "Generated",
"language": "python",
"difficulty": "medium",
"intervention": 0,
"variant_type": "data_type"
},
"task": {
  "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** 2`?",
  "code": "import math\nradius = 2.5\n\ndiameter = 2 * radius\n\ncircumference = 2 * math.pi * radius\n\narea = 3.14159 * radius ** 2\n\nvolume = (4/3) * math.pi * radius ** 3",
  "answer": 19.634937500000002
}
},
{
  "id": "SL-AR-S003-V7",
  "metadata": {
    "name": "StatementLevel-Arithmetic-S003-ZeroRadius",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "boundary_conditions"
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** 2`?",
    "code": "import math\nradius = 0\n\ndiameter = 2 * radius\n\ncircumference = 2 * math.pi * radius\n\narea = 3.14159 * radius ** 2\n\nvolume = (4/3) * math.pi * radius ** 3",
    "answer": 0.0
  }
},
{
  "id": "SL-AR-S003-V8",
  "metadata": {
    "name": "StatementLevel-Arithmetic-S003-PowerVariation",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "hard",
    "intervention": 0,
    "variant_type": "complexity"
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = 3.14159 * radius ** (1 + 1)`?",
    "code": "import math\nradius = 2\n\narea = 3.14159 * radius ** (1 + 1)",
    "answer": 12.566370614359172
  }
}
```

```

        "code": "import math\nradius = 5\n\ndiameter = 2 * radius\n\ncircumference = 2 * math.pi\n* radius\n\narea = 3.14159 * radius ** (1 + 1)\n\nvolume = (4/3) * math.pi * radius ** 3",
        "answer": 78.53975
    },
},
{
    "id": "SL-AR-S004-V1",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-Simplified",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simplified"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | 4`?",
        "code": "flags = 0b1010\nmask = flags | 4",
        "answer": 14
    }
},
{
    "id": "SL-AR-S004-V2",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-HexRepresentation",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "data_representation"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << position)`?",
        "code": "flags = 0xA\nposition = 2\nmask = flags | (1 << position)\n\ncheck = mask & (1 << position)\n\ntoggle = flags ^ (1 << position)",
        "answer": 14
    }
},
{
    "id": "SL-AR-S004-V3",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-XORoperation",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",

```

```
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "bit_operations"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags ^ (1 << position)`?",
        "code": "flags = 0b1010\nposition = 2\nmask = flags ^ (1 << position)\ncheck = mask & (1 << position)\ntoggle = flags ^ (1 << position)",
        "answer": 14
    }
},
{
    "id": "SL-AR-S004-V4",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-ANDOperation",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "bit_operations"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags & ~(1 << position)`?",
        "code": "flags = 0b1110\nposition = 2\nmask = flags & ~(1 << position)\ncheck = mask & (1 << position)\ntoggle = flags ^ (1 << position)",
        "answer": 10
    }
},
{
    "id": "SL-AR-S004-V5",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-MultiplePositions",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complexity"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << position1) | (1 << position2)`?",
        "code": "flags = 0b1010\nposition1 = 2\nposition2 = 3\nmask = flags | (1 << position1) | (1 << position2)\ncheck = mask & (1 << position1)\ntoggle = flags ^ (1 << position2)"
    }
}
```

```
        "code": "flags = 0b1010\nposition1 = 2\nposition2 = 0\nmask = flags | (1 << position1) | (1 << position2)\ncheck = mask & (1 << position1)\ntoggle = flags ^ (1 << position1)",
        "answer": 15
    },
},
{
    "id": "SL-AR-S004-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-NegativePosition",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << abs(position))`?",
        "code": "flags = 0b1010\nposition = -2\nmask = flags | (1 << abs(position))\ncheck = mask & (1 << abs(position))\ntoggle = flags ^ (1 << abs(position))",
        "answer": 14
    }
},
{
    "id": "SL-AR-S004-V7",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-Largeshift",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << position)`?",
        "code": "flags = 0b1010\nposition = 8\nmask = flags | (1 << position)\ncheck = mask & (1 << position)\ntoggle = flags ^ (1 << position)",
        "answer": 266
    }
},
{
    "id": "SL-AR-S004-V8",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S004-ZeroFlags",
        "category": "Statement-Level",
        "subcategory": "Arithmetic"
    }
}
```

```
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable mask after executing the statement `mask = flags | (1 << position)`?",
        "code": "flags = 0b0000\nposition = 2\nmask = flags | (1 << position)\ncheck = mask & (1 << position)\ntoggle = flags ^ (1 << position)",
        "answer": 4
    }
},
{
    "id": "SL-AR-S005-v1",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-Simplified",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simplified"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2`?",
        "code": "base = 10\nresult = base * 2",
        "answer": 20
    }
},
{
    "id": "SL-AR-S005-v2",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-ParenthesesChange",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "precedence_trap"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = (base * 2 + offset // 3 - power ** 2) % 5`?",
        "code": "(base * 2 + offset // 3 - power ** 2) % 5"
    }
}
```

```
        "code": "base = 10\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset //\n3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = (base * 2 + offset // 3 - power ** 2) %\n5",
        "answer": 2
    }
},
{
    "id": "SL-AR-S005-V3",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-FloatOperations",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "type_conversion"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2.0 + offset / 3.0 - power ** 2 % 5`?",
        "code": "base = 10\noffset = 17\npower = 3\nntemp1 = base * 2.0\nntemp2 = offset / 3.0\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2.0 + offset / 3.0 - power ** 2 % 5",
        "answer": 21.666666666666668
    }
},
{
    "id": "SL-AR-S005-V4",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-NegativeNumbers",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "data_type"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset // 3 - power ** 2 % 5`?",
        "code": "base = -10\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset //\n3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2 + offset // 3 - power ** 2 % 5",
        "answer": -19
    }
},
{
    "id": "SL-AR-S005-V5",
    "metadata": {
```

```
        "name": "StatementLevel-Arithmetic-S005-Bitoperations",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "bit_operations"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base << 1 + offset // 3 - power ** 2 % 5`?",
        "code": "base = 10\noffset = 17\npower = 3\nntemp1 = base << 1\nntemp2 = offset // 3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base << 1 + offset // 3 - power ** 2 % 5",
        "answer": 21
    }
},
{
    "id": "SL-AR-S005-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-LargeNumbers",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset // 3 - power ** 2 % 5`?",
        "code": "base = 1000000\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset // 3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2 + offset // 3 - power ** 2 % 5",
        "answer": 2000001
    }
},
{
    "id": "SL-AR-S005-V7",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-Zerovalues",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
}
```

```
"task": {
    "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset // 3 - power ** 2 % 5`?",
    "code": "base = 0\noffset = 17\npower = 3\nntemp1 = base * 2\nntemp2 = offset // 3\nntemp3 = power ** 2\nntemp4 = temp3 % 5\nresult = base * 2 + offset // 3 - power ** 2 % 5",
    "answer": 1
},
{
    "id": "SL-AR-S005-V8",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S005-DependencyChain",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "multi_dependency"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = final_base * 2 + final_offset // 3 - final_power ** 2 % 5`?",
        "code": "initial = 5\nbase = initial * 2\noffset = 17\npower = 3\nfinal_base = base\nfinal_offset = offset\nfinal_power = power\nntemp1 = final_base * 2\nntemp2 = final_offset // 3\nntemp3 = final_power ** 2\nntemp4 = temp3 % 5\nresult = final_base * 2 + final_offset // 3 - final_power ** 2 % 5",
        "answer": 21
    }
},
{
    "id": "SL-AR-S006-V1",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-Simplified",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simplified"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(70.4)`?",
        "code": "average = 70.4\nfinal_score = int(average)",
        "answer": 70
    }
},
{
```

```
"id": "SL-AR-S006-V2",
"metadata": {
    "name": "StatementLevel-Arithmetic-S006-RoundingBehavior",
    "category": "Statement-Level",
    "subcategory": "Arithmetic",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0,
    "variant_type": "type_conversion"
},
"task": {
    "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = round(average * weight + bonus)`?",
    "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nfinal_score = round(average * weight + bonus)\nrounded_score = round(average * weight + bonus)",
    "answer": 76
},
{
    "id": "SL-AR-S006-V3",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-FloorDivision",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "type_conversion"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight // 1 + bonus)`?",
        "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nfinal_score = int(average * weight // 1 + bonus)\nrounded_score = round(average * weight + bonus)",
        "answer": 75
    }
},
{
    "id": "SL-AR-S006-V4",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-NegativeBonus",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
```

```
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "data_type"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight + bonus)`?",
        "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = -5.5\nfinal_score = int(average * weight + bonus)\nrounded_score = round(average * weight + bonus)",
        "answer": 64
    }
},
{
    "id": "SL-AR-S006-V5",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-HighPrecision",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "precision_enhancement"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight * precision + bonus * precision) // precision`?",
        "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nprecision = 1000\nfinal_score = int(average * weight * precision + bonus * precision) // precision\nrounded_score = round(average * weight + bonus)",
        "answer": 75
    }
},
{
    "id": "SL-AR-S006-V6",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-Multipleweights",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complexity"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight1 * weight2 + bonus)`?",
        "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight1 = 0.8\nweight2 = 0.5\nbonus = 5.5\nfinal_score = int(average * weight1 * weight2 + bonus)"
    }
}
```

```

        "code": "scores = [85, 92, 78, 96]\naverage = sum(scores) / len(scores)\nweight1 = 0.9\nweight2 = 0.8\nbonus = 5.5\nfinal_score = int(average * weight1 * weight2 + bonus)\nrounded_score = round(average * weight1 * weight2 + bonus)",
        "answer": 68
    },
},
{
    "id": "SL-AR-S006-V7",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-EmptyScores",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "boundary_conditions"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average * weight + bonus)`?",
        "code": "scores = [100]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nfinal_score = int(average * weight + bonus)\nrounded_score = round(average * weight + bonus)",
        "answer": 85
    }
},
{
    "id": "SL-AR-S006-V8",
    "metadata": {
        "name": "StatementLevel-Arithmetic-S006-BitshiftConversion",
        "category": "Statement-Level",
        "subcategory": "Arithmetic",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "bit_operations"
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable final_score after executing the statement `final_score = int(average) >> 0 + int(bonus)`?",
        "code": "scores = [80, 90, 70, 100]\naverage = sum(scores) / len(scores)\nweight = 0.8\nbonus = 5.5\nfinal_score = int(average) >> 0 + int(bonus)\nrounded_score = round(average * weight + bonus)",
        "answer": 90
    }
}

```

1B - 布尔运算 [Boolean] (50)

提示词：

任务目标

基于给定的布尔运算种子任务，生成多样化的变式任务来全面测试大模型的布尔逻辑推理能力，涵盖比较运算、逻辑运算和短路求值等核心场景。

变式生成维度

1. 比较运算变式

- **比较运算符变式**：`==`, `!=`, `<`, `<=`, `>`, `>=` 的各种替换和组合
- **数值范围变式**：边界值比较 (0, -1, 1, 最大值, 最小值)
- **类型比较变式**：不同数据类型间的比较行为
- **特殊值比较变式**：`NULL/nullptr`, 0, `EOF`等特殊值的比较
- **十六进制比较变式**：`0x80`, `0xFF`等十六进制常量比较

2. 逻辑运算变式

- **逻辑运算符变式**：`&&(and)`, `||(or)`, `!(not)` 的组合使用
- **德摩根定律变式**：`!(A && B) ≈ (!A || !B)` 等价转换
- **运算符优先级变式**：混合使用 `&&/||` 时的优先级测试
- **嵌套逻辑变式**：多层括号的复杂逻辑表达式
- **三元运算符变式**：`condition ? true_val : false_val`

3. 短路求值变式

- **OR短路变式**：第一个条件为真时，第二个条件不被计算
- **AND短路变式**：第一个条件为假时，第二个条件不被计算
- **副作用短路变式**：包含函数调用或变量修改的短路情况
- **复合短路变式**：多个短路条件的组合

4. 空值和指针变式

- **空指针检查变式**：`NULL`判断的各种形式
- **指针比较变式**：指针地址的比较运算
- **空值逻辑变式**：`NULL`在逻辑运算中的行为
- **指针转换变式**：`void*`指针的比较和转换

5. 常量和宏变式

- **宏定义变式**：`#define`常量的比较和逻辑运算
- **枚举变式**：枚举值的比较运算
- **常量表达式变式**：`const`常量的布尔运算
- **字面量变式**：直接使用数字字面量vs使用命名常量

6. 语言移植变式

- **C -> Python变式**：考虑`Truthy`值、`None` vs `NULL`等差异
- **C -> Java变式**：考虑`boolean`类型、`null` vs `NULL`等差异
- **C -> JavaScript变式**：考虑`truthy/falsy`值、类型转换等差异

7. 复杂度变式

- **简化变式**：将复合条件分解为单一比较
- **复杂化变式**：增加更多逻辑运算符和条件
- **链式比较变式**：`a < b < c` 形式的连续比较
- **多分支变式**：`if-else if-else`的复杂条件判断

8. 边界和错误变式

- **边界值变式**: 临界值的比较 (如127 vs 128对于`signed char`)
- **溢出边界变式**: 整数溢出时的比较行为
- **精度问题变式**: 浮点数比较的精度问题
- **类型转换变式**: 隐式类型转换对比较结果的影响

9. 上下文依赖变式

- **条件上下文变式**: 改变`if`语句的上下文环境
- **返回值变式**: 不同返回值类型的布尔表达式
- **函数参数变式**: 作为函数参数的布尔表达式
- **循环条件变式**: 作为循环条件的布尔表达式

生成规则

基本要求

1. 每个种子任务生成10-15个变式
2. 确保每个变式维度至少覆盖2-3个变式
3. 保持原始布尔逻辑的核心, 但改变具体实现
4. 变式难度应该涵盖`easy/medium/hard`三个等级

难度分级标准

- **Easy**: 单一比较运算, 直接值比较, 无复杂逻辑
- **Medium**: 2-3个逻辑运算符组合, 简单短路求值, 基础指针比较
- **Hard**: 复杂嵌套逻辑, 多重短路求值, 特殊值处理, 类型转换陷阱

特殊考虑点

1. **C语言特性**: 指针运算、宏定义、类型转换规则
2. **真值表验证**: 确保逻辑运算结果符合真值表
3. **短路行为**: 准确模拟短路求值的执行顺序
4. **边界情况**: 特别关注`NULL`、`0`、`EOF`等特殊值

输出格式

为每个变式生成完整的JSON格式, 包含:

- 唯一ID (`SL-BL-S00X-VYY`格式)
- 完整的`metadata` (标注变式类型和源种子)
- 清晰的`task`描述和代码
- 正确的布尔值答案 (`true/false`)

质量检查要点

1. **逻辑正确性**: 确保布尔表达式计算正确
2. **语法合规性**: 确保C语言语法正确
3. **测试价值**: 每个变式应该测试特定的布尔推理能力
4. **陷阱设计**: 适当设计容易出错的逻辑陷阱

示例变式类型

基于`SL-BL-S001`的变式示例:

- **逻辑运算符变式**: 将 `||` 改为 `&&`
- **德摩根变式**: `!(inbuffer != NULL && insize != 0)`
- **短路求值变式**: 调换条件顺序测试短路行为
- **边界值变式**: `insize`使用`0`、`1`、`-1`等边界值
- **语言移植变式**: 转换为Python的`None`和`0`比较

基于SL-BL-S005的变式示例：

- **复杂化变式**：增加第三个状态的OR条件
- **宏展开变式**：直接使用数字替代宏定义
- **嵌套变式**：(state == IN_CHAR || state == IN_CHAR_SPACE) && len > 0
- **三元运算符变式**：使用?:运算符重写
- **类型转换变式**：使用不同的整数类型

请基于此提示词，为给定的布尔运算种子任务生成完整的变式集合，确保全面覆盖布尔逻辑推理的各个方面。

```
{  
  "id": "SL-BL-S001-v01",  
  "metadata": {  
    "name": "StatementLevel-Boolean-S001-Variant1",  
    "category": "Statement-Level",  
    "subcategory": "Boolean",  
    "type": "variant",  
    "source_seed": "SL-BL-S001",  
    "variant_type": "logical_operator",  
    "language": "c",  
    "difficulty": "medium",  
    "intervention": 0  
  },  
  "task": {  
    "description": "Given the following code snippet, what is the value of the boolean expression `inbuffer == NULL && insize == 0` when `inbuffer` points to a valid buffer and `insize` is 1024?",  
    "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nif (inbuffer == NULL && insize == 0) {\n    printf(\"Input validation failed\");\n} else {\n    printf(\"Input is valid\");\n}",  
    "answer": false  
  },  
},  
{  
  "id": "SL-BL-S001-v02",  
  "metadata": {  
    "name": "StatementLevel-Boolean-S001-Variant2",  
    "category": "Statement-Level",  
    "subcategory": "Boolean",  
    "type": "variant",  
    "source_seed": "SL-BL-S001",  
    "variant_type": "demorgan_law",  
    "language": "c",  
    "difficulty": "hard",  
    "intervention": 0  
  },  
  "task": {  
    "description": "Given the following code snippet, what is the value of the boolean expression `!(inbuffer != NULL && insize != 0)` when `inbuffer` points to a valid buffer and `insize` is 1024?",  
  }  
}
```



```
"metadata": {
    "name": "StatementLevel-Boolean-S001-Variant5",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S001",
    "variant_type": "comparison_operator",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `inbuffer != NULL && insize > 0` when `inbuffer` points to a valid buffer and `insize` is 1024?",
    "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nif (inbuffer != NULL && insize > 0) {\n    printf(\"Input is valid\");\n} else {\n    printf(\"Input validation failed\");\n}",
    "answer": true
},
{
    "id": "SL-BL-S001-V06",
    "metadata": {
        "name": "StatementLevel-Boolean-S001-Variant6",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S001",
        "variant_type": "ternary_operator",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `inbuffer == NULL || insize == 0` when `inbuffer` points to a valid buffer and `insize` is 1024?",
        "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nint result = (inbuffer == NULL || insize == 0) ? 1 : 0;\nprintf(\"Result: %d\\n\", result);",
        "answer": false
    }
},
{
    "id": "SL-BL-S001-V07",
    "metadata": {
        "name": "StatementLevel-Boolean-S001-Variant7",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S001",
        "variant_type": "nested_logic",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `(inbuffer == NULL || insize == 0) && validate` when `inbuffer` points to a valid buffer, `insize` is 1024, and `validate` is 1?",
        "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nint validate = 1;\nif ((inbuffer == NULL || insize == 0) && validate)\n{\n    printf(\"Input validation failed\\n\");\n} else {\n    printf(\"Input is valid\\n\");\n}",
        "answer": "false"
    }
},
{
    "id": "SL-BL-S001-V08",
    "metadata": {
        "name": "StatementLevel-Boolean-S001-Variant8",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S001",
        "variant_type": "language_port_python",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code snippet, what is the value of the boolean expression `inbuffer is None or insize == 0` when `inbuffer` is a valid object and `insize` is 1024?",
        "code": "inbuffer = b'\\x01\\x02\\x03' # valid bytes object\n\nif inbuffer is None or insize == 0:\n    print(\"Input validation failed\")\nelse:\n    print(\"Input is valid\")",
        "answer": "false"
    }
},
{
    "id": "SL-BL-S001-V09",
    "metadata": {
        "name": "StatementLevel-Boolean-S001-Variant9",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S001",
        "variant_type": "macro_definition",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet, what is the value of the boolean expression `inbuffer == INVALID_BUFFER || insize == INVALID_SIZE` when `inbuffer` points to a valid buffer and `insize` is 1024?",  
        "code": "#define INVALID_BUFFER NULL\n#define INVALID_SIZE 0\nconst unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nif (inbuffer == INVALID_BUFFER || insize == INVALID_SIZE) {\n    printf(\"Input validation failed\\n\");\n} else {\n    printf(\"Input is valid\\n\");\n}",  
        "answer": false  
    },  
},  
{  
    "id": "SL-BL-S001-v10",  
    "metadata": {  
        "name": "StatementLevel-Boolean-S001-Variant10",  
        "category": "Statement-Level",  
        "subcategory": "Boolean",  
        "type": "variant",  
        "source_seed": "SL-BL-S001",  
        "variant_type": "type_conversion",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet, what is the value of the boolean expression `!inbuffer || !insize` when `inbuffer` points to a valid buffer and `insize` is 1024?",  
        "code": "const unsigned char *inbuffer = (const unsigned char *)0x10687f9;\nunsigned long insize = 1024;\nif (!inbuffer || !insize) {\n    printf(\"Input validation failed\\n\");\n} else {\n    printf(\"Input is valid\\n\");\n}",  
        "answer": false  
    },  
},  
{  
    "id": "SL-BL-S002-v01",  
    "metadata": {  
        "name": "StatementLevel-Boolean-S002-Variant1",  
        "category": "Statement-Level",  
        "subcategory": "Boolean",  
        "type": "variant",  
        "source_seed": "SL-BL-S002",  
        "variant_type": "logical_negation",  
        "language": "c",  
        "difficulty": "easy",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet, what is the value of the boolean expression `!!data` when `data` is NULL?",  
        "code": "void *data = NULL;\nsize_t size = 1024;\nif (!!data) {\n    memset(data, 0, size);\n} else {\n    return;\n}",  
        "answer": false  
    },  
}
```

```
},
{
  "id": "SL-BL-S002-v02",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant2",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "comparison_operator",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `data == NULL` when `data` is NULL?",
    "code": "void *data = NULL;\nsize_t size = 1024;\nif (data == NULL) {\nreturn;\n} else {\n    memset(data, 0, size);\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S002-v03",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant3",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "special_value",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `!data` when `data` points to address 0x0?",
    "code": "void *data = (void *)0x0;\nsize_t size = 1024;\nif (!data) {\nreturn;\n} else {\n    memset(data, 0, size);\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S002-v04",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant4",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "pointer_comparison",
    "language": "c",
    "intervention": 0
  }
}
```



```
{
  "id": "SL-BL-S002-V06",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant6",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "language_port_java",
    "language": "java",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following Java code snippet, what is the value of the boolean expression `data == null` when `data` is null?",
    "code": "Object data = null;\nint size = 1024;\nif (data == null) {\n    return;\n} else {\n    System.out.println(\"Processing data\");\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S002-V07",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant7",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "macro_definition",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `IS_INVALID(data)` when `data` is NULL?",
    "code": "#define IS_INVALID(ptr) (!ptr)\nvoid *data = NULL;\nsize_t size = 1024;\nif (IS_INVALID(data)) {\n    return;\n} else {\n    memset(data, 0, size);\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S002-V08",
  "metadata": {
    "name": "StatementLevel-Boolean-S002-Variant8",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S002",
    "variant_type": "type_cast",
    "language": "c",
    "difficulty": "hard",
  }
}
```

```
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `!(char *)data` when `data` is NULL?",
        "code": "void *data = NULL;\nsize_t size = 1024;\nif (!(char *)data) {\nreturn;\n} else {\n    memset(data, 0, size);\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S002-v09",
    "metadata": {
        "name": "StatementLevel-Boolean-S002-variant9",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S002",
        "variant_type": "ternary_operator",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `!data` when `data` is NULL?",
        "code": "void *data = NULL;\nsize_t size = 1024;\nint should_return = !data ? 1 : 0;\nif (should_return) {\n    return;\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S002-v10",
    "metadata": {
        "name": "StatementLevel-Boolean-S002-variant10",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S002",
        "variant_type": "short_circuit_side_effect",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `!data || ++counter > 0` when `data` is NULL and `counter` is 0?",
        "code": "void *data = NULL;\nsize_t size = 1024;\nint counter = 0;\nif (!data ||\n++counter > 0) {\n    return;\n} else {\n    memset(data, 0, size);\n}",
        "answer": true
    }
},
{
```

```
"id": "SL-BL-S003-V01",
"metadata": {
    "name": "StatementLevel-Boolean-S003-Variant1",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S003",
    "variant_type": "comparison_operator",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `res == DW_DLV_OK` when `res` is -1 and `DW_DLV_OK` is 0?",
    "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif (res == DW_DLV_OK) {\n    printf(\"Operation succeeded\");\n} else {\n    printf(\"Operation failed\");\n}",
    "answer": false
},
{
    "id": "SL-BL-S003-V02",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant2",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "boundary_value",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res != DW_DLV_OK` when `res` is 0 and `DW_DLV_OK` is 0?",
        "code": "int res = 0;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif (res != DW_DLV_OK) {\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n}",
        "answer": false
    }
},
{
    "id": "SL-BL-S003-V03",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant3",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "compound_condition",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```

        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res != DW_DLV_OK && res != DW_DLV_NO_ENTRY` when `res` is -1, `DW_DLV_OK` is 0, and `DW_DLV_NO_ENTRY` is 1?",
        "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nconst int DW_DLV_NO_ENTRY = 1;\nif (res != DW_DLV_OK && res != DW_DLV_NO_ENTRY) {\n    printf(\"Operation failed with error\");\n    return DW_DLV_ERROR;\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S003-v04",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant4",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "enum_comparison",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res != DW_DLV_OK` when `res` is DW_DLV_ERROR and `DW_DLV_OK` is 0?",
        "code": "typedef enum {\n    DW_DLV_OK = 0,\n    DW_DLV_ERROR = -1,\n    DW_DLV_NO_ENTRY = 1\n} Dwarf_Error;\nDwarf_Error res = DW_DLV_ERROR;\nif (res != DW_DLV_OK)\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n",
        "answer": true
    }
},
{
    "id": "SL-BL-S003-v05",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant5",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "negative_comparison",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res < 0` when `res` is -1?",
        "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif (res < 0) {\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n}",
        "answer": true
    }
}

```

```

    },
},
{
  "id": "SL-BL-S003-V06",
  "metadata": {
    "name": "StatementLevel-Boolean-S003-Variant6",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S003",
    "variant_type": "macro_expansion",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `IS_ERROR(res)` when `res` is -1?",
    "code": "#define IS_ERROR(x) ((x) != 0)\nint res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif (IS_ERROR(res)) {\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S003-V07",
  "metadata": {
    "name": "StatementLevel-Boolean-S003-Variant7",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S003",
    "variant_type": "logical_negation",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `!(res == DW_DLV_OK)` when `res` is -1 and `DW_DLV_OK` is 0?",
    "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif (!(res == DW_DLV_OK)) {\n    printf(\"Operation failed\");\n    return DW_DLV_ERROR;\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S003-V08",
  "metadata": {
    "name": "StatementLevel-Boolean-S003-Variant8",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S003",
    "variant_type": "logical_negation",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  }
}

```

```
        "variant_type": "ternary_operator",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `res != DW_DLV_OK` when `res` is -1 and `DW_DLV_OK` is 0?",
        "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nint result = (res != DW_DLV_OK) ? DW_DLV_ERROR : DW_DLV_OK;\nprintf(\"Result: %d\\n\", result);",
        "answer": true
    }
},
{
    "id": "SL-BL-S003-V09",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant9",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "language_port_python",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code snippet, what is the value of the boolean expression `res != DW_DLV_OK` when `res` is -1 and `DW_DLV_OK` is 0?",
        "code": "res = -1\nDW_DLV_OK = 0\nDW_DLV_ERROR = -1\nif res != DW_DLV_OK:\n    print(\"Operation failed\")\n    return DW_DLV_ERROR",
        "answer": true
    }
},
{
    "id": "SL-BL-S003-V10",
    "metadata": {
        "name": "StatementLevel-Boolean-S003-Variant10",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S003",
        "variant_type": "type_conversion",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `(unsigned int)res != (unsigned int)DW_DLV_OK` when `res` is -1 and `DW_DLV_OK` is 0?",
        "code": "unsigned int res = -1;\nunsigned int DW_DLV_OK = 0;\nunsigned int DW_DLV_ERROR = -1;\nunsigned int result = (res != DW_DLV_OK) ? DW_DLV_ERROR : DW_DLV_OK;\nprintf(\"Result: %u\\n\", result);"
    }
}
```

```

        "code": "int res = -1;\nconst int DW_DLV_OK = 0;\nconst int DW_DLV_ERROR = -1;\nif\n((unsigned int)res != (unsigned int)DW_DLV_OK) {\n    printf(\"Operation failed\\n\");\n    return DW_DLV_ERROR;\n}\n",
        "answer": true
    },
},
{
    "id": "SL-BL-S004-v01",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant1",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "comparison_operator",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c >= 0x80` when `c` is 65 (ASCII 'A')?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (c >= 0x80) {\n    printf(\"Non-ASCII character\\n\");\n} else {\n    printf(\"ASCII character\\n\");\n}",
        "answer": false
    }
},
{
    "id": "SL-BL-S004-v02",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant2",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "boundary_value",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 128 (0x80)?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 128; // 0x80\nif (c < 0x80)\n    printf(\"ASCII character\\n\");\nelse\n    printf(\"Non-ASCII character\\n\");",
        "answer": false
    }
},
{
    "id": "SL-BL-S004-v03",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant3",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "boundary_value",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 128 (0x80)?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 128; // 0x80\nif (c < 0x80)\n    printf(\"ASCII character\\n\");\nelse\n    printf(\"Non-ASCII character\\n\");",
        "answer": false
    }
}

```

```
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "hex_comparison",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c < 0xFF` when `c` is 65 (ASCII 'A')?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (c < 0xFF) {\n    printf(\"Single byte character\");\n} else {\n    printf(\"Multi-byte character\");\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S004-V04",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant4",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "range_check",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c >= 0x20 && c < 0x80` when `c` is 65 (ASCII 'A')?",
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (c >= 0x20 && c < 0x80) {\n    printf(\"Printable ASCII character\");\n} else {\n    printf(\"Non-printable or non-ASCII character\");\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S004-V05",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant5",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "logical_negation",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
}
```

```

"task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `!(c >= 0x80)` when `c` is 65 (ASCII 'A')?", 
    "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (!(c >= 0x80)) {\n    printf(\"ASCII character\");\n} else {\n    printf(\"Non-ASCII character\");\n}",
    "answer": true
},
{
    "id": "SL-BL-S004-v06",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant6",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "macro_definition",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `IS_ASCII(c)` when `c` is 65 (ASCII 'A')?", 
        "code": "#define ASCII_MAX 0x80\n#define IS_ASCII(x) ((x) < ASCII_MAX)\n\ntypedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif (IS_ASCII(c)) {\n    printf(\"ASCII character\");\n} else {\n    printf(\"Non-ASCII character\");\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S004-v07",
    "metadata": {
        "name": "StatementLevel-Boolean-S004-Variant7",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S004",
        "variant_type": "type_conversion",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `(char)c < (char)0x80` when `c` is 65 (ASCII 'A')?", 
        "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nif ((char)c < (char)0x80) {\n    printf(\"ASCII character\");\n} else {\n    printf(\"Non-ASCII character\");\n}",
        "answer": true
    }
},

```

```
{
  "id": "SL-BL-S004-V08",
  "metadata": {
    "name": "StatementLevel-Boolean-S004-Variant8",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S004",
    "variant_type": "ternary_operator",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 65 (ASCII 'A')?",
    "code": "typedef unsigned int krb5_ucs4;\nkrb5_ucs4 c = 65; // ASCII 'A'\nconst char* type = (c < 0x80) ? \"ASCII\" : \"Non-ASCII\";\nprintf(\"%s character\\n\", type);",
    "answer": true
  }
},
{
  "id": "SL-BL-S004-V09",
  "metadata": {
    "name": "StatementLevel-Boolean-S004-Variant9",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S004",
    "variant_type": "language_port_javascript",
    "language": "javascript",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following javascript code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 65 (ASCII 'A')?",
    "code": "let c = 65; // ASCII 'A'\nif (c < 0x80) {\n  console.log(\"ASCII character\");\n} else {\n  console.log(\"Non-ASCII character\");\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S004-V10",
  "metadata": {
    "name": "StatementLevel-Boolean-S004-Variant10",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S004",
    "variant_type": "overflow_boundary",
    "language": "c",
    "difficulty": "hard",
  }
}
```

```
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `c < 0x80` when `c` is 255 (0xFF)?",
        "code": "typedef unsigned char krb5_ucs4;\nkrb5_ucs4 c = 255; // 0xFF\nif (c < 0x80) {\n    printf(\"ASCII character\");\n} else {\n    printf(\"Non-ASCII character\");\n}",
        "answer": false
    }
},
{
    "id": "SL-BL-S005-V01",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant1",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "logical_operator",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR && state == IN_CHAR_SPACE` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
        "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif (state == IN_CHAR && state == IN_CHAR_SPACE) {\n    return len;\n} else {\n    return 0;\n}",
        "answer": false
    }
},
{
    "id": "SL-BL-S005-V02",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant2",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "macro_expansion",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `state == 1 || state == 2` when `state` is 1?",
        "code": "int state = 1;\nint len = 5;\nif (state == 1 || state == 2) {\n    return len;\n} else {\n    return 0;\n}",
        "answer": true
    }
}
```

```

    },
},
{
  "id": "SL-BL-S005-V03",
  "metadata": {
    "name": "StatementLevel-Boolean-S005-Variant3",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S005",
    "variant_type": "complexity_increase",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR || state == IN_CHAR_SPACE || state == IN_WORD` when `state` is 1, `IN_CHAR` is 1, `IN_CHAR_SPACE` is 2, and `IN_WORD` is 3?",
    "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\n#define IN_WORD 3\nint state = 1;\nint len = 5;\nif (state == IN_CHAR || state == IN_CHAR_SPACE || state == IN_WORD) {\n    return len;\n} else {\n    return 0;\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S005-V04",
  "metadata": {
    "name": "StatementLevel-Boolean-S005-Variant4",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S005",
    "variant_type": "nested_logic",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `(state == IN_CHAR || state == IN_CHAR_SPACE) && len > 0` when `state` is 1, `IN_CHAR` is 1, `IN_CHAR_SPACE` is 2, and `len` is 5?",
    "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif ((state == IN_CHAR || state == IN_CHAR_SPACE) && len > 0) {\n    return len;\n} else {\n    return 0;\n}",
    "answer": true
  }
},
{
  "id": "SL-BL-S005-V05",
  "metadata": {
    "name": "StatementLevel-Boolean-S005-Variant5",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S005",
    "variant_type": "nested_logic",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `(state == IN_CHAR || state == IN_CHAR_SPACE) && len > 0` when `state` is 1, `IN_CHAR` is 1, `IN_CHAR_SPACE` is 2, and `len` is 5?",
    "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif ((state == IN_CHAR || state == IN_CHAR_SPACE) && len > 0) {\n    return len;\n} else {\n    return 0;\n}",
    "answer": true
  }
}

```

```
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "short_circuit",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR_SPACE || state == IN_CHAR` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
        "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif (state == IN_CHAR_SPACE || state == IN_CHAR) {\n    return len;\n} else {\n    return 0;\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S005-V06",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant6",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "enum_definition",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR || state == IN_CHAR_SPACE` when `state` is IN_CHAR?",
        "code": "typedef enum {\n    IN_CHAR = 1,\n    IN_CHAR_SPACE = 2,\n    OUT_WORD = 0\n} State;\nState state = IN_CHAR;\nint len = 5;\nif (state == IN_CHAR || state == IN_CHAR_SPACE) {\n    return len;\n} else {\n    return 0;\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S005-V07",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant7",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "ternary_operator",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
}
```

```

"task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `state == IN_CHAR || state == IN_CHAR_SPACE` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
    "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nint result = (state == IN_CHAR || state == IN_CHAR_SPACE) ? len : 0;\nreturn result;",
    "answer": true
},
{
    "id": "SL-BL-S005-V08",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant8",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "logical_negation",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `!(state != IN_CHAR && state != IN_CHAR_SPACE)` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
        "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\nif (!(state != IN_CHAR && state != IN_CHAR_SPACE)) {\n    return len;\n} else {\n    return 0;\n}",
        "answer": true
    }
},
{
    "id": "SL-BL-S005-V09",
    "metadata": {
        "name": "StatementLevel-Boolean-S005-Variant9",
        "category": "Statement-Level",
        "subcategory": "Boolean",
        "type": "variant",
        "source_seed": "SL-BL-S005",
        "variant_type": "language_port_python",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code snippet, what is the value of the boolean expression `state == IN_CHAR or state == IN_CHAR_SPACE` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2?",
        "code": "IN_CHAR = 1\nIN_CHAR_SPACE = 2\nstate = 1\nlength = 5\nif state == IN_CHAR or state == IN_CHAR_SPACE:\n    return length\nelse:\n    return 0",
        "answer": true
    }
}

```

```

},
{
  "id": "SL-BL-S005-v10",
  "metadata": {
    "name": "StatementLevel-Boolean-S005-Variant10",
    "category": "Statement-Level",
    "subcategory": "Boolean",
    "type": "variant",
    "source_seed": "SL-BL-S005",
    "variant_type": "chain_comparison",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `IN_CHAR <= state <= IN_CHAR_SPACE` when `state` is 1, `IN_CHAR` is 1, and `IN_CHAR_SPACE` is 2? (Note: This is not valid C syntax, but evaluates as `(IN_CHAR <= state) <= IN_CHAR_SPACE`)",
    "code": "#define IN_CHAR 1\n#define IN_CHAR_SPACE 2\nint state = 1;\nint len = 5;\n// This evaluates as: (IN_CHAR <= state) <= IN_CHAR_SPACE\n// which is: (1 <= 1) <= 2,\ni.e., 1 <= 2\nif (IN_CHAR <= state <= IN_CHAR_SPACE) {\n    return len;\n} else {\n    return 0;\n}",
    "answer": true
  }
}

```

1C - API/函数调用 [API/Function Call] (57)

任务目标

基于给定的API/函数调用种子任务，生成多样化的变式任务来全面测试大模型对函数调用行为、返回值、参数传递和API语义的理解能力。

变式生成维度

1. 函数类别变式

- **标准库函数变式**: `string.h`, `math.h`, `stdio.h`, `stdlib.h`等标准库函数
- **系统调用变式**: `unistd.h`, `sys/types.h`等系统级API
- **内存管理变式**: `malloc`, `calloc`, `realloc`, `free`等内存相关函数
- **文件操作变式**: `fopen`, `fread`, `fwrite`, `fclose`等文件I/O函数
- **字符串处理变式**: `strlen`, `strcmp`, `strcpy`, `strcat`等字符串函数
- **数学函数变式**: `pow`, `sqrt`, `sin`, `cos`, `log`等数学计算函数

2. 参数类型变式

- **基础类型参数**: `int`, `float`, `double`, `char`等基础数据类型
- **指针参数变式**: `char*`, `void*`, 函数指针等指针类型参数
- **结构体参数变式**: 传递结构体、结构体指针作为参数
- **数组参数变式**: 数组作为参数的传递和处理
- **可变参数变式**: `printf`, `scanf`等可变参数函数
- **常量参数变式**: `const`修饰的参数传递

3. 返回值类型变式

- **基础返回值变式**: 返回`int`, `float`, `double`等基础类型

- **指针返回值变式**：返回指针（包括NULL的情况）
- **特殊返回值变式**：返回**typedef**定义的类型（如**pid_t**, **size_t**）
- **错误码返回变式**：返回错误状态码或成功/失败标志
- **无返回值变式**：**void**函数的副作用分析
- **复合返回值变式**：返回结构体或联合体

4. 边界条件变式

- **空值输入变式**：NULL指针、空字符串、零长度参数
- **边界值变式**：最大值、最小值、临界值作为参数
- **缓冲区边界变式**：缓冲区大小限制相关的函数调用
- **内存不足变式**：**malloc**等函数在内存不足时的行为
- **文件不存在变式**：文件操作函数处理不存在文件的情况
- **权限错误变式**：权限不足时的函数行为

5. 错误处理变式

- **成功场景变式**：函数正常执行的返回值和行为
- **失败场景变式**：函数执行失败时的错误处理
- **异常输入变式**：非法参数、越界访问等异常情况
- **资源限制变式**：系统资源限制导致的函数失败
- **并发问题变式**：多线程环境下的函数行为

6. 副作用分析变式

- **内存修改变式**：函数对传入指针指向内存的修改
- **全局状态变式**：函数对全局变量或系统状态的影响
- **文件状态变式**：文件操作函数对文件状态的改变
- **缓冲区修改变式**：函数对缓冲区内容的修改和影响
- **输出副作用变式**：**printf**等输出函数的副作用

7. 精度和格式变式

- **数值精度变式**：浮点数计算的精度问题
- **格式字符串变式**：**printf**/**scanf**格式说明符的变化
- **类型转换变式**：参数类型转换对函数行为的影响
- **编码变式**：字符编码相关的函数行为
- **字节序变式**：大小端对函数行为的影响

8. 平台相关变式

- ****sizeof**变式**：不同平台下**sizeof**的不同结果
- **类型大小变式**：**int**, **long**, **pointer**等类型在不同平台的大小
- **路径分隔符变式**：文件路径在不同系统下的表示
- **换行符变式**：不同系统的换行符处理
- **API可用性变式**：某些API在特定平台的可用性

9. 语言移植变式

- **C -> Python变式**：对应Python内置函数或标准库
- **C -> Java变式**：对应Java标准库方法
- **C -> JavaScript变式**：对应JavaScript内置函数
- **跨语言语义变式**：相同功能在不同语言中的行为差异

10. 复杂度变式

- **简化变式**：单一函数调用，直接参数
- **复合变式**：嵌套函数调用，函数作为参数
- **链式调用变式**：函数返回值作为另一个函数的参数
- **条件调用变式**：基于条件的函数调用选择

生成规则

基本要求

1. 每个种子任务生成8-12个变式
2. 确保每个变式维度至少覆盖2-3个变式
3. 保持API调用的核心语义，但改变具体场景
4. 变式难度应该涵盖easy/medium/hard三个等级

难度分级标准

- **Easy**: 标准库基础函数，常见参数，正常返回值
- **Medium**: 复杂参数类型，错误处理，边界条件
- **Hard**: 嵌套调用，特殊场景，平台相关行为，复杂副作用

特殊考虑点

1. **API文档准确性**: 确保函数行为符合标准库文档
2. **平台一致性**: 明确指定假设的平台环境（如`sizeof`值）
3. **错误处理**: 准确模拟各种错误情况的函数行为
4. **内存安全**: 正确处理指针和内存相关的函数调用

输出格式

为每个变式生成完整的JSON格式，包含：

- 唯一ID（SL-API-S00X-VYY格式）
- 完整的metadata（标注变式类型和源种子）
- 清晰的task描述和代码
- 准确的答案（数值、字符串、类型名或特殊值如NULL）

质量检查要点

1. **API正确性**: 确保函数调用语法和语义正确
2. **返回值准确性**: 确保返回值计算和类型正确
3. **边界情况**: 充分测试各种边界和异常情况
4. **实用性**: 每个变式应该测试实际编程中的常见场景

示例变式类型

基于SL-API-S001 (`strlen`) 的变式示例：

- **边界值变式**: 空字符串""的`strlen`返回值
- **复合调用变式**: `strlen(strcpy(dest, src))`的嵌套调用
- **内存安全变式**: `strlen(NULL)`的未定义行为
- **编码变式**: 包含多字节字符的字符串长度
- **语言移植变式**: Python的`len()`函数对应行为

基于SL-API-S003 (`malloc`) 的变式示例：

- **失败场景变式**: `malloc(SIZE_MAX)`返回NULL
- **零大小变式**: `malloc(0)`的实现定义行为
- **对齐变式**: `malloc`返回内存的对齐要求
- **calloc比较变式**: `calloc` vs `malloc`的区别
- **内存泄漏变式**: 忘记`free`的后果分析

基于SL-API-S007 (`snprintf`) 的变式示例：

- **缓冲区溢出变式**: 格式化字符串超过缓冲区大小
- **格式说明符变式**: 不同的%格式符的行为
- **返回值语义变式**: 返回值的精确含义

- ****NULL缓冲区变式****: 缓冲区为NULL时的行为
- ****精度控制变式****: %.2f等精度控制的效果

请基于此提示词, 为给定的API/函数调用种子任务生成完整的变式集合, 确保全面覆盖函数调用推理的各个方面。

```
{
  "id": "SL-API-S001-V01",
  "metadata": {
    "name": "StatementLevel-APICall-Seed1-Variant1",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S001",
    "variant_type": "边界条件变式",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the return value of the function call `strlen(str)` when `str` is an empty string \"\"?",
    "code": "#include <string.h>\nchar *str = \"\";\nsize_t len =\nstrlen(str);\nprintf(\"Length: %zu\\n\", len);",
    "answer": 0
  }
},
{
  "id": "SL-API-S001-V02",
  "metadata": {
    "name": "StatementLevel-APICall-Seed1-Variant2",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S001",
    "variant_type": "复合调用变式",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the return value of the nested function call `strlen(strcpy(dest, src))` when `src` is \"world\"?",
    "code": "#include <string.h>\nchar src[] = \"world\";\nchar dest[10];\nsize_t len =\nstrlen(strcpy(dest, src));\nprintf(\"Length: %zu\\n\", len);",
    "answer": 5
  }
},
{
  "id": "SL-API-S001-V03",
  "metadata": {
    "name": "StatementLevel-APICall-Seed1-Variant3",
    "category": "Statement-Level",
    "subcategory": "API Call"
  }
}
```

```
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S001",
        "variant_type": "错误处理变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what happens when `strlen(NULL)` is called?",
        "code": "#include <string.h>\nchar *str = NULL;\n// strlen(str); // This would cause undefined behavior\nprintf(\"This is undefined behavior\\n\");",
        "answer": "undefined behavior"
    }
},
{
    "id": "SL-API-S001-v04",
    "metadata": {
        "name": "StatementLevel-APICall-Seed1-Variant4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S001",
        "variant_type": "字符串处理变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `strlen(str)` when `str` contains escape characters \"hello\\nworld\"?",
        "code": "#include <string.h>\nchar *str = \"hello\\nworld\";\nsize_t len =\nstrlen(str);\nprintf(\"Length: %zu\\n\", len);",
        "answer": 11
    }
},
{
    "id": "SL-API-S001-v05",
    "metadata": {
        "name": "StatementLevel-APICall-Seed1-Variant5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S001",
        "variant_type": "数组参数变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with a character array, what is the return value of `strlen(arr)`?",
        "code": "char arr[10] = {0};\nprintf(\"Length: %zu\\n\", strlen(arr));"
    }
}
```

```
"code": "#include <string.h>\nchar arr[] = {'h', 'e', 'l', 'l', 'o', '\\0'};\nsize_t\nlen = strlen(arr);\\nprintf(\"Length: %zu\\n\", len);",
"answer": 5
},
{
"id": "SL-API-S001-V06",
"metadata": {
  "name": "StatementLevel-APICall-Seed1-variant6",
  "category": "Statement-Level",
  "subcategory": "API Call",
  "type": "variant",
  "source_seed": "SL-API-S001",
  "variant_type": "常量参数变式",
  "language": "c",
  "difficulty": "easy",
  "intervention": 0
},
"task": {
  "description": "Given the following code snippet with a const string, what is the return value of `strlen(str)`?",
  "code": "#include <string.h>\nconst char *str = \\\"programming\\\";\nsize_t len =\nstrlen(str);\\nprintf(\"Length: %zu\\n\", len);",
  "answer": 11
},
{
"id": "SL-API-S001-V07",
"metadata": {
  "name": "StatementLevel-APICall-Seed1-variant7",
  "category": "Statement-Level",
  "subcategory": "API Call",
  "type": "variant",
  "source_seed": "SL-API-S001",
  "variant_type": "返回值类型变式",
  "language": "c",
  "difficulty": "medium",
  "intervention": 0
},
"task": {
  "description": "Given the following code snippet, what is the data type returned by `strlen(str)`?",
  "code": "#include <string.h>\nchar *str = \\\"test\\\";\nsize_t len = strlen(str); // What\n// type is len?\nprintf(\"Length: %zu\\n\", len);",
  "answer": "size_t"
},
{
"id": "SL-API-S001-V08",
"metadata": {
  "name": "StatementLevel-APICall-Seed1-variant8",
  "category": "Statement-Level",
  "subcategory": "API Call",
```

```
"type": "variant",
"source_seed": "SL-API-S001",
"variant_type": "语言移植变式",
"language": "python",
"difficulty": "easy",
"intervention": 0
},
"task": {
  "description": "Given the following Python code equivalent to the C strlen function, what is the return value of `len(s)` when `s` is \"hello\"?",
  "code": "s = \"hello\"\nlength = len(s)\nprint(f\"Length: {length}\")",
  "answer": 5
},
{
  "id": "SL-API-S002-V01",
  "metadata": {
    "name": "StatementLevel-APICall-Seed2-variant1",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S002",
    "variant_type": "边界值变式",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what is the return value of `pow(2.0, 0)` when the exponent is 0?",
    "code": "#include <math.h>\ndouble base = 2.0;\nint exponent = 0;\ndouble result =\npow(base, exponent);\nprintf(\"Result: %.1f\\n\", result);",
    "answer": 1.0
  },
  {
    "id": "SL-API-S002-V02",
    "metadata": {
      "name": "StatementLevel-APICall-Seed2-variant2",
      "category": "Statement-Level",
      "subcategory": "API Call",
      "type": "variant",
      "source_seed": "SL-API-S002",
      "variant_type": "负数指数变式",
      "language": "c",
      "difficulty": "medium",
      "intervention": 0
    },
    "task": {
      "description": "Given the following code snippet, what is the return value of `pow(2.0, -2)` when the exponent is negative?",
      "code": "#include <math.h>\ndouble base = 2.0;\nint exponent = -2;\ndouble result =\npow(base, exponent);\nprintf(\"Result: %.2f\\n\", result);",
      "answer": 0.25
    }
  }
}
```

```
        "answer": 0.25
    },
},
{
    "id": "SL-API-S002-V03",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "特殊值变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `pow(0.0, 0.0)`?",
        "code": "#include <math.h>\ndouble base = 0.0;\ndouble exponent = 0.0;\ndouble result = pow(base, exponent);\nprintf(\"Result: %.1f\\n\", result);",
        "answer": 1.0
    }
},
{
    "id": "SL-API-S002-V04",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "浮点精度变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `pow(1.5, 3.0)` with floating point precision?",
        "code": "#include <math.h>\ndouble base = 1.5;\ndouble exponent = 3.0;\ndouble result = pow(base, exponent);\nprintf(\"Result: %.3f\\n\", result);",
        "answer": 3.375
    }
},
{
    "id": "SL-API-S002-V05",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "浮点精度变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```
        "variant_type": "错误处理变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `pow(-1.0, 0.5)` when taking square root of negative number?",
        "code": "#include <math.h>\n#include <math.h>\ndouble base = -1.0;\ndouble exponent = 0.5;\ndouble result = pow(base, exponent); // This returns NaN (Not a Number)\nprintf(\"Result: %f\\n\", result);",
        "answer": "NaN"
    }
},
{
    "id": "SL-API-S002-V06",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant6",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "链式调用变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of the nested `pow(pow(2.0, 2), 2)` call?",
        "code": "#include <math.h>\ndouble result = pow(pow(2.0, 2), 2);printf(\"Result: %.0f\\n\", result);",
        "answer": 16.0
    }
},
{
    "id": "SL-API-S002-V07",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant7",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "类型转换变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with integer arguments, what is the return value of `pow(3, 4)`?",
        "code": "#include <math.h>\nint base = 3;\nint exponent = 4;\ndouble result = pow(base, exponent); // integers promoted to double\nprintf(\"Result: %.0f\\n\", result);",
        "answer": 81.0
    }
}
```

```
        "answer": 81.0
    },
},
{
    "id": "SL-API-S002-V08",
    "metadata": {
        "name": "StatementLevel-APICall-Seed2-Variant8",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S002",
        "variant_type": "语言移植变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code equivalent to the C pow function, what is the return value of `pow(2.0, 3)` or `2.0 ** 3`?",
        "code": "base = 2.0\nexponent = 3\nresult = pow(base, exponent)\n# or result = base\n** exponent\nprint(f\"Result: {result}\")",
        "answer": 8.0
    }
},
{
    "id": "SL-API-S003-V01",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3-Variant1",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S003",
        "variant_type": "零大小变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the behavior of `malloc(0)` when requesting zero bytes?",
        "code": "#include <stdlib.h>\nvoid *ptr = malloc(0);\nif (ptr == NULL) {\n    printf(\"malloc(0) returned NULL\\n\");\n} else {\n    printf(\"malloc(0) returned non-\nNULL\\n\");\n    free(ptr);\n}",
        "answer": "implementation-defined"
    }
},
{
    "id": "SL-API-S003-V02",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3-Variant2",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
```

```
        "source_seed": "SL-API-S003",
        "variant_type": "失败场景变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what does `malloc(SIZE_MAX)` return when requesting maximum possible size?",
        "code": "#include <stdlib.h>\n#include <stdint.h>\nvoid *ptr = malloc(SIZE_MAX);\nif (ptr == NULL) {\n    printf(\"Memory allocation failed\\n\");\n} else {\n    printf(\"Memory allocation succeeded\\n\");\n    free(ptr);\n}",
        "answer": "NULL"
    }
},
{
    "id": "SL-API-S003-V03",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3-Variant3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S003",
        "variant_type": "calloc比较变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, how many bytes does `calloc(10, sizeof(int))` allocate, assuming sizeof(int) is 4?",
        "code": "#include <stdlib.h>\n// Assume sizeof(int) = 4 bytes\nint *arr = (int *)calloc(10, sizeof(int));\nif (arr != NULL) {\n    printf(\"Memory allocated and initialized to zero\\n\");\n    free(arr);\n}",
        "answer": 40
    }
},
{
    "id": "SL-API-S003-V04",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3-Variant4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S003",
        "variant_type": "对齐变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the minimum alignment guarantee for the pointer returned by `malloc(1)`?",
        "code": "#include <stdlib.h>\nint main() {\n    void *ptr = malloc(1);\n    if (ptr != NULL) {\n        printf(\"The minimum alignment guarantee is %d bytes\\n\",\n            (int)(&((char *)ptr)[1] - &ptr));\n    }\n    free(ptr);\n    return 0;\n}"
    }
}
```

```
"code": "#include <stdlib.h>\nvoid *ptr = malloc(1);\nif (ptr != NULL) {\n    printf(\"Pointer alignment: suitable for any object\\n\");\n    free(ptr);\n}\n",
"answer": "max_align_t"
},
{
"id": "SL-API-S003-V05",
"metadata": {
    "name": "StatementLevel-APICall-Seed3-variant5",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S003",
    "variant_type": "realloc变式",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet, how many bytes does `realloc(ptr, 80)` allocate when expanding from 40 bytes?",
    "code": "#include <stdlib.h>\nint *ptr = (int *)malloc(sizeof(int) * 10); // 40\nbytes\nptr = (int *)realloc(ptr, sizeof(int) * 20); // Expand to 20 ints\nif (ptr != NULL)\n    printf(\"Memory reallocated\\n\");\n    free(ptr);\n",
    "answer": 80
},
},
{
"id": "SL-API-S003-V06",
"metadata": {
    "name": "StatementLevel-APICall-Seed3-Variant6",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S003",
    "variant_type": "平台相关变式",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet on a 64-bit platform, how many bytes does `malloc(sizeof(long) * 5)` allocate assuming sizeof(long) is 8?",
    "code": "#include <stdlib.h>\n// Assume 64-bit platform: sizeof(long) = 8\nbytes\nlong *arr = (long *)malloc(sizeof(long) * 5);\nif (arr != NULL)\n    printf(\"Memory allocated for 5 longs\\n\");\n    free(arr);\n",
    "answer": 40
},
},
{
"id": "SL-API-S003-V07",
"metadata": {
    "name": "StatementLevel-APICall-Seed3-variant7",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S003",
    "variant_type": "platform-specific",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet on a 64-bit platform, how many bytes does `malloc(sizeof(long) * 5)` allocate assuming sizeof(long) is 8?",
    "code": "#include <stdlib.h>\n// Assume 64-bit platform: sizeof(long) = 8\nbytes\nlong *arr = (long *)malloc(sizeof(long) * 5);\nif (arr != NULL)\n    printf(\"Memory allocated for 5 longs\\n\");\n    free(arr);\n",
    "answer": 40
}
}
```

```
"category": "Statement-Level",
"subcategory": "API Call",
"type": "variant",
"source_seed": "SL-API-S003",
"variant_type": "结构体参数变式",
"language": "c",
"difficulty": "medium",
"intervention": 0
},
"task": {
    "description": "Given the following code snippet with a struct, how many bytes does `malloc(sizeof(struct Point) * 3)` allocate?",
    "code": "#include <stdlib.h>\nstruct Point {\n    int x; // 4 bytes\n    int y; // 4 bytes\n}; // Total: 8 bytes per struct\nstruct Point *points = (struct Point *)malloc(sizeof(struct Point) * 3);\nif (points != NULL) {\n    printf(\"Memory allocated for 3 points\\n\");\n    free(points);\n}",
    "answer": 24
}
},
{
    "id": "SL-API-S003-V08",
    "metadata": {
        "name": "StatementLevel-APICall-Seed3-Variant8",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S003",
        "variant_type": "返回值类型变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return type of the `malloc()` function?",
        "code": "#include <stdlib.h>\nvoid *ptr = malloc(100);\nif (ptr != NULL) {\n    printf(\"malloc returns void pointer\\n\");\n    free(ptr);\n}",
        "answer": "void *"
    }
},
{
    "id": "SL-API-S004-V01",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant1",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "相关系统调用变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return type of the `getpid()` function?"
    }
}
```

```
"task": {
    "description": "Given the following code snippet, what type of value does the
function call `getppid()` return?",
    "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t parent_pid =
getppid();\nprintf(\"Parent Process ID: %d\\n\", parent_pid);",
    "answer": "pid_t"
},
{
    "id": "SL-API-S004-V02",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant2",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "用户ID变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what type of value does the
function call `getuid()` return?",
        "code": "#include <unistd.h>\n#include <sys/types.h>\nuid_t user_id =
getuid();\nprintf(\"User ID: %d\\n\", user_id);",
        "answer": "uid_t"
    }
},
{
    "id": "SL-API-S004-V03",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "组ID变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what type of value does the
function call `getgid()` return?",
        "code": "#include <unistd.h>\n#include <sys/types.h>\ngid_t group_id =
getgid();\nprintf(\"Group ID: %d\\n\", group_id);",
        "answer": "gid_t"
    }
},
{
    "id": "SL-API-S004-V04",
    "metadata": {
```

```
        "name": "StatementLevel-APICall-Seed4-Variant4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "错误处理变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, can the function call `getpid()` ever fail and return an error?",
        "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t current_pid = getpid();\n// getpid() never fails\nprintf(\"Process ID: %d\\n\", current_pid);",
        "answer": "no"
    }
},
{
    "id": "SL-API-S004-V05",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "fork变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what type of value does the function call `fork()` return to identify the child process?",
        "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t child_pid = fork();\nif (child_pid == 0) {\n    printf(\"This is the child process\\n\");\n} else if (child_pid > 0)\n{\n    printf(\"This is the parent, child PID: %d\\n\", child_pid);\n}",
        "answer": "pid_t"
    }
},
{
    "id": "SL-API-S004-V06",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant6",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "参数传递变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
}
```

```
"task": {
    "description": "Given the following code snippet, how many parameters does the function call `getpid()` take?",
    "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t current_pid = getpid();\n// No parameters\nprintf(\"Process ID: %d\\n\", current_pid);",
    "answer": 0
},
{
    "id": "SL-API-S004-V07",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant7",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "平台相关变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the typical size of `pid_t` on most modern Unix systems?",
        "code": "#include <unistd.h>\n#include <sys/types.h>\npid_t current_pid = getpid();\nprintf(\"PID size: %zu bytes\\n\", sizeof(pid_t));",
        "answer": 4
    }
},
{
    "id": "SL-API-S004-V08",
    "metadata": {
        "name": "StatementLevel-APICall-Seed4-Variant8",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S004",
        "variant_type": "语言移植变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code equivalent to the c getpid function, what type does `os.getpid()` return?",
        "code": "import os\ncurrent_pid = os.getpid()\nprint(f\"Process ID: {current_pid}\")",
        "answer": "int"
    }
},
{
    "id": "SL-API-S005-V01",
    "metadata": {
```

```
        "name": "StatementLevel-APICall-Seed5-Variant1",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S005",
        "variant_type": "相等字符串变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `strcmp(str1, str2)` when both strings are identical?",
        "code": "#include <string.h>\nchar *str1 = \"hello\";\nchar *str2 = \"hello\";\nint result = strcmp(str1, str2);\\nprintf(\"Comparison result: %d\\n\", result);",
        "answer": 0
    }
},
{
    "id": "SL-API-S005-V02",
    "metadata": {
        "name": "StatementLevel-APICall-Seed5-Variant2",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S005",
        "variant_type": "大小写敏感变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `strcmp(str1, str2)` when comparing \"Hello\" and \"hello\"?",
        "code": "#include <string.h>\nchar *str1 = \"Hello\";\nchar *str2 = \"hello\";\nint result = strcmp(str1, str2);\\nprintf(\"Comparison result: %d\\n\", result);",
        "answer": -1
    }
},
{
    "id": "SL-API-S005-V03",
    "metadata": {
        "name": "StatementLevel-APICall-Seed5-Variant3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S005",
        "variant_type": "空字符串变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet, what is the return value of\n`strcmp(str1, str2)` when `str1` is empty and `str2` is \"test\"?",\n        "code": "#include <string.h>\nchar *str1 = \"/\";\nchar *str2 = \"test\";\nint result\n= strcmp(str1, str2);\\nprintf(\"Comparison result: %d\\n\", result);",\n        "answer": -1\n    }\n},\n{\n    "id": "SL-API-S005-v04",\n    "metadata": {\n        "name": "StatementLevel-APICall-Seed5-Variant4",\n        "category": "Statement-Level",\n        "subcategory": "API Call",\n        "type": "variant",\n        "source_seed": "SL-API-S005",\n        "variant_type": "strcmp变式",\n        "language": "c",\n        "difficulty": "medium",\n        "intervention": 0\n    },\n    "task": {\n        "description": "Given the following code snippet, what is the return value of\n`strncmp(str1, str2, 3)` when comparing first 3 characters?",\n        "code": "#include <string.h>\nchar *str1 = \"application\";\nchar *str2 =\n\"apple\";\nint result = strncmp(str1, str2, 3);\\nprintf(\"Comparison result: %d\\n\", result);",\n        "answer": 0\n    }\n},\n{\n    "id": "SL-API-S005-v05",\n    "metadata": {\n        "name": "StatementLevel-APICall-Seed5-Variant5",\n        "category": "Statement-Level",\n        "subcategory": "API Call",\n        "type": "variant",\n        "source_seed": "SL-API-S005",\n        "variant_type": "strcasecmp变式",\n        "language": "c",\n        "difficulty": "medium",\n        "intervention": 0\n    },\n    "task": {\n        "description": "Given the following code snippet, what is the return value of\n`strcasecmp(str1, str2)` when doing case-insensitive comparison?",\n        "code": "#include <string.h>\\n#include <strings.h>\nchar *str1 = \"Hello\";\nchar *str2 = \"HELLO\";\nint result = strcasecmp(str1, str2);\\nprintf(\"Case-insensitive\ncomparison: %d\\n\", result);",\n        "answer": 0\n    }\n},\n{\n    "id": "SL-API-S005-v06",\n
```

```
"metadata": {
    "name": "StatementLevel-APICall-Seed5-Variant6",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S005",
    "variant_type": "错误处理变式",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet, what happens when `strcmp(NULL, \"test\")` is called?",
    "code": "#include <string.h>\nchar *str1 = NULL;\nchar *str2 = \"test\";\n//\nstrcmp(str1, str2); // This would cause undefined behavior\nprintf(\"Comparing NULL pointer is undefined behavior\\n\");",
    "answer": "undefined behavior"
}
},
{
    "id": "SL-API-S005-V07",
    "metadata": {
        "name": "StatementLevel-APICall-Seed5-Variant7",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S005",
        "variant_type": "数值字符串变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
},
    "task": {
        "description": "Given the following code snippet, what is the return value of `strcmp(str1, str2)` when comparing \"10\" and \"2\"?",
        "code": "#include <string.h>\nchar *str1 = \"10\";\nchar *str2 = \"2\";\nint result = strcmp(str1, str2); // Lexicographic comparison, not numeric\nprintf(\"Comparison result: %d\\n\", result);",
        "answer": -1
}
},
{
    "id": "SL-API-S005-V08",
    "metadata": {
        "name": "StatementLevel-APICall-Seed5-Variant8",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S005",
        "variant_type": "语言移植变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
}
}
```

```
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code equivalent to C strcmp function, what does the comparison `str1 < str2` return when `str1` is \"apple\" and `str2` is \"banana\"?",
        "code": "str1 = \"apple\"\nstr2 = \"banana\"\nresult = str1 < str2 # Returns boolean, not integer\nprint(f\"Comparison result: {result}\")",
        "answer": "True"
    }
},
{
    "id": "SL-API-S006-v01",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6-Variant1",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S006",
        "variant_type": "成功场景变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what type of value does `fopen(filename, \"w\")` return when successfully creating a new file?",
        "code": "#include <stdio.h>\nchar *filename = \"output.txt\";\nFILE *fp = fopen(filename, \"w\");\nif (fp != NULL) {\n    printf(\"File opened successfully for writing\\n\");\n    fclose(fp);\n}",
        "answer": "FILE *"
    }
},
{
    "id": "SL-API-S006-v02",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6-Variant2",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S006",
        "variant_type": "权限错误变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what does `fopen(\"/root/file.txt\", \"w\")` return when a regular user lacks write permission to /root?",
        "code": "#include <stdio.h>\nchar *filename = \"/root/file.txt\"; // No write permission for regular user\nFILE *fp = fopen(filename, \"w\");\nif (fp == NULL) {\n    printf(\"Permission denied\\n\");\n} else {\n    fclose(fp);\n}",
        "answer": "Permission denied"
    }
}
```

```
        "answer": "NULL"
    }
},
{
    "id": "SL-API-S006-V03",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6-Variant3",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S006",
        "variant_type": "不同模式变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what does `fopen(filename, \"a\")` do when the file already exists?",
        "code": "#include <stdio.h>\nchar *filename = \"existing.txt\";\nFILE *fp =\nfopen(filename, \"a\"); // Append mode\nif (fp != NULL) {\n    printf(\"File opened in\nappend mode\\n\");\n    fclose(fp);\n}\n",
        "answer": "opens for appending"
    }
},
{
    "id": "SL-API-S006-V04",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6-Variant4",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S006",
        "variant_type": "二进制模式变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the difference between `fopen(filename, \"r\")` and `fopen(filename, \"rb\")`?",
        "code": "#include <stdio.h>\nchar *filename = \"data.bin\";\nFILE *fp1 =\nfopen(filename, \"r\"); // Text mode\nFILE *fp2 = fopen(filename, \"rb\"); // Binary\nmode\nif (fp1) fclose(fp1);\nif (fp2) fclose(fp2);",
        "answer": "binary mode vs text mode"
    }
},
{
    "id": "SL-API-S006-V05",
    "metadata": {
        "name": "StatementLevel-APICall-Seed6-Variant5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S006",
        "variant_type": "二进制模式变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```
"type": "variant",
"source_seed": "SL-API-S006",
"variant_type": "路径分隔符变式",
"language": "c",
"difficulty": "medium",
"intervention": 0
},
"task": {
  "description": "Given the following code snippet on windows, what should the path separator be in the filename for `fopen()`?",
  "code": "#include <stdio.h>\n// On Windows\nchar *filename =\n\"C:\\\\\\Users\\\\\\file.txt\"; // Correct windows path\nFILE *fp = fopen(filename,\n\"r\");\nif (fp != NULL) {\n    printf(\"File opened on windows\\n\");\n    fclose(fp);\n}"
  "answer": "\\\\"
},
{
  "id": "SL-API-S006-v06",
  "metadata": {
    "name": "StatementLevel-APICall-Seed6-Variant6",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S006",
    "variant_type": "freopen变式",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what does `freopen(filename, \"w\", stdout)` do to the stdout stream?",
    "code": "#include <stdio.h>\nchar *filename = \"output.txt\";\nFILE *fp =\nfreopen(filename, \"w\", stdout);\nprintf(\"This goes to the file\\n\");\nif (fp != NULL)\n{\n    // stdout is now redirected to file\n}"
    "answer": "redirects stdout to file"
  }
},
{
  "id": "SL-API-S006-v07",
  "metadata": {
    "name": "StatementLevel-APICall-Seed6-Variant7",
    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S006",
    "variant_type": "tmpfile变式",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
```

```
        "description": "Given the following code snippet, what type of file does `tmpfile()` create?",  
        "code": "#include <stdio.h>\nFILE *temp_fp = tmpfile();\nif (temp_fp != NULL) {\nprintf(\"Temporary file created\\n\");\n    // File is automatically deleted when closed\nfclose(temp_fp);\n}",  
        "answer": "temporary file"  
    },  
,  
{  
    "id": "SL-API-S006-V08",  
    "metadata": {  
        "name": "StatementLevel-APICall-Seed6-Variant8",  
        "category": "Statement-Level",  
        "subcategory": "API Call",  
        "type": "variant",  
        "source_seed": "SL-API-S006",  
        "variant_type": "语言移植变式",  
        "language": "python",  
        "difficulty": "easy",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following Python code equivalent to C fopen function, what does `open(filename, 'r')` return when the file doesn't exist?",  
        "code": "filename = \"nonexistent.txt\"\ntry:\n    fp = open(filename, 'r')\n    print(\"File opened successfully\")\n    fp.close()\nexcept FileNotFoundError:\n    print(\"File not found\")",  
        "answer": "FileNotFoundException"  
    },  
,  
{  
    "id": "SL-API-S007-V01",  
    "metadata": {  
        "name": "StatementLevel-APICall-Seed7-Variant1",  
        "category": "Statement-Level",  
        "subcategory": "API Call",  
        "type": "variant",  
        "source_seed": "SL-API-S007",  
        "variant_type": "缓冲区溢出变式",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet, what is the return value of `snprintf(buffer, size, format, value)` when the formatted string exceeds buffer size?",  
        "code": "#include <stdio.h>\nchar buffer[5];\nint size = sizeof(buffer);\nconst char *format = \"%d\";\nint value = 123456789; // This will exceed buffer size\nint result =\n    snprintf(buffer, size, format, value);\nprintf(\"Buffer: %.4s, Result: %d\\n\", buffer, result);",  
        "answer": 9  
    },  
,
```

```
{  
  "id": "SL-API-S007-V02",  
  "metadata": {  
    "name": "StatementLevel-APICall-Seed7-Variant2",  
    "category": "Statement-Level",  
    "subcategory": "API Call",  
    "type": "variant",  
    "source_seed": "SL-API-S007",  
    "variant_type": "格式说明符变式",  
    "language": "c",  
    "difficulty": "medium",  
    "intervention": 0  
  },  
  "task": {  
    "description": "Given the following code snippet, what is the return value of `snprintf()` when formatting a float with precision?",  
    "code": "#include <stdio.h>\nchar buffer[20];\nint size = sizeof(buffer);\nconst char *format = \"%.\n2f\";\ndouble value = 3.14159;\nint result = snprintf(buffer, size, format, value);\nprintf(\"Buffer: %s, Result: %d\\n\", buffer, result);",  
    "answer": 4  
  }  
,  
{  
  "id": "SL-API-S007-V03",  
  "metadata": {  
    "name": "StatementLevel-APICall-Seed7-Variant3",  
    "category": "Statement-Level",  
    "subcategory": "API Call",  
    "type": "variant",  
    "source_seed": "SL-API-S007",  
    "variant_type": "NULL缓冲区变式",  
    "language": "c",  
    "difficulty": "hard",  
    "intervention": 0  
  },  
  "task": {  
    "description": "Given the following code snippet, what is the return value of `snprintf(NULL, 0, format, value)` when buffer is NULL and size is 0?",  
    "code": "#include <stdio.h>\nconst char *format = \"%s %d\";\nchar *str = \"Hello\";\nint num = 42;\nint result = snprintf(NULL, 0, format, str, num);\n// This returns the number of characters that would be written\nprintf(\"Required buffer size: %d\\n\", result);",  
    "answer": 8  
  }  
,  
{  
  "id": "SL-API-S007-V04",  
  "metadata": {  
    "name": "StatementLevel-APICall-Seed7-Variant4",  
    "category": "Statement-Level",  
    "subcategory": "API Call",  
    "type": "variant",  
    "source_seed": "SL-API-S007",  
  }  
}
```

```
        "variant_type": "sprintf比较变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the key safety difference between `sprintf()` and `snprintf()`?",
        "code": "#include <stdio.h>\nchar buffer[10];\n// sprintf(buffer, \"%d\", 123456789); // Unsafe - no bounds checking\nint result = snprintf(buffer, sizeof(buffer), \"%d\", 123456789); // Safe\nprintf(\"snprintf is bounds-checked\\n\");",
        "answer": "bounds checking"
    }
},
{
    "id": "SL-API-S007-v05",
    "metadata": {
        "name": "StatementLevel-APICall-Seed7-Variant5",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S007",
        "variant_type": "多参数变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `snprintf()` with multiple format specifiers?",
        "code": "#include <stdio.h>\nchar buffer[30];\nint size = sizeof(buffer);\nconst char *format = \"%s: %d years old\";\nchar *name = \"Alice\";\nint age = 25;\nint result = snprintf(buffer, size, format, name, age);\nprintf(\"Buffer: %s, Result: %d\\n\", buffer, result);",
        "answer": 17
    }
},
{
    "id": "SL-API-S007-v06",
    "metadata": {
        "name": "StatementLevel-APICall-Seed7-Variant6",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S007",
        "variant_type": "十六进制格式变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what is the return value of `snprintf()` when formatting an integer as hexadecimal?",
        "code": "#include <stdio.h>\nchar buffer[10];\nint size = sizeof(buffer);\nconst char *format = \"%x\";\nint result = snprintf(buffer, size, format, 123456789);"
    }
}
```

```
        "code": "#include <stdio.h>\nchar buffer[20];\nint size = sizeof(buffer);\nconst\nchar *format = \\\"0x%\\\";\nint value = 255;\nint result = sprintf(buffer, size, format,\nvalue);\nprintf(\"Buffer: %s, Result: %d\\n\", buffer, result);",
        "answer": 4
    },
},
{
    "id": "SL-API-S007-V07",
    "metadata": {
        "name": "StatementLevel-APICall-Seed7-Variant7",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S007",
        "variant_type": "零大小缓冲区变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet, what happens when `sprintf()` is\ncalled with size = 1 (only space for null terminator)?",
        "code": "#include <stdio.h>\nchar buffer[1];\nint size = 1; // Only space for null\nterminator\nconst char *format = \\\"%d\\\";\nint value = 42;\nint result = sprintf(buffer,\nsize, format, value);\nprintf(\"Buffer contains: '%s', Result: %d\\n\", buffer, result);",
        "answer": 2
    }
},
{
    "id": "SL-API-S007-V08",
    "metadata": {
        "name": "StatementLevel-APICall-Seed7-variant8",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S007",
        "variant_type": "语言移植变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code equivalent to C sprintf function,\nwhat does string formatting with f-strings return?",
        "code": "value = 12345\nformatted = f\"{value}\"\nprint(f\"Formatted: {formatted},\nLength: {len(formatted)}\")",
        "answer": "string object"
    }
},
{
    "id": "SL-API-S007-V09",
    "metadata": {
        "name": "StatementLevel-APICall-Seed7-variant9",
        "category": "Statement-Level",
        "subcategory": "API Call",
        "type": "variant",
        "source_seed": "SL-API-S007",
        "variant_type": "字符串格式化变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following Python code equivalent to C sprintf function,\nwhat does string formatting with f-strings return?",
        "code": "value = 12345\nformatted = f\"{value}\"\nprint(f\"Formatted: {formatted},\nLength: {len(formatted)}\")",
        "answer": "string object"
    }
}
```

```

    "category": "Statement-Level",
    "subcategory": "API Call",
    "type": "variant",
    "source_seed": "SL-API-S007",
    "variant_type": "vsnprintf变式",
    "language": "c",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet, what additional parameter does `vsnprintf()` require compared to `snprintf()`?",
    "code": "#include <stdio.h>\n#include <stdarg.h>\nint my_printf(char *buffer, size_t size, const char *format, ...){\n    va_list args;\n    va_start(args, format);\n    int result = vsnprintf(buffer, size, format, args);\n    va_end(args);\n    return result;\n}",
    "answer": "va_list"
  }
}

```

1D - 变量赋值 [Assignment] (80)

变量赋值变式生成提示词

任务目标

基于给定的变量赋值种子任务，生成多样化的变式任务来全面测试大模型对赋值操作语义、变量状态变化、内存引用和数据流的理解能力。

变式生成维度

1. 赋值类型变式

- **简单赋值变式**: 基础的 `x = value` 形式赋值
- **复合赋值变式**: `+=, -=, *=, /=, %=, |=, &=, ^=` 等复合运算符
- **链式赋值变式**: `x = y = z = value` 的连续赋值
- **多重赋值变式**: `a, b = value1, value2` 的并行赋值
- **解包赋值变式**: 序列解包、字典解包、星号表达式
- **条件赋值变式**: 三元运算符或条件表达式的赋值

2. 数据类型变式

- **基础类型变式**: `int, float, string, boolean` 的赋值
- **容器类型变式**: `list, tuple, dict, set` 的赋值和解包
- **引用类型变式**: 对象引用、别名效应的理解
- **可变/不可变变式**: 可变对象vs不可变对象的赋值行为差异
- **指针赋值变式**: C语言中指针的赋值和解引用
- **类型转换变式**: 隐式/显式类型转换的赋值

3. 内存和引用变式

- **值拷贝变式**: 值类型的深拷贝行为
- **引用拷贝变式**: 引用类型的浅拷贝行为
- **别名效应变式**: 多个变量指向同一对象的情况
- **指针运算变式**: 指针赋值、地址运算、解引用
- **内存共享变式**: 多个变量共享内存时的修改影响
- **作用域变式**: 不同作用域中的变量赋值和可见性

4. 解包和展开变式

- **列表解包变式**：`[a, b, c] = [1, 2, 3]` 的各种形式
- **元组解包变式**：嵌套元组的解包赋值
- **星号表达式变式**：`*args, **kwargs` 的收集和展开
- **忽略变量变式**：使用 `_` 忽略不需要的值
- **不匹配长度变式**：解包时左右两边数量不匹配的情况
- **嵌套解包变式**：多层嵌套结构的解包

5. 复合运算变式

- **算术复合变式**：`+=, -=, *=, /=` 等数学运算符
- **位运算复合变式**：`|=, &=, ^=, <<=, >>=` 等位运算符
- **字符串复合变式**：`+=` 对字符串的连接操作
- **列表复合变式**：`+=` 对列表的扩展操作
- **原地修改变式**：原地修改 **vs** 创建新对象的区别
- **运算优先级变式**：复合赋值中的运算优先级

6. 错误和边界变式

- **类型不匹配变式**：不兼容类型间的赋值尝试
- **解包错误变式**：解包数量不匹配的错误情况
- **只读变量变式**：尝试修改常量或只读变量
- **未定义变量变式**：使用未定义变量进行赋值
- **空值赋值变式**：`None, NULL, undefined` 的赋值
- **循环引用变式**：自引用或循环引用的赋值

7. 作用域和生命周期变式

- **局部变量变式**：函数内部的局部变量赋值
- **全局变量变式**：全局变量的修改和访问
- **闭包变量变式**：闭包中的变量捕获和修改
- **静态变量变式**：静态变量的赋值和持久性
- **临时变量变式**：临时对象的赋值和生命周期
- **变量遮蔽变式**：同名变量在不同作用域的遮蔽

8. 语言特性变式

- **Python特性变式**：多重赋值、解包、`walrus` 运算符
- **C语言特性变式**：指针赋值、结构体赋值、数组赋值
- **Java特性变式**：对象引用赋值、原始类型赋值
- **JavaScript特性变式**：`var/let/const`、解构赋值
- **跨语言对比变式**：相同操作在不同语言中的行为差异

9. 性能和优化变式

- **内存效率变式**：大对象的赋值和拷贝成本
- **引用计数变式**：引用计数变化对内存的影响
- **写时拷贝变式**：`copy-on-write` 机制的赋值优化
- **原地操作变式**：原地修改 **vs** 重新赋值的性能差异
- **缓存效应变式**：变量赋值对缓存的影响

10. 复杂场景变式

- **嵌套赋值变式**：多层嵌套的复杂赋值操作
- **条件赋值变式**：基于条件的选择性赋值
- **循环赋值变式**：循环中的变量赋值和累积
- **递归赋值变式**：递归函数中的变量赋值
- **并发赋值变式**：多线程环境下的变量赋值安全性

生成规则

基本要求

1. 每个种子任务生成10-15个变式
2. 确保每个变式维度至少覆盖2-3个变式
3. 保持赋值操作的核心语义，但改变具体场景
4. 变式难度应该涵盖easy/medium/hard三个等级

难度分级标准

- **Easy**: 简单赋值，基础数据类型，直接值赋值
- **Medium**: 复合赋值，容器解包，引用类型，类型转换
- **Hard**: 复杂解包，嵌套赋值，内存引用，错误处理，跨语言特性

特殊考虑点

1. **语义准确性**: 确保赋值操作的语义正确
2. **内存模型**: 正确理解值传递和引用传递
3. **类型系统**: 考虑强类型和弱类型语言的差异
4. **执行顺序**: 理解赋值操作的执行顺序和副作用

输出格式

为每个变式生成完整的JSON格式，包含：

- 唯一ID (SL-BL-S00X-VYY格式)
- 完整的metadata (标注变式类型和源种子)
- 清晰的task描述和代码
- 准确且唯一的答案

请基于此提示词，为给定的种子任务生成完整的变式集合，确保全面覆盖布尔逻辑推理的各个方面。

```
{  
  "id": "SL-AS-S001-V01",  
  "metadata": {  
    "name": "StatementLevel-Assignment-Seed1-variant1",  
    "category": "Statement-Level",  
    "subcategory": "Assignment",  
    "type": "variant",  
    "source": "Generated",  
    "source_seed": "SL-AS-S001",  
    "variant_type": "数据类型变式",  
    "language": "python",  
    "difficulty": "easy",  
    "intervention": 0  
  },  
  "task": {  
    "description": "Given the following code snippet with string assignment, what is the value of variable x after executing the assignment statement `x = y`?",  
    "code": "y = \"hello\"\nz = \"world\"\nx = y\nprint(f\"x = {x}\")",  
    "answer": "hello"  
  },  
  {  
    "id": "SL-AS-S001-V02",  
    "metadata": {
```

```
        "name": "StatementLevel-Assignment-Seed1-Variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S001",
        "variant_type": "数据类型变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with boolean assignment, what is the value of variable x after executing the assignment statement `x = y`?",
        "code": "y = True\nz = False\nx = y\nprint(f\"x = {x}\")",
        "answer": true
    }
},
{
    "id": "SL-AS-S001-V03",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed1-Variant3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S001",
        "variant_type": "数据类型变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with float assignment, what is the value of variable x after executing the assignment statement `x = y`?",
        "code": "y = 3.14\nz = 2.71\nx = y\nprint(f\"x = {x}\")",
        "answer": 3.14
    }
},
{
    "id": "SL-AS-S001-V04",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed1-Variant4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S001",
        "variant_type": "容器类型变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
}
```

```
"task": {
    "description": "Given the following code snippet with list assignment, what is the length of list x after executing the assignment statement `x = y`?",
    "code": "y = [1, 2, 3, 4]\nz = [5, 6]\nx = y\nprint(f\"x = {x}, length = {len(x)}\")",
    "answer": 4
},
{
    "id": "SL-AS-S001-v05",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed1-Variant5",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S001",
        "variant_type": "引用类型变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with list reference assignment, what is the value of x[0] after modifying y[0]?",
        "code": "y = [1, 2, 3]\nz = [4, 5, 6]\nx = y\ny[0] = 999\nprint(f\"x = {x}, y = {y}\")",
        "answer": 999
    }
},
{
    "id": "SL-AS-S001-v06",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed1-Variant6",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S001",
        "variant_type": "类型转换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with type conversion assignment, what is the type of variable x after executing the assignment statement `x = int(y)`?",
        "code": "y = \"42\"\nz = \"100\"\nx = int(y)\nprint(f\"x = {x}, type = {type(x).__name__}\")",
        "answer": "int"
    }
},
{
```

```
"id": "SL-AS-S001-v07",
"metadata": {
    "name": "StatementLevel-Assignment-Seed1-variant7",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S001",
    "variant_type": "空值赋值变式",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet with None assignment, what is the value of variable x after executing the assignment statement `x = y`?",
    "code": "y = None\nz = 100\nx = y\nprint(f\"x = {x}\")",
    "answer": null
},
{
"id": "SL-AS-S001-v08",
"metadata": {
    "name": "StatementLevel-Assignment-Seed1-variant8",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S001",
    "variant_type": "条件赋值变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet with conditional assignment, what is the value of variable x after executing the assignment statement `x = y if condition else z`?",
    "code": "y = 42\nz = 100\ncondition = True\nx = y if condition else z\nprint(f\"x = {x}\")",
    "answer": 42
},
{
"id": "SL-AS-S001-v09",
"metadata": {
    "name": "StatementLevel-Assignment-Seed1-variant9",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S001",
    "variant_type": "嵌套赋值变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
}
}
```

```
"language": "python",
"difficulty": "hard",
"intervention": 0
},
"task": {
  "description": "Given the following code snippet with nested assignment, what is the value of variable x after executing the assignment statement `x = data['key']`?",
  "code": "data = {'key': 42, 'other': 100}\ny = data['key']\nx = y\nprint(f\"x = {x}\")",
  "answer": 42
}
},
{
  "id": "SL-AS-S001-v10",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed1-Variant10",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S001",
    "variant_type": "多变量赋值变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with multiple variable assignment, what is the value of variable x after executing all assignment statements?",
    "code": "y = 42\nz = 100\nw = 200\nx = y\nx = z\nprint(f\"x = {x}\")",
    "answer": 100
  }
},
{
  "id": "SL-AS-S002-v01",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed2-Variant1",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S002",
    "variant_type": "数据类型变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with string swap, what is the value of variable a after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = \"hello\"\nb = \"world\"\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\nprint(f\"After swap: a={a}, b={b}\")",
    "answer": "world"
  }
}
```

```
        },
    },
    {
        "id": "SL-AS-S002-v02",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed2-variant2",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S002",
            "variant_type": "三元交换变式",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with three-variable swap, what is the value of variable a after executing the multiple assignment statement `a, b, c = b, c, a`?",
            "code": "a = 1\nb = 2\nc = 3\nprint(f\"Before swap: a={a}, b={b}, c={c}\")\n\na, b, c = b, c, a\nprint(f\"After swap: a={a}, b={b}, c={c}\")",
            "answer": 2
        }
    },
    {
        "id": "SL-AS-S002-v03",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed2-variant3",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S002",
            "variant_type": "列表交换变式",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with list swap, what is the first element of list a after executing the multiple assignment statement `a, b = b, a`?",
            "code": "a = [1, 2, 3]\nb = [4, 5, 6]\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\nprint(f\"After swap: a={a}, b={b}\")",
            "answer": 4
        }
    },
    {
        "id": "SL-AS-S002-v04",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed2-variant4",
            "category": "Statement-Level",
            "subcategory": "Assignment",
```

```
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S002",
        "variant_type": "表达式交换变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with expression swap, what is the value of variable a after executing the multiple assignment statement `a, b = b + 1, a * 2`?",
        "code": "a = 5\nb = 3\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b + 1, a * 2\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 4
    }
},
{
    "id": "SL-AS-S002-V05",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed2-Variant5",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S002",
        "variant_type": "部分交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with partial assignment, what is the value of variable a after executing the multiple assignment statement `a, _ = b, a`?",
        "code": "a = 10\nb = 20\nprint(f\"Before swap: a={a}, b={b}\")\n\na, _ = b,\nprint(f\"After partial swap: a={a}, b={b}\")",
        "answer": 20
    }
},
{
    "id": "SL-AS-S002-V06",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed2-Variant6",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S002",
        "variant_type": "嵌套解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
}
```

```
"task": {
    "description": "Given the following code snippet with nested unpacking, what is the value of variable x after executing the multiple assignment statement `(x, y), (z, w) = (z, w), (x, y)`?",
    "code": "x, y = 1, 2\nz, w = 3, 4\nprint(f\"Before swap: x={x}, y={y}, z={z}, w={w}\")\n(x, y), (z, w) = (z, w), (x, y)\nprint(f\"After swap: x={x}, y={y}, z={z}, w={w}\")",
    "answer": 3
},
{
    "id": "SL-AS-S002-v07",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed2-variant7",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S002",
        "variant_type": "同值交换变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with same value swap, what is the value of variable a after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 42\nb = 42\nprint(f\"Before swap: a={a}, b={b}\")\n(a, b) = (b, a)\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 42
    },
    {
        "id": "SL-AS-S002-v08",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed2-variant8",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S002",
            "variant_type": "元组交换变式",
            "language": "python",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with tuple swap, what is the second element of tuple a after executing the multiple assignment statement `a, b = b, a`?",
            "code": "a = (1, 2)\nb = (3, 4)\nprint(f\"Before swap: a={a}, b={b}\")\n(a, b) = (b, a)\nprint(f\"After swap: a={a}, b={b}\")",
            "answer": 4
        }
    }
}
```

```
},
{
  "id": "SL-AS-S002-v09",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed2-Variant9",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S002",
    "variant_type": "链式交换变式",
    "language": "python",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with chained swap, what is the value of variable a after executing both swap operations?",
    "code": "a = 1\nb = 2\nnc = 3\nprint(f\"Initial: a={a}, b={b}, c={c}\")\n\na, b = b,\na\nb, c = c, b\nprint(f\"After swaps: a={a}, b={b}, c={c}\")",
    "answer": 2
  }
},
{
  "id": "SL-AS-S002-v10",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed2-Variant10",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S002",
    "variant_type": "类型混合交换变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with mixed type swap, what is the type of variable a after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = 42\nb = \"hello\"\nprint(f\"Before swap: a={a} ({type(a).__name__}), b={b} ({type(b).__name__})\")\n\na, b = b, a\nprint(f\"After swap: a={a} ({type(a).__name__}), b={b} ({type(b).__name__})\")",
    "answer": "str"
  }
},
{
  "id": "SL-AS-S003-v01",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed3-Variant1",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
```

```
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "字符串解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with string unpacking, what is the value of variable first after executing the unpacking assignment `first, second, *rest = text`?",
        "code": "text = \"hello\"\nfirst, second, *rest = text\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": "h"
    }
},
{
    "id": "SL-AS-S003-v02",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "元组解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with tuple unpacking, what is the value of variable first after executing the unpacking assignment `first, second, *rest = data`?",
        "code": "data = (10, 20, 30, 40, 50)\nfirst, second, *rest = data\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 10
    }
},
{
    "id": "SL-AS-S003-v03",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "末尾解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
}
```

```
"task": {
    "description": "Given the following code snippet with tail unpacking, what is the value of variable last after executing the unpacking assignment `*front, second_last, last = numbers`?",
    "code": "numbers = [1, 2, 3, 4, 5]\n*front, second_last, last =\nnumbers\nprint(f\"front={front}, second_last={second_last}, last={last}\")",
    "answer": 5
},
{
    "id": "SL-AS-S003-v04",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "中间解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with middle unpacking, what is the length of list middle after executing the unpacking assignment `first, *middle, last = numbers`?",
        "code": "numbers = [1, 2, 3, 4, 5]\nfirst, *middle, last = numbers\nprint(f\"first={first}, middle={middle}, last={last}\")",
        "answer": 3
},
{
    "id": "SL-AS-S003-v05",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant5",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "嵌套解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with nested unpacking, what is the value of variable first after executing the unpacking assignment `(first, second), *rest = data`?",
        "code": "data = [(1, 2), (3, 4), (5, 6)]\n(first, second), *rest =\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 1
}
```

```
        },
    },
    {
        "id": "SL-AS-S003-v06",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed3-Variant6",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S003",
            "variant_type": "空序列解包变式",
            "language": "python",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with empty sequence unpacking, what is the length of list rest after executing the unpacking assignment `first, second, *rest = numbers`?",
            "code": "numbers = [1, 2]\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={rest}\")",
            "answer": 0
        }
    },
    {
        "id": "SL-AS-S003-v07",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed3-Variant7",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S003",
            "variant_type": "忽略变量变式",
            "language": "python",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with ignored variables, what is the value of variable first after executing the unpacking assignment `first, _, *rest = numbers`?",
            "code": "numbers = [1, 2, 3, 4, 5]\nfirst, _, *rest = numbers\nprint(f\"first={first}, rest={rest}\")",
            "answer": 1
        }
    },
    {
        "id": "SL-AS-S003-v08",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed3-Variant8",
            "category": "Statement-Level",
```

```
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "字典解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with dictionary unpacking, what is the value of variable first after executing the unpacking assignment `first, second, *rest = data.values()`?",
        "code": "data = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}\nfirst, second, *rest = data.values()\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 1
    }
},
{
    "id": "SL-AS-S003-V09",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant9",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "生成器解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with generator unpacking, what is the value of variable first after executing the unpacking assignment `first, second, *rest = gen`?",
        "code": "gen = (x * 2 for x in range(1, 6))\nfirst, second, *rest = gen\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 2
    }
},
{
    "id": "SL-AS-S003-V10",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed3-Variant10",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S003",
        "variant_type": "范围解包变式",
        "language": "python",
        "difficulty": "medium",
    }
}
```

```
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with range unpacking, what is the value of variable first after executing the unpacking assignment `first, second, *rest = range(1, 6)`?",  

        "code": "numbers = range(1, 6)\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={list(rest)}\")",
        "answer": 1
    }
},
{
    "id": "SL-AS-S004-v01",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed4-Variant1",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S004",
        "variant_type": "多级指针变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with double pointer, what is the value that **ptr2 points to after executing the assignment statements?",  

        "code": "int value = 100;\nint *ptr1 = &value;\nint **ptr2 = &ptr1;\nprintf(\"value: %d\\n\", **ptr2);",
        "answer": 100
    }
},
{
    "id": "SL-AS-S004-v02",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed4-Variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S004",
        "variant_type": "数组指针变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with array pointer, what is the value that ptr[1] points to after executing the assignment statement `ptr = arr`?",  

        "code": "int arr[] = {10, 20, 30, 40};\nint *ptr;\nptr = arr;\nprintf(\"ptr[1] = %d\\n\", ptr[1]);",
        "answer": 20
    }
}
```

```
        },
    },
    {
        "id": "SL-AS-S004-v03",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed4-variant3",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S004",
            "variant_type": "指针算术变式",
            "language": "c",
            "difficulty": "hard",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with pointer arithmetic, what is the value that *(ptr + 2) points to after executing the assignment statements?",
            "code": "int arr[] = {100, 200, 300, 400};\nint *ptr = arr;\nprintf(\"*(ptr + 2) = %d\\n\", *(ptr + 2));",
            "answer": 300
        }
    },
    {
        "id": "SL-AS-S004-v04",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed4-variant4",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S004",
            "variant_type": "结构体指针变式",
            "language": "c",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with struct pointer, what is the value of ptr->x after executing the assignment statement `ptr = &point`?",
            "code": "struct Point { int x, y; };\nstruct Point point = {100, 200};\nstruct Point *ptr;\nptr = &point;\nprintf(\"ptr->x = %d\\n\", ptr->x);",
            "answer": 100
        }
    },
    {
        "id": "SL-AS-S004-v05",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed4-variant5",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
```

```
        "source": "Generated",
        "source_seed": "SL-AS-S004",
        "variant_type": "空指针变式",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with NULL pointer, what is the value of ptr after executing the assignment statement `ptr = NULL`?",
        "code": "int value = 100;\nint *ptr = &value;\nptr = NULL;\nprintf(\"ptr = %p\\n\", ptr);",
        "answer": "null
    }
},
{
    "id": "SL-AS-S004-V06",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed4-Variant6",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S004",
        "variant_type": "函数指针变式",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with function pointer, what is the return value when calling the function through ptr after executing the assignment statement `ptr = &add`?",
        "code": "int add(int a, int b) { return a + b; }\nint (*ptr)(int, int);\nptr = &add;\nint result = ptr(30, 70);\nprintf(\"Result: %d\\n\", result);",
        "answer": "100
    }
},
{
    "id": "SL-AS-S004-V07",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed4-Variant7",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S004",
        "variant_type": "字符串指针变式",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet with string pointer, what is the first character that ptr points to after executing the assignment statement `ptr = str`?",  
        "code": "char str[] = \\\"Hello\\\";\\nchar *ptr;\\nptr = str;\\nprintf(\"First char: %c\\n\", *ptr);",  
        "answer": "H"  
    },  
},  
{  
    "id": "SL-AS-S004-v08",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed4-Variant8",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S004",  
        "variant_type": "指针交换变式",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with pointer swap, what is the value that ptr1 points to after the swap?",  
        "code": "int a = 100, b = 200;\\nint *ptr1 = &a, *ptr2 = &b;\\nint *temp = ptr1;\\nptr1 = ptr2;\\nptr2 = temp;\\nprintf(\"*ptr1 = %d\\n\", *ptr1);",  
        "answer": 200  
    },  
},  
{  
    "id": "SL-AS-S004-v09",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed4-Variant9",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S004",  
        "variant_type": "动态内存变式",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with dynamic memory, what is the value that ptr points to after executing the assignment and memory allocation?",  
        "code": "int *ptr;\\nptr = (int*)malloc(sizeof(int));\\n*ptr = 100;\\nprintf(\"*ptr = %d\\n\", *ptr);",  
        "answer": 100  
    },  
},  
{  
    "id": "SL-AS-S004-v10",  
}
```

```
"metadata": {
    "name": "StatementLevel-Assignment-Seed4-Variant10",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S004",
    "variant_type": "常量指针变式",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0
},
"task": {
    "description": "Given the following code snippet with constant pointer, what is the value that ptr points to after executing the assignment statement `ptr = &value`?",
    "code": "const int value = 100;\nconst int *ptr;\nptr = &value;\nprintf(\"*ptr = %d\\n\", *ptr);",
    "answer": 100
},
{
    "id": "SL-AS-S005-V01",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed5-Variant1",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S005",
        "variant_type": "四变量链式赋值变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
},
"task": {
    "description": "Given the following code snippet with four-variable chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = w = 10`?",
    "code": "x = 1\ny = 2\nz = 3\nw = 4\nprint(f\"Before: x={x}, y={y}, z={z}, w={w}\")\n\nx = y = z = w = 10\nprint(f\"After: x={x}, y={y}, z={z}, w={w}\")",
    "answer": 10
},
{
    "id": "SL-AS-S005-V02",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed5-Variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S005",
        "variant_type": "表达式链式赋值变式",
    }
}
```

```
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with expression chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = 2 * 3 + 1`?",
        "code": "x = 1\ny = 2\nz = 3\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = 2 * 3 + 1\nprint(f\"After: x={x}, y={y}, z={z}\")",
        "answer": 7
    }
},
{
    "id": "SL-AS-S005-v03",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed5-Variant3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S005",
        "variant_type": "列表链式赋值变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with list chained assignment, what is the first element of list x after executing the chained assignment `x = y = z = [1, 2, 3]`?",
        "code": "x = []\ny = []\nz = []\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = [1, 2, 3]\nprint(f\"After: x={x}, y={y}, z={z}\")",
        "answer": 1
    }
},
{
    "id": "SL-AS-S005-v04",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed5-Variant4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S005",
        "variant_type": "字符串链式赋值变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet with string chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = \"hello\"`?",  
        "code": "x = \"a\"\ny = \"b\"\nz = \"c\"\nprint(f\"Before: x={x}, y={y}, z={z}\")\nx = y = z = \"hello\"\nprint(f\"After: x={x}, y={y}, z={z}\")",  
        "answer": "hello"  
    },  
,  
{  
    "id": "SL-AS-S005-v05",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5-variant5",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S005",  
        "variant_type": "函数调用链式赋值变式",  
        "language": "python",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with function call chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = max(3, 7, 2)`?",  
        "code": "x = 0\ny = 0\nz = 0\nprint(f\"Before: x={x}, y={y}, z={z}\")\nx = y = z = max(3, 7, 2)\nprint(f\"After: x={x}, y={y}, z={z}\")",  
        "answer": 7  
    },  
,  
{  
    "id": "SL-AS-S005-v06",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5-variant6",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S005",  
        "variant_type": "嵌套链式赋值变式",  
        "language": "python",  
        "difficulty": "hard",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with nested chained assignment, what is the value of variable a after executing both chained assignments?",  
        "code": "a = b = 1\na = c = d = 2\nprint(f\"Before: a={a}, b={b}, c={c}, d={d}\")\na = b = c = d = 5\nprint(f\"After: a={a}, b={b}, c={c}, d={d}\")",  
        "answer": 5  
    },  
,
```

```
{  
    "id": "SL-AS-S005-V07",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5-Variant7",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S005",  
        "variant_type": "布尔链式赋值变式",  
        "language": "python",  
        "difficulty": "easy",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with boolean chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = True`?",  
        "code": "x = False\ny = False\nz = False\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = True\nprint(f\"After: x={x}, y={y}, z={z}\")",  
        "answer": true  
    }  
},  
{  
    "id": "SL-AS-S005-V08",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5-Variant8",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S005",  
        "variant_type": "字典链式赋值变式",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 0  
    },  
    "task": {  
        "description": "Given the following code snippet with dictionary chained assignment, what is the value of x['key'] after executing the chained assignment `x = y = z = {'key': 42}`?",  
        "code": "x = {}\ny = {}\nz = {}\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = {'key': 42}\nprint(f\"After: x={x}, y={y}, z={z}\")",  
        "answer": 42  
    }  
},  
{  
    "id": "SL-AS-S005-V09",  
    "metadata": {  
        "name": "StatementLevel-Assignment-Seed5-Variant9",  
        "category": "Statement-Level",  
        "subcategory": "Assignment",  
        "type": "variant",  
        "source": "Generated",  
        "source_seed": "SL-AS-S005",  
        "variant_type": "字典链式赋值变式",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 0  
    }  
}
```

```
"source_seed": "SL-AS-S005",
"variant_type": "None链式赋值变式",
"language": "python",
"difficulty": "easy",
"intervention": 0
},
"task": {
    "description": "Given the following code snippet with None chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = None`?",
    "code": "x = 1\ny = 2\nz = 3\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = None\nprint(f\"After: x={x}, y={y}, z={z}\")",
    "answer": null
}
},
{
    "id": "SL-AS-S005-v10",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed5-Variant10",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S005",
        "variant_type": "变量引用链式赋值变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with variable reference chained assignment, what is the value of variable x after executing the chained assignment `x = y = z = existing_var`?",
        "code": "existing_var = 100\nx = 1\ny = 2\nz = 3\nprint(f\"Before: x={x}, y={y}, z={z}\")\n\nx = y = z = existing_var\nprint(f\"After: x={x}, y={y}, z={z}\")",
        "answer": 100
    }
},
{
    "id": "SL-AS-S006-v01",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed6-Variant1",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S006",
        "variant_type": "减法复合赋值变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
```

```
        "description": "Given the following code snippet with subtraction compound assignment, what is the value of variable count after executing the compound assignment statement `count -= increment * 2`?",  
        "code": "count = 20\nincrement = 3\nmultiplier = 2\nprint(f\"Initial count: {count}\")\nccount -= increment * 2\nprint(f\"Final count: {ccount}\")",  
        "answer": 14  
    },  
    {  
        "id": "SL-AS-S006-V02",  
        "metadata": {  
            "name": "StatementLevel-Assignment-Seed6-variant2",  
            "category": "Statement-Level",  
            "subcategory": "Assignment",  
            "type": "variant",  
            "source": "Generated",  
            "source_seed": "SL-AS-S006",  
            "variant_type": "乘法复合赋值变式",  
            "language": "python",  
            "difficulty": "medium",  
            "intervention": 0  
        },  
        "task": {  
            "description": "Given the following code snippet with multiplication compound assignment, what is the value of variable count after executing the compound assignment statement `count *= increment + 1`?",  
            "code": "count = 5\nincrement = 2\nprint(f\"Initial count: {count}\")\nccount *= increment + 1\nprint(f\"Final count: {ccount}\")",  
            "answer": 15  
        },  
    },  
    {  
        "id": "SL-AS-S006-V03",  
        "metadata": {  
            "name": "StatementLevel-Assignment-Seed6-variant3",  
            "category": "Statement-Level",  
            "subcategory": "Assignment",  
            "type": "variant",  
            "source": "Generated",  
            "source_seed": "SL-AS-S006",  
            "variant_type": "除法复合赋值变式",  
            "language": "python",  
            "difficulty": "medium",  
            "intervention": 0  
        },  
        "task": {  
            "description": "Given the following code snippet with division compound assignment, what is the value of variable count after executing the compound assignment statement `count // divisor`?",  
            "code": "count = 20\n divisor = 3\nprint(f\"Initial count: {count}\")\nccount // divisor\nprint(f\"Final count: {ccount}\")",  
            "answer": 6  
        },  
    }
```

```
},
{
  "id": "SL-AS-S006-v04",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed6-Variant4",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S006",
    "variant_type": "模运算复合赋值变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with modulo compound assignment, what is the value of variable count after executing the compound assignment statement `count %= modulus`?",
    "code": "count = 17\nmodulus = 5\nprint(f\"Initial count: {count}\")\ncount %= modulus\nprint(f\"Final count: {count}\")",
    "answer": 2
  }
},
{
  "id": "SL-AS-S006-v05",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed6-Variant5",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S006",
    "variant_type": "幂运算复合赋值变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with power compound assignment, what is the value of variable count after executing the compound assignment statement `count **= exponent`?",
    "code": "count = 3\nexponent = 2\nprint(f\"Initial count: {count}\")\ncount **= exponent\nprint(f\"Final count: {count}\")",
    "answer": 9
  }
},
{
  "id": "SL-AS-S006-v06",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed6-Variant6",
    "category": "Statement-Level",
    "subcategory": "Assignment",
```

```
"type": "variant",
"source": "Generated",
"source_seed": "SL-AS-S006",
"variant_type": "字符串复合赋值变式",
"language": "python",
"difficulty": "medium",
"intervention": 0
},
"task": {
    "description": "Given the following code snippet with string compound assignment, what is the length of variable text after executing the compound assignment statement `text += suffix * 2`?",
    "code": "text = \"hello\"\nsuffix = \"!\"\nprint(f\"Initial text: '{text}'\")\n\ntext += suffix * 2\nprint(f\"Final text: '{text}'\")",
    "answer": 7
}
},
{
    "id": "SL-AS-S006-V07",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed6-Variant7",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S006",
        "variant_type": "列表复合赋值变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with list compound assignment, what is the length of list items after executing the compound assignment statement `items += new_items`?",
        "code": "items = [1, 2, 3]\nnew_items = [4, 5]\nprint(f\"Initial items: {items}\")\n\ncount += new_items\nprint(f\"Final items: {items}\")",
        "answer": 5
}
},
{
    "id": "SL-AS-S006-V08",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed6-Variant8",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S006",
        "variant_type": "位运算复合赋值变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    }
}
```

```
},
"task": {
    "description": "Given the following code snippet with bitwise OR compound assignment, what is the value of variable flags after executing the compound assignment statement `flags |= new_flag`?",
    "code": "flags = 5 # 101 in binary\nnew_flag = 2 # 010 in binary\nprint(f\"Initial flags: {flags} ({bin(flags)})\")\nflags |= new_flag\nprint(f\"Final flags: {flags} ({bin(flags)})\")",
    "answer": 7
},
{
},
{
    "id": "SL-AS-S006-V09",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed6-Variant9",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S006",
        "variant_type": "位与复合赋值变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with bitwise AND compound assignment, what is the value of variable mask after executing the compound assignment statement `mask &= filter_value`?",
        "code": "mask = 15 # 1111 in binary\nfilter_value = 10 # 1010 in binary\nprint(f\"Initial mask: {mask} ({bin(mask)})\")\nmask &= filter_value\nprint(f\"Final mask: {mask} ({bin(mask)})\")",
        "answer": 10
    }
},
{
    "id": "SL-AS-S006-V10",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed6-Variant10",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S006",
        "variant_type": "链式复合赋值变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with chained compound assignments, what is the value of variable count after executing both compound assignment statements?",
```

```
        "code": "count = 10\nincrement = 2\nmultiplier = 3\nprint(f\"Initial count: {count}\")\nccount += increment\nccount *= multiplier\nprint(f\"Final count: {ccount}\")",
        "answer": 36
    },
},
{
    "id": "SL-AS-S007-V01",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-variant1",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "浮点数交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with float swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 3.14\nb = 2.71\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\n\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 3.14
    }
},
{
    "id": "SL-AS-S007-V02",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "负数交换变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with negative number swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = -5\nb = 15\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\n\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": -5
    }
},
{
    "id": "SL-AS-S007-V03",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-variant3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "负数和浮点数混合交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "零值交换变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with zero value swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 0\nb = 42\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 0
    }
},
{
    "id": "SL-AS-S007-v04",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-Variant4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "布尔交换变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with boolean swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = True\nb = False\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": true
    }
},
{
    "id": "SL-AS-S007-v05",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-Variant5",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "大数交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    }
}
```

```
},
"task": {
    "description": "Given the following code snippet with large number swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = 1000000\nb = 2000000\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
    "answer": 1000000
},
},
{
    "id": "SL-AS-S007-v06",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-Variant6",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "字符交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with character swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 'x'\nb = 'y'\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": "x"
    }
},
{
    "id": "SL-AS-S007-v07",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed7-Variant7",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "计算结果交换变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with calculated values swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "a = 2 ** 3\nb = 3 ** 2\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 8
    }
}
```

```
},
{
  "id": "SL-AS-S007-v08",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed7-Variant8",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S007",
    "variant_type": "None值交换变式",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with None value swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = None\nb = 100\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
    "answer": null
  }
},
{
  "id": "SL-AS-S007-v09",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed7-Variant9",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
    "source": "Generated",
    "source_seed": "SL-AS-S007",
    "variant_type": "复数交换变式",
    "language": "python",
    "difficulty": "hard",
    "intervention": 0
  },
  "task": {
    "description": "Given the following code snippet with complex number swap, what is the real part of variable b after executing the multiple assignment statement `a, b = b, a`?",
    "code": "a = 3 + 4j\nb = 1 + 2j\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b,\nb\nprint(f\"After swap: a={a}, b={b}\")",
    "answer": 3
  }
},
{
  "id": "SL-AS-S007-v10",
  "metadata": {
    "name": "StatementLevel-Assignment-Seed7-Variant10",
    "category": "Statement-Level",
    "subcategory": "Assignment",
    "type": "variant",
```

```
        "source": "Generated",
        "source_seed": "SL-AS-S007",
        "variant_type": "变量引用交换变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with variable reference swap, what is the value of variable b after executing the multiple assignment statement `a, b = b, a`?",
        "code": "x = 50\ny = 60\na = x\nb = y\nprint(f\"Before swap: a={a}, b={b}\")\n\na, b = b, a\nprint(f\"After swap: a={a}, b={b}\")",
        "answer": 50
    }
},
{
    "id": "SL-AS-S008-v01",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant1",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "更长序列解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with longer sequence unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = numbers`?",
        "code": "numbers = [1, 2, 3, 4, 5, 6, 7, 8]\n\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 6
    }
},
{
    "id": "SL-AS-S008-v02",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant2",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "字符串解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
}
```

```
"task": {
    "description": "Given the following code snippet with string unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = text`?",
    "code": "text = \"python\"\nfirst, second, *rest = text\nprint(f\"first={first}, second={second}, rest={rest}\")",
    "answer": 4
},
{
    "id": "SL-AS-S008-V03",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant3",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "单元素序列解包变式",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with single element unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = numbers`?",
        "code": "numbers = [42]\nfirst, second, *rest = numbers\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 0
    }
},
{
    "id": "SL-AS-S008-V04",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant4",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "元组解包变式",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with tuple unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = data`?",
        "code": "data = (10, 20, 30, 40, 50, 60)\nfirst, second, *rest = data\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 4
}
```

```
        },
    },
    {
        "id": "SL-AS-S008-v05",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed8-Variant5",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S008",
            "variant_type": "三变量解包变式",
            "language": "python",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with three-variable unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, third, *rest = numbers`?",
            "code": "numbers = [1, 2, 3, 4, 5]\nfirst, second, third, *rest =\nnumbers\nprint(f\"first={first}, second={second}, third={third}, rest={rest}\")",
            "answer": 2
        }
    },
    {
        "id": "SL-AS-S008-v06",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed8-Variant6",
            "category": "Statement-Level",
            "subcategory": "Assignment",
            "type": "variant",
            "source": "Generated",
            "source_seed": "SL-AS-S008",
            "variant_type": "范围对象解包变式",
            "language": "python",
            "difficulty": "medium",
            "intervention": 0
        },
        "task": {
            "description": "Given the following code snippet with range object unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = data`?",
            "code": "data = range(1, 8)\nfirst, second, *rest = data\nprint(f\"first={first}, second={second}, rest={list(rest)}\")",
            "answer": 5
        }
    },
    {
        "id": "SL-AS-S008-v07",
        "metadata": {
            "name": "StatementLevel-Assignment-Seed8-Variant7",
            "category": "Statement-Level",
```

```
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "空列表解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet where unpacking an empty list would cause an error, what would happen when executing the unpacking assignment `first, second, *rest = numbers`?",
        "code": "numbers = []\n# This would cause a ValueError\n# first, second, *rest = numbers",
        "answer": "ValueError"
    }
},
{
    "id": "SL-AS-S008-V08",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant8",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "生成器解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with generator unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = gen`?",
        "code": "gen = (x for x in [1, 2, 3, 4, 5, 6])\nfirst, second, *rest = gen\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 4
    }
},
{
    "id": "SL-AS-S008-V09",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant9",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "字典值解包变式",
        "language": "python",
        "difficulty": "hard",
```

```

        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with dictionary values unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = data.values()`?",
        "code": "data = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}\nfirst, second, *rest = data.values()\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 4
    }
},
{
    "id": "SL-AS-S008-v10",
    "metadata": {
        "name": "StatementLevel-Assignment-Seed8-Variant10",
        "category": "Statement-Level",
        "subcategory": "Assignment",
        "type": "variant",
        "source": "Generated",
        "source_seed": "SL-AS-S008",
        "variant_type": "嵌套列表解包变式",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0
    },
    "task": {
        "description": "Given the following code snippet with nested list unpacking, what is the length of the list stored in variable rest after executing the unpacking assignment `first, second, *rest = data[0]`?",
        "code": "data = [[1, 2, 3, 4, 5], [6, 7, 8]]\nfirst, second, *rest = data[0]\nprint(f\"first={first}, second={second}, rest={rest}\")",
        "answer": 3
    }
}

```

1E - 常量赋值 [Constant] (37)

```

# 常量赋值变式生成提示词

## 任务目标
基于给定的常量赋值种子任务，生成多样化的变式任务来全面测试大模型对常量值语义、字面量表示、类型推断和常量特性的理解能力。

## 变式生成维度

### 1. 常量类型变式
- **整数常量变式**：正整数、负整数、零、不同进制表示（0x, 0o, 0b）
- **浮点常量变式**：小数、科学计数法、特殊值（inf, -inf, nan）
- **字符串常量变式**：单引号、双引号、三引号、原始字符串、转义字符
- **布尔常量变式**：True, False 及其在不同语言中的表示
- **空值常量变式**：None, NULL, null, undefined 等空值表示
- **字符常量变式**：单字符、Unicode字符、特殊字符

```

2. 字面量表示变式

- **数值字面量变式**: 整数、浮点数、复数的不同写法
- **进制表示变式**: 二进制(0b)、八进制(0o)、十六进制(0x)
- **科学计数法变式**: 1e5, 2.5e-3 等指数表示
- **字符串字面量变式**: 单引号vs双引号、多行字符串、格式化字符串
- **转义序列变式**: \n, \t, \\, \", \' 等转义字符
- **Unicode字面量变式**: \u编码、\x编码、原始Unicode字符

3. 常量定义变式

- **const关键字变式**: C/C++/JavaScript中的const声明
- **final关键字变式**: Java中的final常量
- **static final变式**: Java中的类常量
- **#define宏变式**: C预处理器宏定义
- **enum常量变式**: 枚举类型的常量定义
- **readonly变式**: 只读属性的常量定义

4. 类型推断变式

- **隐式类型变式**: 编译器/解释器自动推断常量类型
- **显式类型变式**: 明确指定常量的数据类型
- **类型转换变式**: 常量在赋值时的自动类型转换
- **精度损失变式**: 高精度常量赋值给低精度变量
- **溢出变式**: 超出类型范围的常量赋值
- **类型兼容变式**: 不同但兼容类型间的常量赋值

5. 特殊值变式

- **边界值变式**: 最大值、最小值、临界值常量
- **特殊浮点变式**: NaN, Infinity, -Infinity, 0.0, -0.0
- **空值变式**: None, NULL, nullptr, null, undefined
- **默认值变式**: 类型的默认初始值常量
- **魔数变式**: 具有特殊含义的数字常量
- **符号常量变式**: 预定义的符号常量

6. 字符串常量变式

- **空字符串变式**: "", ' ', 不同语言的空字符串表示
- **多行字符串变式**: 三引号字符串、换行符处理
- **格式化字符串变式**: f-string, format string, template literal
- **原始字符串变式**: r"string", raw string literal
- **字节字符串变式**: b"string", bytes literal
- **Unicode字符串变式**: u"string", unicode literal

7. 复合常量变式

- **数组常量变式**: {1, 2, 3}, [1, 2, 3] 等数组字面量
- **结构体常量变式**: 结构体的字面量初始化
- **对象字面量变式**: JavaScript/Python的对象字面量
- **元组常量变式**: (1, 2, 3) 元组字面量
- **字典常量变式**: {"key": "value"} 字典字面量
- **集合常量变式**: {1, 2, 3} 集合字面量

8. 语言特性变式

- **Python特性变式**: 多种字符串表示、数值类型、True/False/None
- **C语言特性变式**: 整型常量、字符常量、指针常量、宏定义
- **Java特性变式**: final常量、字符串常量池、包装类常量

- **JavaScript特性变式**: `const/let`、模板字符串、`Symbol`常量
- **跨语言对比变式**: 相同常量在不同语言中的表示差异

9. 常量运算变式

- **常量表达式变式**: 编译时可计算的常量表达式
- **常量折叠变式**: 编译器优化的常量计算
- **常量传播变式**: 常量在代码中的传播和替换
- **运算结果常量变式**: 常量运算的结果类型和值
- **短路求值变式**: 常量条件的短路计算
- **三元运算常量变式**: 条件表达式中的常量选择

10. 内存和存储变式

- **字符串常量池变式**: 相同字符串常量的内存共享
- **静态存储变式**: 常量在静态存储区的分配
- **栈VS堆变式**: 常量在不同内存区域的存储
- **只读内存变式**: 常量存储在只读内存段
- **常量表变式**: 编译器生成的常量表
- **内联常量变式**: 编译时内联的常量替换

生成规则

基本要求

1. 每个种子任务生成10-15个变式
2. 确保每个变式维度至少覆盖2-3个变式
3. 保持常量赋值的核心语义，但改变常量类型和表示形式
4. 变式难度应该涵盖easy/medium/hard三个等级

难度分级标准

- **Easy**: 基础字面量常量，简单数据类型，直接赋值
- **Medium**: 特殊表示形式，类型转换，字符串处理，进制转换
- **Hard**: 特殊值处理，复合常量，跨语言差异，内存模型，编译时计算

特殊考虑点

1. **字面量准确性**: 确保常量字面量的语法正确
2. **类型一致性**: 保证常量类型与语言规范一致
3. **精度问题**: 注意浮点常量的精度表示
4. **字符编码**: 正确处理字符串的编码问题
5. **语言差异**: 考虑不同语言的常量表示差异

输出格式

生成的变式应该放在一个JSON数组中，每个变式对象用逗号分隔：

```
```json
[
 {
 "id": "SL-CT-S00X-V001",
 "metadata": {
 "name": "变式名称",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python/c/java/javascript",
 }
 }
]
```

```
 "difficulty": "easy/medium/hard",
 "intervention": 0,
 "variant_type": "变式类型标签"
 },
 "task": {
 "description": "清晰的任务描述",
 "code": "完整的代码片段",
 "answer": "准确的常量值"
 }
},
{下一个变式...}
]
```

质量检查要点

常量正确性：确保常量值的计算和表示正确

语法合规性：确保常量字面量在目标语言中语法正确

类型一致性：确保答案类型与常量类型匹配

精度准确性：浮点常量的精度表示要准确

示例变式类型

基于SL-CT-S001（整数常量）的变式示例：

进制表示变式：num = 0x2A (十六进制42)

负数变式：num = -42

科学计数法变式：num = 4.2e1

类型转换变式：num = int(42.0)

边界值变式：num = 2147483647 (最大int值)

基于SL-CT-S002（字符串常量）的变式示例：

转义字符变式：message = "Hello, \nworld!"

原始字符串变式：message = r"Hello, world!"

Unicode变式：message = "Hello, 世界!"

格式化字符串变式：message = f"Hello, {'world'}!"

字节字符串变式：message = b"Hello, world!"

基于SL-CT-S005（NULL指针）的变式示例：

指针常量变式：status = (void\*)0x0

不同NULL表示变式：status = 0 vs status = NULL

nullptr变式：status = nullptr (C++)

条件编译变式：#define NULL ((void\*)0)

请基于此提示词生成完整的变式集合，输出格式为包含所有变式的JSON数组。

```
[

{
 "id": "SL-CT-S001-v001",
 "metadata": {
 "name": "十六进制整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 }
}
```

```
 "variant_type": "进制表示变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = 0x2A`?",
 "code": "num = 0x2A\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
 "answer": 42
 }
},
{
 "id": "SL-CT-S001-V002",
 "metadata": {
 "name": "二进制整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "进制表示变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = 0b101010`?",
 "code": "num = 0b101010\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
 "answer": 42
 }
},
{
 "id": "SL-CT-S001-V003",
 "metadata": {
 "name": "八进制整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "进制表示变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = 0o52`?",
 "code": "num = 0o52\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
 "answer": 42
 }
},
{
 "id": "SL-CT-S001-V004",
 "metadata": {
```

```
 "name": "负整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "负数变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = -42`?",
 "code": "num = -42\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
 "answer": -42
 }
},
{
 "id": "SL-CT-S001-V005",
 "metadata": {
 "name": "科学计数法整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "科学计数法变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `num = int(4.2e1)`?",
 "code": "num = int(4.2e1)\nprint(f\"Number: {num}\")\nprint(f\"Type: {type(num)}\")",
 "answer": 42
 }
},
{
 "id": "SL-CT-S001-V006",
 "metadata": {
 "name": "边界值最大整数常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "边界值变式"
 },
 "task": {
```



```
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "常量定义变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable num in the statement `const num = 42`?",
 "code": "const num = 42;\nconsole.log(`Number: ${num}`);\nconsole.log(`Type: ${typeof num}`);",
 "answer": 42
 }
},
{
 "id": "SL-CT-S001-V010",
 "metadata": {
 "name": "C语言宏定义整数常量",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "宏定义变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value of NUM when used in the statement `int x = NUM`?",
 "code": "#include <stdio.h>\n#define NUM 42\nint main() {\n int x = NUM;\n printf(\"Number: %d\\n\", x);\n return 0;\n}",
 "answer": 42
 }
},
{
 "id": "SL-CT-S002-V001",
 "metadata": {
 "name": "单引号字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "字符串字面量变式"
 },
 "task": {
```



```
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "Unicode字面量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable message in the statement `message = \"Hello, 世界!\"`?",
 "code": "message = \"Hello, 世界!\"\nprint(f\"Message:\n{message}\")\nprint(f\"Length: {len(message)}\")",
 "answer": "Hello, 世界!"
 }
},
{
 "id": "SL-CT-S002-v005",
 "metadata": {
 "name": "格式化字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "格式化字符串变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable message in the statement `message = f\"Hello, {'world'}!\"`?",
 "code": "message = f\"Hello, {'world'}!\"\nprint(f\"Message:\n{message}\")\nprint(f\"Length: {len(message)}\")",
 "answer": "Hello, world!"
 }
},
{
 "id": "SL-CT-S002-v006",
 "metadata": {
 "name": "字节字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "字节字符串变式"
 },
 "task": {
```

```
 "description": "Given the following code snippet, what is the decoded string value of the bytes assigned to variable message in the statement `message = b\"Hello, world!\"`?",
 "code": "message = b\"Hello, world!\"\nprint(f\"Bytes: {message}\")\nprint(f\"Decoded: {message.decode('utf-8')}\")",
 "answer": "Hello, world!"
 },
},
{
 "id": "SL-CT-S002-v007",
 "metadata": {
 "name": "空字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "空字符串变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable message in the statement `message = \"\"`?",
 "code": "message = \"\"\nprint(f\"Message: '{message}'\")\nprint(f\"Length: {len(message)}\")",
 "answer": ""
 },
},
{
 "id": "SL-CT-S002-v008",
 "metadata": {
 "name": "多行字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "多行字符串变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable message in the statement using triple quotes?",
 "code": "message = """Hello,\nworld!\nHow are you?"""\nprint(f\"Message: {repr(message)}\")\nprint(f\"Lines: {len(message.split(chr(10)))}\")",
 "answer": "Hello,\nworld!\nHow are you?"
 },
},
{
 "id": "SL-CT-S002-v009",
 "metadata": {
```

```
 "name": "C语言字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "语言特性变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the string value assigned to variable message in the statement `char message[] = \\\"Hello, World!\\\"`?",
 "code": "#include <stdio.h>\n#include <string.h>\nint main() {\n char message[] = \\\"Hello, world!\\\";\n printf(\"Message: %s\\n\", message);\n printf(\"Length: %lu\\n\", strlen(message));\n return 0;\n}",
 "answer": "Hello, World!"
 }
},
{
 "id": "SL-CT-S002-V010",
 "metadata": {
 "name": "JavaScript模板字符串常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "模板字符串变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable message in the statement using template literals?",
 "code": "const name = 'World';\nconst message = `Hello,\n${name}!`;\nconsole.log(`Message: ${message}`);\nconsole.log(`Length: ${message.length}`);",
 "answer": "Hello, World!"
 }
},
{
 "id": "SL-CT-S003-V001",
 "metadata": {
 "name": "False布尔常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "布尔常量变式"
 },
}
```

```
"task": {
 "description": "Given the following code snippet, what is the value assigned to variable is_valid in the statement `is_valid = False`?",
 "code": "is_valid = False\nprint(f\"valid: {is_valid}\")\nprint(f\"Type: {type(is_valid)}\")",
 "answer": false
},
{
 "id": "SL-CT-S003-v002",
 "metadata": {
 "name": "C语言真值常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "语言特性变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the integer value assigned to variable is_valid in the statement `int is_valid = 1`?",
 "code": "#include <stdio.h>\nint main() {\n int is_valid = 1;\n\n printf(\"valid: %d\\n\", is_valid);\n printf(\"Boolean: %s\\n\", is_valid ? \"true\" : \"false\");\n\n return 0;\n}",
 "answer": 1
 }
},
{
 "id": "SL-CT-S003-v003",
 "metadata": {
 "name": "Java布尔常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "java",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "语言特性变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable isvalid in the statement `boolean isvalid = true`?",
 "code": "public class Test {\n public static void main(String[] args) {\n boolean isvalid = true;\n System.out.println(\"Valid: \" + isvalid);\n System.out.println(\"Class: \" + Boolean.class.getSimpleName());\n }\n}",
 "answer": true
 }
},
```

```
"id": "SL-CT-S003-V004",
"metadata": {
 "name": "JavaScript布尔常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "语言特性变式"
},
"task": {
 "description": "Given the following code snippet, what is the value assigned to variable isvalid in the statement `const isvalid = false`?",
 "code": "const isvalid = false;\nconsole.log(`valid: ${isvalid}`);\nconsole.log(`Type: ${typeof isvalid}`);",
 "answer": false
},
{
 "id": "SL-CT-S003-V005",
 "metadata": {
 "name": "条件表达式布尔常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "常量表达式变式"
},
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable is_valid in the statement `is_valid = 5 > 3`?",
 "code": "is_valid = 5 > 3\nprint(f\"valid: {is_valid}\")\nprint(f\"Type: {type(is_valid)}\")",
 "answer": true
 }
},
{
 "id": "SL-CT-S004-V001",
 "metadata": {
 "name": "科学计数法浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "科学计数法变式"
}
```

```
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable pi in the statement `pi = 3.14159e0`?",
 "code": "pi = 3.14159e0\nradius = 5\narea = pi * radius ** 2\nprint(f\"Pi: {pi}\")\nprint(f\"Area: {area}\")",
 "answer": "3.141592653589793"
 }
 },
 {
 "id": "SL-CT-S004-V002",
 "metadata": {
 "name": "负科学计数法浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "科学计数法变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable small_num in the statement `small_num = 2.5e-3`?",
 "code": "small_num = 2.5e-3\nprint(f\"Small number: {small_num}\")\nprint(f\"Scientific: {small_num:.2e}\")",
 "answer": "0.0025"
 }
 },
 {
 "id": "SL-CT-S004-V003",
 "metadata": {
 "name": "无穷大浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "特殊浮点变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable infinity in the statement `infinity = float('inf')`?",
 "code": "infinity = float('inf')\nprint(f\"Infinity: {infinity}\")\nprint(f\"Is infinite: {infinity == float('inf')}\")",
 "answer": "inf"
 }
 },
 {
 "id": "SL-CT-S004-V004",
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable infinity in the statement `infinity = float('inf')`?",
 "code": "infinity = float('inf')\nprint(f\"Infinity: {infinity}\")\nprint(f\"Is infinite: {infinity == float('inf')}\")",
 "answer": "inf"
 }
 }
]
```

```
"metadata": {
 "name": "NaN浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "特殊浮点变式"
},
"task": {
 "description": "Given the following code snippet, what is the value assigned to variable not_a_number in the statement `not_a_number = float('nan')`?",
 "code": "import math\nnot_a_number = float('nan')\nprint(f\"NaN: {not_a_number}\")\nprint(f\"Is NaN: {math.isnan(not_a_number)}\")",
 "answer": "nan"
},
{
 "id": "SL-CT-S004-V005",
 "metadata": {
 "name": "负零浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "特殊浮点变式"
},
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable neg_zero in the statement `neg_zero = -0.0`?",
 "code": "neg_zero = -0.0\npos_zero = 0.0\nprint(f\"Negative zero: {neg_zero}\")\nprint(f\"Equal to positive zero: {neg_zero == pos_zero}\")",
 "answer": "-0.0"
},
{
 "id": "SL-CT-S004-V006",
 "metadata": {
 "name": "分数表示浮点常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "类型转换变式"
},
}
```

```
"task": {
 "description": "Given the following code snippet, what is the value assigned to variable pi_approx in the statement `pi_approx = 22/7`?",
 "code": "pi_approx = 22/7\nprintf(f\"Pi approximation:\n{pi_approx}\")\nprintf(f\"Rounded: {round(pi_approx, 5)}\")",
 "answer": 3.142857142857143
},
{
 "id": "SL-CT-S005-v001",
 "metadata": {
 "name": "整数零指针常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "指针常量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the integer value of the pointer status after executing the statement `status = 0`?",
 "code": "#include <stdio.h>\nvoid *status = 0;\nint value = 100;\nif (status == NULL) {\n printf(\"Status is null\\n\");\n} else {\n printf(\"Status is not null\\n\");\n}",
 "answer": 0
 }
},
{
 "id": "SL-CT-S005-v002",
 "metadata": {
 "name": "强制转换NULL指针常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "指针常量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the integer value of the pointer status after executing the statement `status = (void*)0x0`?",
 "code": "#include <stdio.h>\nvoid *status = (void*)0x0;\nint value = 100;\nif (status == NULL) {\n printf(\"Status is null\\n\");\n} else {\n printf(\"Status is not null\\n\");\n}",
 "answer": 0
 }
},
{
```

```
"id": "SL-CT-S005-V003",
"metadata": {
 "name": "nullptr指针常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "cpp",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "nullptr变式"
},
"task": {
 "description": "Given the following code snippet, what is the integer value of the pointer status after executing the statement `status = nullptr`?",
 "code": "#include <iostream>\nusing namespace std;\nint main() {\n void\n*status = nullptr;\n if (status == nullptr) {\n cout << \"Status is null\" <<\n endl;\n } else {\n cout << \"Status is not null\" << endl;\n }\n return\n0;\n}",
 "answer": 0
},
{
 "id": "SL-CT-S005-V004",
 "metadata": {
 "name": "Python None常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "空值常量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable status in the statement `status = None`?",
 "code": "status = None\nprint(f\"Status: {status}\")\nprint(f\"Type: {type(status)}\")\nprint(f\"Is None: {status is None}\")",
 "answer": null
 }
},
{
 "id": "SL-CT-S005-V005",
 "metadata": {
 "name": "JavaScript null常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "easy",
 }
}
```

```

 "intervention": 0,
 "variant_type": "空值常量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable status in the statement `const status = null`?",
 "code": "const status = null;\nconsole.log(`status: ${status}`);\nconsole.log(`Type: ${typeof status}`);\nconsole.log(`Is null: ${status === null}`);",
 "answer": "null"
 }
},
{
 "id": "SL-CT-S005-V006",
 "metadata": {
 "name": "JavaScript undefined常量赋值",
 "category": "Statement-Level",
 "subcategory": "Constant",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "空值常量变式"
 },
 "task": {
 "description": "Given the following code snippet, what is the value assigned to variable status in the statement `const status = undefined`?",
 "code": "const status = undefined;\nconsole.log(`status: ${status}`);\nconsole.log(`Type: ${typeof status}`);\nconsole.log(`Is undefined: ${status === undefined}`);",
 "answer": "undefined"
 }
}
]

```

## 1F - 大混合 (50)

```

语句级推理混合变式生成提示词

任务目标
基于给定的语句级推理种子任务，生成融合多种语句级推理类型的综合变式任务，全面测试大模型对算术运算、布尔运算、API/函数调用、变量赋值和常量赋值等多种语句级推理能力的综合理解和交叉应用。

核心设计理念
不局限于单一推理类型，而是创造包含多种语句级推理元素的复合场景，测试模型在复杂语句环境下的综合推理能力。

混合变式生成维度

1. 运算与赋值混合变式
- **算术运算+变量赋值**：复杂表达式的计算结果赋值给变量
- **布尔运算+条件赋值**：基于布尔表达式结果的条件性赋值
- **位运算+复合赋值**：位运算符与复合赋值运算符的组合

```

- \*\*运算优先级+多重赋值\*\*：复杂运算优先级与多变量同时赋值
- \*\*类型转换+算术运算\*\*：隐式/显式类型转换与数学计算的结合

### ### 2. 函数调用与运算混合变式

- \*\*函数返回值+算术运算\*\*：函数调用结果参与数学计算
- \*\*嵌套函数调用+布尔判断\*\*：多层函数调用的返回值用于逻辑判断
- \*\*API调用+错误处理\*\*：函数调用失败时的错误码与布尔逻辑结合
- \*\*参数计算+函数调用\*\*：复杂表达式作为函数参数
- \*\*函数副作用+变量状态\*\*：函数调用对全局变量的影响

### ### 3. 常量与变量交互变式

- \*\*常量定义+变量运算\*\*：使用常量参与变量的算术运算
- \*\*魔数替换+表达式简化\*\*：将魔数提取为常量对表达式的影响
- \*\*常量折叠+运行时计算\*\*：编译时常量计算与运行时变量计算的区别
- \*\*枚举常量+逻辑判断\*\*：枚举值在条件判断中的使用
- \*\*宏定义+复杂表达式\*\*：宏展开对复杂表达式计算的影响

### ### 4. 控制流与推理混合变式

- \*\*条件表达式+多类型运算\*\*：`if-else`中包含不同类型的运算
- \*\*短路求值+函数调用\*\*：逻辑短路与函数调用副作用的结合
- \*\*循环计数+累积运算\*\*：循环变量与累积计算的结合
- \*\*break/continue+状态变化\*\*：控制流改变对变量状态的影响
- \*\*异常处理+资源管理\*\*：`try-catch`与内存/文件资源的管理

### ### 5. 内存与引用混合变式

- \*\*指针运算+数组访问\*\*：指针算术与数组元素访问的结合
- \*\*引用传递+函数修改\*\*：引用参数在函数中的修改对原变量的影响
- \*\*动态分配+计算使用\*\*：`malloc/new`的结果用于后续计算
- \*\*别名效应+赋值操作\*\*：多个变量指向同一内存时的赋值影响
- \*\*作用域+生命周期\*\*：变量作用域与对象生命周期的交互

### ### 6. 类型系统混合变式

- \*\*隐式转换+运算精度\*\*：类型自动转换对计算精度的影响
- \*\*泛型+具体类型\*\*：泛型函数在具体类型下的行为
- \*\*多态+方法调用\*\*：对象多态性与方法调用结果的关系
- \*\*联合类型+类型检查\*\*：`Union`类型在运行时的类型检查
- \*\*void指针+类型转换\*\*：`void*`指针的类型转换与使用

### ### 7. 并发与同步混合变式

- \*\*原子操作+共享变量\*\*：原子操作对共享变量状态的影响
- \*\*锁机制+资源访问\*\*：同步锁与共享资源访问的结合
- \*\*线程局部+全局状态\*\*：线程局部存储与全局变量的交互
- \*\*信号量+条件判断\*\*：信号量状态与条件逻辑的结合
- \*\*竞态条件+结果预测\*\*：多线程竞态对计算结果的影响

### ### 8. 错误处理混合变式

- \*\*边界检查+运算结果\*\*：数组边界检查与索引计算的结合
- \*\*空值检查+函数调用\*\*：`NULL`检查与后续函数调用的安全性
- \*\*溢出检测+数值运算\*\*：整数溢出检测与算术运算的结合
- \*\*资源泄漏+状态跟踪\*\*：内存泄漏与程序状态跟踪的结合
- \*\*异常传播+错误码\*\*：异常处理与错误码返回的混合使用

### ### 9. 平台与环境混合变式

- **\*\*字节序+数据解释\*\***: 大小端字节序对数据解释的影响
- **\*\*编译器优化+代码行为\*\***: 编译器优化对代码执行结果的影响
- **\*\*运行时环境+API行为\*\***: 不同运行时环境下API的行为差异
- **\*\*内存对齐+结构体布局\*\***: 内存对齐要求对结构体布局的影响
- **\*\*操作系统+系统调用\*\***: 不同操作系统下系统调用的行为差异

### ### 10. 语言特性混合变式

- **\*\*语法糖+底层实现\*\***: 语言语法糖与其底层实现的对应关系
- **\*\*运算符重载+自定义行为\*\***: 运算符重载对表达式计算的影响
- **\*\*闭包+变量捕获\*\***: 闭包对外部变量的捕获和修改
- **\*\*装饰器+函数行为\*\***: 装饰器对函数行为的修改
- **\*\*元编程+代码生成\*\***: 元编程技术对代码执行的影响

## ## 复杂度层次设计

### ### 简单混合 (Easy)

- 2种推理类型的基础组合
- 线性执行流程, 无复杂控制逻辑
- 明确的输入输出关系
- 基础数据类型, 无特殊边界情况

### ### 中等混合 (Medium)

- 3-4种推理类型的中度组合
- 包含条件分支或简单循环
- 涉及类型转换或简单错误处理
- 中等复杂度的表达式和函数调用

### ### 复杂混合 (Hard)

- 4种以上推理类型的深度融合
- 复杂控制流和嵌套结构
- 多层次的类型转换和错误处理
- 涉及内存管理、并发或平台相关特性
- 需要深入理解语言语义和执行模型

## ## 生成策略

### ### 种子分析策略

1. **\*\*识别核心推理类型\*\***: 分析种子任务的主要推理类型(A/B/C/D/E)
2. **\*\*提取关键特征\*\***: 识别种子中的关键语言特性和计算模式
3. **\*\*设计融合点\*\***: 确定可以融合其他推理类型的切入点
4. **\*\*保持核心复杂度\*\***: 在添加新元素时保持原有难度层次

### ### 变式设计原则

1. **\*\*渐进式复杂化\*\***: 从简单混合逐步增加到复杂混合
2. **\*\*语义一致性\*\***: 确保混合后的代码语义清晰且正确
3. **\*\*测试价值最大化\*\***: 每个混合变式都应测试特定的推理能力组合
4. **\*\*现实场景导向\*\***: 优先设计在实际编程中常见的混合场景

### ### 质量保证机制

1. **\*\*多维度验证\*\***: 从语法、语义、逻辑多个维度验证变式正确性
2. **\*\*边界情况覆盖\*\***: 确保包含各种边界和异常情况
3. **\*\*跨语言一致性\*\***: 在语言移植时保持推理逻辑的一致性
4. **\*\*难度梯度合理\*\***: 确保easy/medium/hard的难度分布合理

```
输出格式要求
```

生成的混合变式应该放在一个JSON数组中：

```
```json
[
  {
    "id": "SL-MIX-S00X-V001",
    "metadata": {
      "name": "混合变式名称",
      "category": "Statement-Level",
      "subcategory": "Mixed",
      "type": "variant",
      "source": "Generated",
      "language": "目标语言",
      "difficulty": "easy/medium/hard",
      "intervention": 0,
      "variant_type": "推理类型组合标签",
      "mixed_types": ["Arithmetic", "Boolean", "API Call", "Assignment",
      "Constant"]
    },
    "task": {
      "description": "综合性任务描述",
      "code": "包含多种推理类型的代码片段",
      "answer": "准确的答案"
    }
  },
  {下一个变式...}
]
```

特殊字段说明

mixed_types: 标识该变式包含的推理类型组合

variant_type: 描述具体的混合模式(如"arithmetic_assignment", "boolean_api_call"等)

生成目标

生成50-100个混合变式，确保：

每种难度等级至少包含4-6个变式

涵盖至少8-10种不同的推理类型组合

包含2-3个跨语言的混合变式

至少包含3-4个现实编程场景的复杂混合案例

请基于此提示词，为给定的种子任务生成全面的混合变式集合，输出格式为包含所有变式的JSON数组

```
[

  {
    "id": "SL-MIX-S001-V001",
    "metadata": {
      "name": "Arithmetic Operation and Variable Assignment Mix",
      "category": "Statement-Level",
      "subcategory": "Mixed",
      "type": "variant",
      "source": "Generated",
      "language": "python",
```

```
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "arithmetic_assignment",
        "mixed_types": ["Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = base * 2 + offset`?",
        "code": "base = 15\noffset = 7\ntemp = base * 2\nresult = base * 2 + offset\nprint(f\"Result: {result}\")",
        "answer": 37
    }
},
{
    "id": "SL-MIX-S001-V002",
    "metadata": {
        "name": "Boolean Operation and Function Call Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boolean_api_call",
        "mixed_types": ["Boolean", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `len(text) > 0 and text.isdigit()`?",
        "code": "text = \"12345\"\nlength = len(text)\nis_digit = text.isdigit()\nresult = len(text) > 0 and text.isdigit()\nprint(f\"Result: {result}\")",
        "answer": true
    }
},
{
    "id": "SL-MIX-S001-V003",
    "metadata": {
        "name": "Constant and Arithmetic Operation Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "constant_arithmetic",
        "mixed_types": ["Constant", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable area after executing the statement `area = PI * radius ** 2`?",
        "code": "PI = 3.14159\nradius = 5\narea = PI * radius ** 2\nprint(f\"Area: {area}\")",
        "answer": 78.54
    }
}
```

```
        "code": "PI = 3.14159\nradius = 3\n diameter = 2 * radius\narea = PI * radius **\n2\nprint(f\"Area: {area}\")",
        "answer": 28.27431
    },
},
{
    "id": "SL-MIX-S001-V004",
    "metadata": {
        "name": "Function Call and Assignment Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "api_call_assignment",
        "mixed_types": ["API Call", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable min_val after executing the multiple assignment `max_val, min_val = max(numbers), min(numbers)`?",
        "code": "numbers = [8, 3, 15, 1, 9, 12]\nsum_val = sum(numbers)\nmax_val, min_val = max(numbers), min(numbers)\nprint(f\"Max: {max_val}, Min: {min_val}\")",
        "answer": 1
    },
},
{
    "id": "SL-MIX-S001-V005",
    "metadata": {
        "name": "Bitwise Operation and Boolean Logic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "bitwise_boolean",
        "mixed_types": ["Arithmetic", "Boolean"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of the boolean expression `(flags & mask) != 0 and position < 8`?",
        "code": "flags = 0b1010\nposition = 3\nmask = 1 << position\ncheck = (flags & mask) != 0 and position < 8\nprint(f\"Check: {check}\")",
        "answer": false
    },
},
{
    "id": "SL-MIX-S001-V006",
    "metadata": {
```

```
        "name": "Type Conversion and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "type_conversion_arithmetic",
        "mixed_types": ["Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = int(average * weight) + bonus`?",
        "code": "scores = [85, 90, 78]\naverage = sum(scores) / len(scores)\nweight = 0.9\nbonus = 5\nresult = int(average * weight) + bonus\nprint(f\"Result: {result}\")",
        "answer": 81
    }
},
{
    "id": "SL-MIX-S001-V007",
    "metadata": {
        "name": "Nested Function Call and Assignment Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "nested_api_assignment",
        "mixed_types": ["API Call", "Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = len(str(max(numbers) + min(numbers)))`?",
        "code": "numbers = [15, 8, 23, 4, 19]\nmax_num = max(numbers)\nmin_num = min(numbers)\nsum_extremes = max_num + min_num\nresult = len(str(max(numbers) + min(numbers)))\nprint(f\"Result: {result}\")",
        "answer": 2
    }
},
{
    "id": "SL-MIX-S001-V008",
    "metadata": {
        "name": "Conditional Expression Multi-Type Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
```

```
        "variant_type": "conditional_multi_type",
        "mixed_types": ["Boolean", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = value * 2 if value > threshold else value // 2`?",
        "code": "value = 15\nthreshold = 10\ndouble_val = value * 2\nhalf_val = value // 2\nresult = value * 2 if value > threshold else value // 2\nprint(f\"Result: {result}\")",
        "answer": 30
    }
},
{
    "id": "SL-MIX-S001-V009",
    "metadata": {
        "name": "String Operation and Boolean Logic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "string_boolean",
        "mixed_types": ["API Call", "Boolean", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable is_valid after executing the statement `is_valid = text.startswith('A') and len(text) >= 5`?",
        "code": "text = \"Apple\"\nfirst_char = text[0]\nlength = len(text)\nstarts_with_a = text.startswith('A')\nis_valid = text.startswith('A') and len(text) >= 5\nprint(f\"valid: {is_valid}\")",
        "answer": true
    }
},
{
    "id": "SL-MIX-S001-V010",
    "metadata": {
        "name": "Compound Assignment and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "compound_assignment_arithmetic",
        "mixed_types": ["Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable counter after executing the statement `counter *= step + 1`?",
        "code": "step = 2\nstep += 1\ncounter = 1\ncounter *= step + 1\nprint(f\"counter: {counter}\")",
        "answer": 6
    }
}
```

```
        "code": "counter = 8\nstep = 2\nincrement = step + 1\ncounter *= step +\n1\nprint(f\"Counter: {counter}\")",
        "answer": 24
    },
},
{
    "id": "SL-MIX-S001-V011",
    "metadata": {
        "name": "List Comprehension and Boolean Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "list_comprehension_boolean",
        "mixed_types": ["API Call", "Boolean", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable count after executing the statement `count = len([x for x in numbers if x % 2 == 0 and x > 5])`?",
        "code": "numbers = [2, 7, 8, 3, 12, 4, 15, 6]\neven_numbers = [x for x in\nnumbers if x % 2 == 0]\nlarge_numbers = [x for x in numbers if x > 5]\ncount = len([x for x\nin numbers if x % 2 == 0 and x > 5])\nprint(f\"Count: {count}\")",
        "answer": 3
    },
},
{
    "id": "SL-MIX-S001-V012",
    "metadata": {
        "name": "Dictionary Access and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "dict_access_arithmetic",
        "mixed_types": ["Assignment", "Arithmetic", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable total after executing the statement `total = data['price'] * data['quantity'] +\n            data['tax']`?",
        "code": "data = {'price': 25, 'quantity': 3, 'tax': 5}\nbase_cost =\n            data['price'] * data['quantity']\ntax_amount = data['tax']\ntotal = data['price'] *\n            data['quantity'] + data['tax']\nprint(f\"Total: {total}\")",
        "answer": 80
    },
},
{
}
```

```
{
  "id": "SL-MIX-S001-V013",
  "metadata": {
    "name": "Short Circuit Evaluation Mix",
    "category": "Statement-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0,
    "variant_type": "short_circuit_evaluation",
    "mixed_types": ["Boolean", "API Call"]
  },
  "task": {
    "description": "Given the following code snippet, what is the value of the boolean expression `len(data) > 0 and data[0] > 10`?",
    "code": "data = [15, 8, 3]\nlength = len(data)\nfirst_element = data[0] if\nlen(data) > 0 else None\nresult = len(data) > 0 and data[0] > 10\nprint(f\"Result:\n{result}\")",
    "answer": true
  }
},
{
  "id": "SL-MIX-S001-V014",
  "metadata": {
    "name": "Mathematical Function and Assignment Mix",
    "category": "Statement-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0,
    "variant_type": "math_function_assignment",
    "mixed_types": ["API Call", "Assignment", "Arithmetic"]
  },
  "task": {
    "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = round(math.sqrt(value), 2)`?",
    "code": "import math\nvalue = 50\nsquare_root = math.sqrt(value)\nresult = round(math.sqrt(value), 2)\nprint(f\"Result: {result}\")",
    "answer": 7.07
  }
},
{
  "id": "SL-MIX-S001-V015",
  "metadata": {
    "name": "String Formatting and Arithmetic Mix",
    "category": "Statement-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 0
  }
}
```

```
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "string_format_arithmetic",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the length of the string stored in variable formatted after executing the statement `formatted = f'{value:.2f}%'`?",
        "code": "value = 85.6789\nrounded_value = round(value, 2)\nformatted = f'{value:.2f}%'\\nlength = len(formatted)\\nprint(f\"Formatted: {formatted}, Length: {length}\")",
        "answer": 6
    }
},
{
    "id": "SL-MIX-S001-V016",
    "metadata": {
        "name": "Range and Sum Function Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "range_sum_mix",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable total after executing the statement `total = sum(range(start, end, step))`?",
        "code": "start = 2\\nend = 10\\nstep = 2\\nrange_values = list(range(start, end, step))\\ntotal = sum(range(start, end, step))\\nprint(f\"Total: {total}\")",
        "answer": 20
    }
},
{
    "id": "SL-MIX-S001-V017",
    "metadata": {
        "name": "Complex Boolean and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complex_boolean_arithmetic",
        "mixed_types": ["Boolean", "Arithmetic", "Assignment"]
    },
    "task": {
```



```
        "code": "flags = 0b1100\nmask = 0b0011\nand_result = flags & mask\nnor_result = flags | mask\nxor_result = flags ^ mask\ncondition = (flags & mask) == 0\nresult = (flags | mask) if (flags & mask) == 0 else (flags ^ mask)\nprint(f\"Result: {result}\")",
        "answer": 15
    }
},
{
    "id": "SL-MIX-S001-V020",
    "metadata": {
        "name": "List Slicing and Mathematical Operations Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "list_slicing_math",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = sum(numbers[::2]) - sum(numbers[1::2])`?",
        "code": "numbers = [10, 3, 8, 5, 12, 7, 4]\neven_indices = numbers[::2]\nodd_indices = numbers[1::2]\nsum_even = sum(even_indices)\nsum_odd = sum(odd_indices)\nresult = sum(numbers[::2]) - sum(numbers[1::2])\nprint(f\"Result: {result}\")",
        "answer": 19
    }
},
{
    "id": "SL-MIX-S001-V021",
    "metadata": {
        "name": "C Pointer Arithmetic and Assignment Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "pointer_arithmetic_assignment",
        "mixed_types": ["Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the value that ptr points to after executing the statement `ptr = arr + 3`?",
        "code": "#include <stdio.h>\nint arr[] = {10, 20, 30, 40, 50};\nint *ptr = arr;\nprintf(\"Initial: %d\\n\", *ptr);\nptr = arr + 3;\nprintf(\"After: %d\\n\", *ptr);",
        "answer": 40
    }
},
{

```

```
"id": "SL-MIX-S001-V022",
"metadata": {
    "name": "C Function Call and Boolean Logic Mix",
    "category": "Statement-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 0,
    "variant_type": "c_function_boolean",
    "mixed_types": ["API Call", "Boolean", "Assignment"]
},
"task": {
    "description": "Given the following C code snippet, what is the value of the boolean expression `strlen(str) > 3 && str[0] == 'T'` when str is \"Test\"?",
    "code": "#include <stdio.h>\n#include <string.h>\nchar *str = \"Test\";\nsize_t\nlength = strlen(str);\nchar first_char = str[0];\nint result = strlen(str) > 3 && str[0] == 'T';\nprintf(\"Result: %d\\n\", result);",
    "answer": 1
},
{
    "id": "SL-MIX-S001-V023",
    "metadata": {
        "name": "C Struct Access and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "struct_access_arithmetic",
        "mixed_types": ["Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the value of variable area after executing the statement `area = rect.width * rect.height`?",
        "code": "#include <stdio.h>\n\nstruct {\n    int width;\n    int height;\n} Rectangle;\n\nRectangle rect = {8, 5};\n\nint perimeter = 2 * (rect.width + rect.height);\n\nint area = rect.width * rect.height;\n\nprintf(\"Area: %d\\n\", area);",
        "answer": 40
    }
},
{
    "id": "SL-MIX-S001-V024",
    "metadata": {
        "name": "C Memory Allocation and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 0,
        "variant_type": "memory_allocation_arithmetic",
        "mixed_types": ["Assignment", "Arithmetic"]
    }
}
```

```
        "language": "c",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "memory_allocation_arithmetic",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the number of bytes allocated by the statement `ptr = malloc(count * sizeof(double))` when count is 6?",
        "code": "#include <stdio.h>\n#include <stdlib.h>\nint count = 6;\nsize_t element_size = sizeof(double);\nsize_t total_size = count * sizeof(double);\nvoid *ptr = malloc(count * sizeof(double));\nprintf(\"Allocated: %zu bytes\\n\", total_size);",
        "answer": 48
    }
},
{
    "id": "SL-MIX-S001-V025",
    "metadata": {
        "name": "Dictionary Comprehension and Boolean Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "dict_comprehension_boolean",
        "mixed_types": ["API Call", "Boolean", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet, what is the number of items in the dictionary after executing the statement `result = {k: v for k, v in data.items() if v > 10 and k.startswith('item')}?`",
        "code": "data = {'item1': 15, 'test2': 8, 'item3': 25, 'item4': 5, 'item5': 20}\nfiltered_by_value = {k: v for k, v in data.items() if v > 10}\nfiltered_by_key = {k: v for k, v in data.items() if k.startswith('item')}\nresult = {k: v for k, v in data.items() if v > 10 and k.startswith('item')}\ncount = len(result)\nprint(f\"Count: {count}\")",
        "answer": 3
    }
},
{
    "id": "SL-MIX-S001-V026",
    "metadata": {
        "name": "Generator Expression and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "generator_arithmetic",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    }
}
```

```
        },
        "task": {
            "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = sum(x**2 for x in range(1, 5) if x % 2 == 1)`?",
            "code": "numbers = list(range(1, 5))\nodd_numbers = [x for x in numbers if x % 2 == 1]\nsquared_odds = [x**2 for x in odd_numbers]\nresult = sum(x**2 for x in range(1, 5) if x % 2 == 1)\nprint(f\"Result: {result}\")",
            "answer": 10
        }
    },
    {
        "id": "SL-MIX-S001-V027",
        "metadata": {
            "name": "Exception Handling and Assignment Mix",
            "category": "Statement-Level",
            "subcategory": "Mixed",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0,
            "variant_type": "exception_assignment",
            "mixed_types": ["Assignment", "Boolean", "API Call"]
        },
        "task": {
            "description": "Given the following code snippet, what is the value of variable result after executing the try-except block when divisor is 0?",
            "code": "dividend = 10\n divisor = 0\n result = 0\n try:\n     result = dividend / divisor\n except ZeroDivisionError:\n     result = -1\n finally:\n     result += 1\nprint(f\"Result: {result}\")",
            "answer": 0
        }
    },
    {
        "id": "SL-MIX-S001-V028",
        "metadata": {
            "name": "Lambda Function and Arithmetic Mix",
            "category": "Statement-Level",
            "subcategory": "Mixed",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0,
            "variant_type": "lambda_arithmetic",
            "mixed_types": ["API Call", "Arithmetic", "Assignment"]
        },
        "task": {
            "description": "Given the following code snippet, what is the value of variable result after executing the statement `result = list(map(lambda x: x * 2 + 1, [1, 2, 3, 4]))[2]`?",
            "code": "result = list(map(lambda x: x * 2 + 1, [1, 2, 3, 4]))[2]",
            "answer": 5
        }
    }
]
```

```

        "code": "numbers = [1, 2, 3, 4]\ntransform = lambda x: x * 2 + 1\nmapped_values\n= list(map(transform, numbers))\nresult = list(map(lambda x: x * 2 + 1, [1, 2, 3, 4]))\n[2]\nprint(f\"Result: {result}\")",
        "answer": 7
    }
},
{
    "id": "SL-MIX-S001-V029",
    "metadata": {
        "name": "Class Property and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "class_property_arithmetic",
        "mixed_types": ["Assignment", "Arithmetic", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value of variable total after executing the statement `total = circle.radius * 2 * PI + circle.area()`?",
        "code": "import math\nPI = 3.14159\n\nclass Circle:\n    def __init__(self, radius):\n        self.radius = radius\n    def area(self):\n        return PI * self.radius ** 2\n\ncircle = Circle(3)\ncircumference = circle.radius * 2 * PI\narea_value = circle.area()\ntotal = circle.radius * 2 * PI + circle.area()\nprint(f\"Total: {total}\")",
        "answer": 47.12385
    }
},
{
    "id": "SL-MIX-S001-V030",
    "metadata": {
        "name": "Multiple Inheritance and Method Call Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "inheritance_method_call",
        "mixed_types": ["API Call", "Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet, what is the value returned by the method call `obj.calculate(5, 3)`?",
        "code": "class BaseA:\n    def calculate(self, x, y):\n        return x + y\n\nclass BaseB:\n    def calculate(self, x, y):\n        return x * y\n\nclass Combined(BaseA, BaseB):\n    def calculate(self, x, y):\n        return super().calculate(x, y) + 10\n\nobj = Combined()\nbase_result = BaseA.calculate(obj, 5, 3)\nfinal_result =\nobj.calculate(5, 3)\nprint(f\"Result: {final_result}\")",
        "answer": 18
    }
}

```

```
        },
    },
    {
        "id": "SL-MIX-S001-V031",
        "metadata": {
            "name": "Decorator and Function Call Mix",
            "category": "Statement-Level",
            "subcategory": "Mixed",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0,
            "variant_type": "decorator_function_call",
            "mixed_types": ["API Call", "Arithmetic", "Assignment"]
        },
        "task": {
            "description": "Given the following code snippet with decorator, what is the value returned by the function call `multiply_by_two(7)`?",
            "code": "def double_result(func):\n    def wrapper(*args, **kwargs):\n        result = func(*args, **kwargs)\n        return result * 2\n    return wrapper\n\n@double_result\ndef multiply_by_two(x):\n    return x * 2\n\noriginal_result = 7\n* 2\ndecorated_result = multiply_by_two(7)\nprint(f\"Result: {decorated_result}\")",
            "answer": 28
        }
    },
    {
        "id": "SL-MIX-S001-V032",
        "metadata": {
            "name": "Context Manager and Assignment Mix",
            "category": "Statement-Level",
            "subcategory": "Mixed",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0,
            "variant_type": "context_manager_assignment",
            "mixed_types": ["Assignment", "API Call", "Boolean"]
        },
        "task": {
            "description": "Given the following code snippet, what is the value of variable content_length after the with block when the file contains 'Hello World'?",
            "code": "import tempfile\nimport os\n\n# Create a temporary file with content\nwith tempfile.NamedTemporaryFile(mode='w', delete=False) as temp_file:\n    temp_file.write('Hello World')\n    temp_name = temp_file.name\n    content_length = 0\n\nwith open(temp_name, 'r') as file:\n    content = file.read()\n    content_length = len(content)\n\nos.unlink(temp_name)\nprint(f\"Content length: {content_length}\")",
            "answer": 11
        }
    },
    {
        "id": "SL-MIX-S001-V033",
        "metadata": {
            "name": "Context Manager and Assignment Mix",
            "category": "Statement-Level",
            "subcategory": "Mixed",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 0,
            "variant_type": "context_manager_assignment",
            "mixed_types": ["Assignment", "API Call", "Boolean"]
        },
        "task": {
            "description": "Given the following code snippet, what is the value of variable content_length after the with block when the file contains 'Hello World'?",
            "code": "import tempfile\nimport os\n\n# Create a temporary file with content\nwith tempfile.NamedTemporaryFile(mode='w', delete=False) as temp_file:\n    temp_file.write('Hello World')\n    temp_name = temp_file.name\n    content_length = 0\n\nwith open(temp_name, 'r') as file:\n    content = file.read()\n    content_length = len(content)\n\nos.unlink(temp_name)\nprint(f\"Content length: {content_length}\")",
            "answer": 11
        }
    }
]
```



```
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "threading_shared_variable",
        "mixed_types": ["Assignment", "Arithmetic", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet with thread-safe operations, what is the final value of counter.value after all operations?",
        "code": "import threading\n\nclass ThreadSafeCounter:\n    def __init__(self):\n        self.value = 0\n        self.lock = threading.Lock()\n\n    def increment(self, amount=1):\n        with self.lock:\n            self.value += amount\n\ncounter = ThreadSafeCounter()\ncounter.increment(5)\ncounter.increment(3)\ncounter.increment(-2)\nfinal_value = counter.value\nprint(f\"Final value: {final_value}\")",
        "answer": 6
    }
},
{
    "id": "SL-MIX-S001-V036",
    "metadata": {
        "name": "C Array Manipulation and Pointer Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "c_array_pointer_mix",
        "mixed_types": ["Assignment", "Arithmetic", "API Call"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the value stored at arr[2] after executing the statement `*(ptr + 2) = *(ptr + 1) + *(ptr + 3)`?",
        "code": "#include <stdio.h>\nint main() {\n    int arr[] = {10, 20, 30, 40, 50};\n    int *ptr = arr;\n    int sum = *(ptr + 1) + *(ptr + 3);\n    *(ptr + 2) = *(ptr + 1) + *(ptr + 3);\n    printf(\"arr[2] = %d\\n\", arr[2]);\n    return 0;\n}",
        "answer": 60
    }
},
{
    "id": "SL-MIX-S001-V037",
    "metadata": {
        "name": "C String Manipulation and Length Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0,
    }
}
```

```
        "variant_type": "c_string_manipulation",
        "mixed_types": ["API Call", "Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the value of variable result after executing the statement `result = strlen(dest) + strlen(src)`?",
        "code": "#include <stdio.h>\n#include <string.h>\nint main() {\n    char dest[20] = \\\"Hello\\\";\n    char src[] = \\\" World\\\";\n    size_t dest_len = strlen(dest);\n    size_t src_len = strlen(src);\n    strcat(dest, src);\n    int result = strlen(dest) +\n    strlen(src);\n    printf(\"Result: %d\\n\", result);\n    return 0;\n}",
        "answer": 17
    }
},
{
    "id": "SL-MIX-S001-V038",
    "metadata": {
        "name": "C File I/O and Error Handling Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "c_file_io_error",
        "mixed_types": ["API Call", "Boolean", "Assignment"]
    },
    "task": {
        "description": "Given the following C code snippet, what is the value of variable bytes_read when attempting to read from a valid file containing 'Test Data'?",
        "code": "#include <stdio.h>\n#include <string.h>\nint main() {\n    FILE *file = fopen(\"test.txt\", \"w\");\n    if (file) {\n        fprintf(file, \"Test Data\");\n        fclose(file);\n    }\n    file = fopen(\"test.txt\", \"r\");\n    char buffer[100];\n    size_t bytes_read = 0;\n    if (file != NULL) {\n        bytes_read = fread(buffer, 1,\n        sizeof(buffer), file);\n        fclose(file);\n    }\n    printf(\"Bytes read: %zu\\n\", bytes_read);\n    return 0;\n}",
        "answer": 9
    }
},
{
    "id": "SL-MIX-S001-V039",
    "metadata": {
        "name": "Async Programming and Arithmetic Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "async_arithmetic",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
}
```

```

"task": {
    "description": "Given the following code snippet with async function, what is the value returned by calling `asyncio.run(calculate_sum(10, 15))`?",
    "code": "import asyncio\n\nasync def calculate_sum(a, b):\n    await\n    asyncio.sleep(0) # simulate async operation\n    result = a + b\n    multiplied = result *\n    2\n    return multiplied\n\n# Direct calculation for comparison\n    direct_result = (10 + 15) *\n    2\nfinal_result = asyncio.run(calculate_sum(10, 15))\nprint(f\"Result: {final_result}\")",
    "answer": 50
},
{
    "id": "SL-MIX-S001-V040",
    "metadata": {
        "name": "Metaclass and Method Resolution Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "metaclass_method_resolution",
        "mixed_types": ["API Call", "Assignment", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet with metaclass, what is the value returned by calling `instance.compute(4)`?",
        "code": "class MetaClass(type):\n    def __new__(cls, name, bases, attrs):\n        if 'compute' in attrs:\n            original_compute = attrs['compute']\n            def enhanced_compute(self, x):\n                return original_compute(self, x) + 10\n            attrs['compute'] = enhanced_compute\n        return super().__new__(cls, name, bases, attrs)\n\nclass Calculator(metaclass=MetaClass):\n    def compute(self, x):\n        return x * 3\n\ninstance = Calculator()\noriginal_calculation = 4 * 3\nenhanced_result =\ninstance.compute(4)\nprint(f\"Result: {enhanced_result}\")",
        "answer": 22
    }
},
{
    "id": "SL-MIX-S001-V041",
    "metadata": {
        "name": "Numpy Array Operations Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "numpy_array_operations",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {

```

```
        "description": "Given the following code snippet using numpy operations, what is the sum of all elements in the result array after executing `result = arr1 * 2 + arr2`?",  
        "code": "# Simulating numpy-like operations without importing numpy\nclass SimpleArray:\n    def __init__(self, data):\n        self.data = data\n    def __mul__(self, scalar):\n        return SimpleArray([x * scalar for x in self.data])\n    def __add__(self, other):\n        return SimpleArray([a + b for a, b in zip(self.data, other.data)])\n    def sum(self):\n        return sum(self.data)\n\narr1 = SimpleArray([1, 2, 3, 4])\narr2 = SimpleArray([5, 6, 7, 8])\nmultiplied = arr1 * 2\nresult = arr1 * 2 + arr2\n\nprint(f\"Sum: {result.sum()}\")",  
        "answer": 36  
    },  
    {  
        "id": "SL-MIX-S001-V042",  
        "metadata": {  
            "name": "Database Query Simulation Mix",  
            "category": "Statement-Level",  
            "subcategory": "Mixed",  
            "type": "variant",  
            "source": "Generated",  
            "language": "python",  
            "difficulty": "hard",  
            "intervention": 0,  
            "variant_type": "database_query_simulation",  
            "mixed_types": ["API Call", "Boolean", "Arithmetic"]  
        },  
        "task": {  
            "description": "Given the following code snippet simulating database operations, what is the value of variable total_salary after executing the query simulation?",  
            "code": "# Simulating database operations\nemployees = [\n    {'name': 'Alice', 'dept': 'IT', 'salary': 50000, 'active': True},\n    {'name': 'Bob', 'dept': 'HR', 'salary': 45000, 'active': True},\n    {'name': 'Charlie', 'dept': 'IT', 'salary': 55000, 'active': False},\n    {'name': 'Diana', 'dept': 'IT', 'salary': 60000, 'active': True}\n]\n\n# Query:  
SELECT SUM(salary) FROM employees WHERE dept='IT' AND active=True\n\nfiltered_employees = [emp for emp in employees if emp['dept'] == 'IT' and emp['active']]  
salaries = [emp['salary'] for emp in filtered_employees]\n\nprint(f\"Total salary: {sum(salaries)}\")",  
            "answer": 110000  
        },  
        {  
            "id": "SL-MIX-S001-V043",  
            "metadata": {  
                "name": "Graph Algorithm Simulation Mix",  
                "category": "Statement-Level",  
                "subcategory": "Mixed",  
                "type": "variant",  
                "source": "Generated",  
                "language": "python",  
                "difficulty": "hard",  
                "intervention": 0,  
                "variant_type": "graph_algorithm_simulation",  
                "mixed_types": ["API Call", "Arithmetic", "Boolean"]  
            }  
        }  
    }  
}
```

```

    },
    "task": {
        "description": "Given the following code snippet implementing a simple graph traversal, what is the value of variable path_length after finding the shortest path?",
        "code": "# Simple graph representation and path finding\ngraph = {\n    'A': ['B', 'C'],\n    'B': ['A', 'D', 'E'],\n    'C': ['A', 'F'],\n    'D': ['B'],\n    'E': ['B', 'F'],\n    'F': ['C', 'E']\n}\ndef find_shortest_path(graph, start, end, path=[]):\n    path = path + [start]\n    if start == end:\n        return path\n    if start not in graph:\n        return None\n    shortest = None\n    for node in graph[start]:\n        if node not in path:\n            newpath = find_shortest_path(graph, node, end, path)\n            if newpath:\n                if not shortest or len(newpath) < len(shortest):\n                    shortest = newpath\n    return shortest\nshortest_path = find_shortest_path(graph, 'A', 'F')\npath_length = len(shortest_path) if shortest_path else 0\nprint(f\"Path length: {path_length}\")",
        "answer": 3
    }
},
{
    "id": "SL-MIX-S001-V044",
    "metadata": {
        "name": "Memory Pool and Allocation Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "memory_pool_allocation",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following C code snippet with custom memory management, what is the value of variable allocated_blocks after all allocation operations?",
        "code": "#include <stdio.h>\n#include <stdlib.h>\n\ntypedef struct {\n    void *memory;\n    size_t size;\n    int allocated_blocks;\n} MemoryPool;\n\nMemoryPool* create_pool(size_t total_size) {\n    MemoryPool *pool = malloc(sizeof(MemoryPool));\n    pool->memory = malloc(total_size);\n    pool->size = total_size;\n    pool->allocated_blocks = 0;\n    return pool;\n}\n\nvoid* pool_alloc(MemoryPool *pool, size_t size) {\n    if (pool && pool->memory) {\n        pool->allocated_blocks++;\n        return pool->memory; // Simplified allocation\n    }\n    return NULL;\n}\n\nint main() {\n    MemoryPool *pool = create_pool(1024);\n    void *ptr1 = pool_alloc(pool, 64);\n    void *ptr2 = pool_alloc(pool, 128);\n    void *ptr3 = pool_alloc(pool, 32);\n\n    int allocated_blocks = pool->allocated_blocks;\n    printf(\"Allocated blocks: %d\\n\", allocated_blocks);\n\n    free(pool->memory);\n    free(pool);\n    return allocated_blocks;\n}",
        "answer": 3
    }
},
{
    "id": "SL-MIX-S001-V045",
    "metadata": {

```

```

        "name": "Binary Tree Traversal Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "binary_tree_traversal",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet with binary tree operations, what is the value of variable node_count after traversing the tree?",
        "code": "class TreeNode:\n    def __init__(self, val=0, left=None, right=None):\n        self.val = val\n        self.left = left\n        self.right = right\n    def count_nodes(root):\n        if not root:\n            return 0\n        return 1 + count_nodes(root.left) + count_nodes(root.right)\n\n# Create binary tree: 1-2-4, 1-3-5-6\nroot = TreeNode(1)\nroot.left = TreeNode(2)\nroot.right = TreeNode(3)\nroot.left.left = TreeNode(4)\nroot.right.left = TreeNode(5)\nroot.right.right = TreeNode(6)\nnode_count = count_nodes(root)\nprint(f\"Node count: {node_count}\")",
        "answer": 6
    }
},
{
    "id": "SL-MIX-S001-V046",
    "metadata": {
        "name": "Recursive Fibonacci with Memoization Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "recursive_fibonacci_memo",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet with memoized fibonacci, what is the value returned by calling `fibonacci(7)`?",
        "code": "def fibonacci(n, memo={}):\n    if n in memo:\n        return memo[n]\n    if n <= 1:\n        return n\n    memo[n] = fibonacci(n-1, memo) + fibonacci(n-2, memo)\n    return memo[n]\n\n# Calculate some values\nfib_5 = fibonacci(5)\nfib_6 = fibonacci(6)\nfib_7 = fibonacci(7)\nprint(f\"Fibonacci(7): {fib_7}\")",
        "answer": 13
    }
},
{
    "id": "SL-MIX-S001-V047",
    "metadata": {
        "name": "Closure Variable Capture Mix",
        "category": "Statement-Level",

```

```
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "closure_variable_capture",
        "mixed_types": ["Assignment", "Arithmetic", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet with closure, what is the value returned by calling `multiplier(4)`?",
        "code": "def create_multiplier(factor):\n    base_value = 10\n    def\n    multiplier(x):\n        nonlocal base_value\n        base_value += 1\n        return x *\n    factor + base_value\n    return multiplier\n\nmult_func = create_multiplier(3)\nresult1 =\nmult_func(2) # base_value becomes 11, returns 2*3+11=17\nresult2 = mult_func(4) #\nbase_value becomes 12, returns 4*3+12=24\nprint(f\"Result: {result2}\")",
        "answer": 24
    }
},
{
    "id": "SL-MIX-S001-V048",
    "metadata": {
        "name": "Generator Function with State Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "generator_function_state",
        "mixed_types": ["API Call", "Arithmetic", "Assignment"]
    },
    "task": {
        "description": "Given the following code snippet with generator, what is the third value yielded by the generator `sequence_gen()`?",
        "code": "def sequence_gen():\n    value = 1\n    while value <= 10:\n        yield value * 2\n        value += 2\n    gen = sequence_gen()\n    first = next(gen) # value=1, yields 1*2=2, value becomes 3\n    second = next(gen) # value=3, yields 3*2=6, value becomes 5\n    third = next(gen) # value=5, yields 5*2=10, value becomes 7\n    print(f\"Third value: {third}\")",
        "answer": 10
    }
},
{
    "id": "SL-MIX-S001-V049",
    "metadata": {
        "name": "Property Descriptor and Validation Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
```

```

        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "property_descriptor_validation",
        "mixed_types": ["Assignment", "Boolean", "Arithmetic"]
    },
    "task": {
        "description": "Given the following code snippet with property descriptor, what is the value of `obj.temperature` after setting it to 150?",
        "code": "class TemperatureDescriptor:\n    def __get__(self, obj,\nobjtype=None):\n        return obj._temperature\n    def __set__(self, obj, value):\n        if value < 0:\n            obj._temperature = 0\n        elif value > 100:\n            obj._temperature = 100\n        else:\n            obj._temperature = value\n\nclass Thermostat:\n    temperature = TemperatureDescriptor()\n    \n    def __init__(self):\n        self._temperature = 20\n\nobj = Thermostat()\nobj.temperature = 75  # Sets to 75\nobj.temperature = 150  # Clamps to 100\nfinal_temp = obj.temperature\nprint(f\"Final temperature: {final_temp}\")",
        "answer": 100
    }
},
{
    "id": "SL-MIX-S001-V050",
    "metadata": {
        "name": "Complex Number Operations Mix",
        "category": "Statement-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 0,
        "variant_type": "complex_number_operations",
        "mixed_types": ["Arithmetic", "Assignment", "API Call"]
    },
    "task": {
        "description": "Given the following code snippet with complex number operations, what is the real part of variable result after executing `result = (z1 * z2) + z3.conjugate()`?",
        "code": "z1 = complex(3, 4)      # 3 + 4j\nz2 = complex(2, -1)      # 2 - 1j\nz3 = complex(1, 2)      # 1 + 2j\n\n# z1 * z2 = (3+4j) * (2-1j) = 6-3j+8j-4j² = 6+5j+4 = 10+5j\nproduct = z1 * z2\n# z3.conjugate() = 1 - 2j\nconjugate = z3.conjugate()\n# result = (10+5j) + (1-2j) = 11+3j\nresult = (z1 * z2) + z3.conjugate()\nreal_part = result.real\nprint(f\"Real part: {real_part}\")",
        "answer": 11.0
    }
}
]

```

2 - 代码块级推理 [Block-Level] (326)

2A - 线性代码块 [Linear] (72)

```
# 线性代码块推理变式生成提示词
```

任务目标

基于给定的线性代码块种子任务，生成多样化的变式任务来全面测试大模型对顺序执行代码块的理解能力，包括变量状态跟踪、数据流分析、执行顺序推理和最终状态预测。

线性代码块特征分析

线性代码块是指按顺序执行、无分支控制结构的代码段，重点测试模型对语句执行顺序、变量状态变化和数据依赖关系的理解。

变式生成维度

1. 执行模式变式

- **顺序执行变式**：严格按语句顺序执行的代码块
- **依赖链执行变式**：后续语句依赖前面语句结果的执行模式
- **并行无关变式**：语句间无依赖关系的独立执行
- **累积执行变式**：逐步累积计算结果的执行模式
- **转换管道变式**：数据经过多步转换的管道式执行
- **初始化序列变式**：对象或结构体的初始化执行序列

2. 数据流模式变式

- **单向数据流变式**：数据从输入到输出的单向流动
- **链式数据流变式**：数据在变量间的链式传递
- **累积数据流变式**：数据逐步累积和聚合
- **变换数据流变式**：数据类型或格式的逐步变换
- **分离数据流变式**：一个数据源分离成多个变量
- **合并数据流变式**：多个数据源合并成一个结果

3. 变量生命周期变式

- **变量初始化变式**：变量的初始赋值和设置
- **变量更新变式**：变量值的逐步更新和修改
- **变量累积变式**：变量值的累积计算
- **变量转换变式**：变量类型或格式的转换
- **变量传递变式**：变量值在不同变量间的传递
- **变量销毁变式**：变量生命周期的结束和资源释放

4. 计算复杂度变式

- **简单计算变式**：基础算术运算的线性计算
- **复合计算变式**：多步骤的复合数学计算
- **字符串处理变式**：字符串操作的线性处理
- **数组处理变式**：数组元素的线性处理（非循环）
- **结构体操作变式**：结构体字段的逐一设置
- **指针操作变式**：指针和地址的线性操作

5. 内存管理变式

- **内存分配变式**：`malloc/calloc`等内存分配操作
- **内存初始化变式**：分配内存的初始化设置
- **指针赋值变式**：指针变量的赋值和重新指向
- **内存拷贝变式**：`memcpy/strcpy`等内存拷贝操作

- **内存释放变式**：`free`等内存释放操作
- **内存状态跟踪变式**：内存状态在代码块中的变化

6. 函数调用变式

- **单函数调用变式**：单个函数调用及其结果处理
- **连续函数调用变式**：多个函数的连续调用
- **嵌套函数调用变式**：函数调用结果作为参数的嵌套调用
- **副作用函数变式**：有副作用的函数调用对变量的影响
- **错误处理变式**：函数返回错误时的处理逻辑
- **资源管理变式**：函数调用涉及资源申请和释放

7. 类型和转换变式

- **隐式类型转换变式**：自动类型转换对计算的影响
- **显式类型转换变式**：强制类型转换的使用
- **精度变化变式**：类型转换导致的精度损失
- **符号转换变式**：有符号和无符号类型的转换
- **指针类型转换变式**：不同指针类型间的转换
- **结构体转换变式**：结构体和基础类型的转换

8. 错误和异常变式

- **空指针检查变式**：`NULL`指针的检查和处理
- **边界条件变式**：数组越界、缓冲区溢出等边界情况
- **资源不足变式**：内存不足等资源限制情况
- **参数验证变式**：输入参数的有效性检查
- **状态一致性变式**：对象状态的一致性维护
- **错误传播变式**：错误在代码块中的传播

9. 平台相关变式

- **字节序变式**：大小端字节序对数据的影响
- **对齐变式**：内存对齐对结构体布局的影响
- **指针大小变式**：32位/64位平台的指针大小差异
- **类型大小变式**：不同平台上数据类型大小的差异
- **编译器优化变式**：编译器优化对代码执行的影响
- **标准库差异变式**：不同平台标准库实现的差异

10. 语言特性变式

- **Python特性变式**：动态类型、多重赋值、解包等特性
- **C语言特性变式**：指针、结构体、预处理器等特性
- **Java特性变式**：对象初始化、引用传递等特性
- **JavaScript特性变式**：动态类型、对象属性等特性
- **跨语言移植变式**：相同逻辑在不同语言中的实现差异

复杂度层次设计

简单线性 (Easy)

- 3-5条语句的简单顺序执行
- 基础数据类型的简单操作
- 明确的数据流向，无复杂依赖
- 直观的计算过程，易于手工跟踪

中等线性 (Medium)

- 6-10条语句的中等复杂度执行
- 包含函数调用或类型转换

- 中等复杂度的数据依赖关系
- 需要仔细跟踪变量状态变化

复杂线性 (Hard)

- 10条以上语句的复杂执行序列
- 涉及内存管理、指针操作或复杂计算
- 多层次的数据依赖和变换
- 需要深入理解语言语义和执行模型

生成策略

种子分析策略

1. **识别执行模式**: 分析种子的执行流程特征
2. **提取数据流**: 识别变量间的数据依赖关系
3. **分析复杂点**: 找出容易产生理解偏差的关键点
4. **确定变式方向**: 选择合适的变式生成维度

变式设计原则

1. **保持线性特征**: 确保所有变式都是线性执行, 无分支
2. **渐进复杂化**: 从简单变式逐步增加到复杂变式
3. **现实场景**: 优先设计实际编程中常见的线性代码模式
4. **测试覆盖**: 确保每个变式都测试特定的推理能力

质量保证

1. **执行顺序验证**: 确保语句执行顺序的推理正确
2. **状态跟踪验证**: 验证变量状态变化的准确性
3. **数据流验证**: 确保数据依赖关系的正确理解
4. **边界情况验证**: 包含各种边界和异常情况

输出格式要求

```
```json
[
 {
 "id": "BL-LN-S00X-V001",
 "metadata": {
 "name": "线性代码块变式名称",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "目标语言",
 "difficulty": "easy/medium/hard",
 "intervention": 0,
 "variant_type": "变式类型标签",
 "execution_pattern": "sequential/dependent/accumulative/transform"
 },
 "task": {
 "description": "线性代码块推理任务描述",
 "code": "线性执行的代码块",
 "answer": "最终状态或计算结果"
 }
 },
]
```

{下一个变式...}

## 特殊字段说明

`execution_pattern`: 标识代码块的执行模式特征

`variant_type`: 描述具体的变式类型

## 生成目标

为每个提供的种子任务生成10-15个线性代码块变式，确保：

每种难度等级至少包含3-5个变式

涵盖至少6-8种不同的执行模式

包含2-3个跨语言的变式

至少包含3-4个涉及内存管理或复杂数据流的变式

请基于此提示词，为给定的线性代码块种子任务生成全面的变式集合，输出格式为包含所有变式的JSON数组。

```
"task": {
 "description": "Given the following Python buffer initialization, what is the final value of buf['length']?",
 "code": "def init_dynamic_buffer(buf):\n buf['type'] = 'DYNAMIC'\n buf['capacity'] = 128\n buf['data'] = [None] * buf['capacity']\n buf['length'] = 0",
 "answer": 0
},
{
 "id": "BL-LN-S001-V003",
 "metadata": {
 "name": "错误处理扩展变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "error_handling",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following extended initialization with error checking, what is the final value of buf->len when malloc succeeds?",
 "code": "int k5_buf_init_extended(struct k5buf *buf) {\n buf->buftype = K5BUF_DYNAMIC;\n buf->space = 256;\n buf->data = malloc(buf->space);\n if (buf->data == NULL) {\n buf->len = -1;\n return -1;\n }\n memset(buf->data, 0, buf->space);\n buf->len = 0;\n return 0;\n}",
 "answer": 0
},
{
 "id": "BL-LN-S001-V004",
 "metadata": {
 "name": "累积初始化变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "accumulative_init",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following cumulative buffer setup, what is the final value of buf->total_allocated?",
 "code": "void setup_buffer_pool(struct buffer_pool *buf) {\n buf->block_size = 32;\n buf->num_blocks = 4;\n buf->total_allocated = buf->block_size * buf->num_blocks;\n buf->used_blocks = 0;\n buf->free_blocks = buf->num_blocks;\n}",
 "answer": 128
}
```

```
 }
 },
 {
 "id": "BL-LN-S001-V005",
 "metadata": {
 "name": "指针操作链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "pointer_operations",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following pointer manipulation block, what is the final value pointed by end_ptr?",
 "code": "void setup_buffer_ptrs(struct k5buf *buf) {\n buf->data =\n malloc(64);\n buf->start_ptr = buf->data;\n buf->current_ptr = buf->start_ptr;\n buf->end_ptr = buf->start_ptr + 64;\n buf->len = buf->end_ptr - buf->start_ptr;\n}",
 "answer": 64
 }
 },
 {
 "id": "BL-LN-S001-V006",
 "metadata": {
 "name": "Java对象初始化变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "java",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following Java buffer initialization, what is the final value of buffer.length?",
 "code": "public void initDynamicBuffer(DynamicBuffer buffer) {\n buffer.bufferType = BufferType.DYNAMIC;\n buffer.capacity = 128;\n buffer.data = new byte[buffer.capacity];\n buffer.length = 0;\n buffer.growthFactor = 2.0;\n}",
 "answer": 0
 }
 },
 {
 "id": "BL-LN-S001-V007",
 "metadata": {
 "name": "内存对齐变式",
 "category": "Block-Level",
 "subcategory": "Linear"
 }
 }
]
```

```
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "memory_alignment",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following aligned memory allocation, what is the final value of buf->aligned_size?",
 "code": "void init_aligned_buffer(struct aligned_buf *buf) {\n buf->requested_size = 100;\n buf->alignment = 16;\n buf->aligned_size = (buf->requested_size + buf->alignment - 1) & ~buf->alignment;\n buf->data = aligned_alloc(buf->alignment, buf->aligned_size);\n buf->len = 0;\n}",
 "answer": 112
 }
},
{
 "id": "BL-LN-S001-v008",
 "metadata": {
 "name": "缓冲区状态机变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "state_machine",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following buffer state initialization, what is the final value of buf->state?",
 "code": "void init_buffer_state(struct state_buf *buf) {\n buf->state = BUF_STATE_UNINITIALIZED;\n buf->data = malloc(256);\n buf->state = (buf->data != NULL) ? BUF_STATE_ALLOCATED : BUF_STATE_ERROR;\n buf->len = 0;\n buf->state = BUF_STATE_READY;\n}",
 "answer": "BUF_STATE_READY"
 }
},
{
 "id": "BL-LN-S001-v009",
 "metadata": {
 "name": "位操作缓冲区变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 3
 }
}
```

```
 "intervention": 2,
 "variant_type": "bitwise_operations",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following bit manipulation buffer setup, what is the final value of buf->flags?",
 "code": "void init_bit_buffer(struct bit_buf *buf) {\n buf->flags = 0;\nbuf->flags |= FLAG_DYNAMIC;\n buf->flags |= FLAG_GROWABLE;\n buf->flags &= ~FLAG_READONLY;\n buf->len = 0;\n buf->capacity = 128;\n}",
 "answer": "FLAG_DYNAMIC | FLAG_GROWABLE"
 }
},
{
 "id": "BL-LN-S001-V010",
 "metadata": {
 "name": "JavaScript对象变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following JavaScript buffer initialization, what is the final value of buffer.length?",
 "code": "function initDynamicBuffer(buffer) {\n buffer.type = 'DYNAMIC';\nbuffer.capacity = 128;\n buffer.data = new ArrayBuffer(buffer.capacity);\nbuffer.length = 0;\n buffer.view = new Uint8Array(buffer.data);\n}",
 "answer": 0
 }
},
{
 "id": "BL-LN-S001-V011",
 "metadata": {
 "name": "多重指针变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "multiple_pointers",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following multiple pointer setup, what is the final value of **double_ptr?",
```



```
{
 "id": "BL-LN-S002-V002",
 "metadata": {
 "name": "字符串处理变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "string_processing",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following string configuration block, what is the final length of service_name?",
 "code": "service_prefix = 'web'\nservice_id = '001'\nservice_suffix = 'prod'\nservice_name = f\"{service_prefix}-{service_id}-{service_suffix}\"\\nport = 8080",
 "answer": 12
 },
},
{
 "id": "BL-LN-S002-V003",
 "metadata": {
 "name": "C语言配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following C configuration block, what is the final value of config->retry_count?",
 "code": "void init_config(struct config *config) {\\n config->retry_count = 3;\\n config->timeout_ms = 5000;\\n config->debug_mode = 0;\\n config->buffer_size = 1024;\\n}",
 "answer": 3
 },
},
{
 "id": "BL-LN-S002-V004",
 "metadata": {
 "name": "数值计算链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following C configuration block, what is the final value of config->retry_count?",
 "code": "void init_config(struct config *config) {\\n config->retry_count = 3;\\n config->timeout_ms = 5000;\\n config->debug_mode = 0;\\n config->buffer_size = 1024;\\n}",
 "answer": 3
 },
},
}
```

```
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "numerical_chain",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following numerical calculation chain, what is the final value of performance_score?",
 "code": "base_score = 100\nlatency_penalty = 15\nthroughput_bonus = 25\nerror_penalty = 5\nperformance_score = base_score - latency_penalty + throughput_bonus - error_penalty",
 "answer": 105
 }
},
{
 "id": "BL-LN-S002-V005",
 "metadata": {
 "name": "布尔逻辑变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "boolean_logic",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following boolean configuration, what is the final value of system_ready?",
 "code": "auth_enabled = True\nlogging_enabled = True\ncache_enabled = False\ndebug_mode = False\nsystem_ready = auth_enabled and logging_enabled",
 "answer": true
 }
},
{
 "id": "BL-LN-S002-V006",
 "metadata": {
 "name": "Java配置对象变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "java",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
```

```
 "description": "Given the following Java configuration setup, what is the final value of config.maxRetries?",
 "code": "public void setupConfiguration(Configuration config) {\nconfig.maxRetries = 3;\n config.timeoutSeconds = 30;\n config.logLevel =\nLogLevel.INFO;\n config.enableDebug = false;\n config.encoding = \"UTF-8\";\n}\n",
 "answer": 3
 },
 {
 "id": "BL-LN-S002-V007",
 "metadata": {
 "name": "环境变量处理变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "environment_processing",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following environment variable processing, what is the final value of worker_count?",
 "code": "default_workers = 4\nenv_workers = '8'\nmax_workers = 16\nworker_count = min(int(env_workers), max_workers)\nthread_pool_size = worker_count * 2",
 "answer": 8
 },
 },
 {
 "id": "BL-LN-S002-V008",
 "metadata": {
 "name": "列表初始化变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "list_initialization",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following list initialization, what is the final length of supported_formats?",
 "code": "image_formats = ['jpg', 'png', 'gif']\nvideo_formats = ['mp4',\n'avi']\nunsupported_formats = image_formats + video_formats\nmax_file_size =\n10485760\nallow_upload = True",
 "answer": 5
 },
 },
```

```
{
 "id": "BL-LN-S002-V009",
 "metadata": {
 "name": "字典配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "dictionary_configuration",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following dictionary configuration, what is the final value of config['total_memory']?",
 "code": "config = {}\\nconfig['cpu_cores'] = 4\\nconfig['memory_per_core'] = 2048\\nconfig['total_memory'] = config['cpu_cores'] * config['memory_per_core']\\nconfig['disk_space'] = 100000",
 "answer": 8192
 }
,
{
 "id": "BL-LN-S002-V010",
 "metadata": {
 "name": "JavaScript配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following JavaScript configuration, what is the final value of config.maxRetries?",
 "code": "const config = {}\\nconfig.maxRetries = 3;\\nconfig.timeoutMs = 5000;\\nconfig.debugMode = false;\\nconfig.apiVersion = 'v2';",
 "answer": 3
 }
,
{
 "id": "BL-LN-S002-V011",
 "metadata": {
 "name": "类型转换配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 }
}
```

```
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "type_conversion",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following type conversion configuration, what is the final type of port_number?",
 "code": "port_str = '8080'\nport_number = int(port_str)\nhost_name = 'localhost'\nssl_enabled = 'true'\nssl_bool = ssl_enabled.lower() == 'true'",
 "answer": "int"
 }
},
{
 "id": "BL-LN-S002-V012",
 "metadata": {
 "name": "C结构体配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "struct_configuration",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following C struct configuration, what is the final value of settings->flags?",
 "code": "void configure_system(struct system_settings *settings) {\n settings->max_connections = 1000;\n settings->timeout_seconds = 30;\n settings->log_level = 2;\n settings->flags = 0x01 | 0x04 | 0x10;\n settings->buffer_size = 4096;\n}",
 "answer": 21
 }
},
{
 "id": "BL-LN-S003-V001",
 "metadata": {
 "name": "简化函数调用变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "function_call_chain",
 "execution_pattern": "dependent"
 },
 "task": {
```

```
 "description": "Given the following simplified file processing, what is the return value when file_open succeeds?",
 "code": "int simple_file_process(const char *filename) {\n FILE *fp = fopen(filename, \"r\");\n if (fp == NULL) return -1;\n int result = process_content(fp);\n fclose(fp);\n return result == 0 ? 0 : -1;\n }",
 "answer": 0
 },
 {
 "id": "BL-LN-S003-V002",
 "metadata": {
 "name": "Python异常处理变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "exception_handling",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following Python file processing, what is the return value when file processing succeeds?",
 "code": "def process_data_file(filename):\n result = -1\n try:\n with open(filename, 'r') as f:\n data = f.read()\n parsed =\n parse_data(data)\n result = 0 if validate_data(parsed) else -1\n except\n IOError:\n result = -1\n return result",
 "answer": 0
 },
 },
 {
 "id": "BL-LN-S003-V003",
 "metadata": {
 "name": "资源管理链式变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "resource_management",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following resource management chain, what is the final value of ctx->status when all operations succeed?",
 }
 }
}
```

```
 "code": "int init_context(struct context *ctx, const char *config_file) {\n ctx->status = STATUS_INIT;\n ctx->config = load_config(config_file);\n if (ctx->config == NULL) {\n ctx->status = STATUS_ERROR;\n return -1;\n }\n ctx->resources = allocate_resources(ctx->config);\n ctx->status = (ctx->resources != NULL) ?\n STATUS_READY : STATUS_ERROR;\n return ctx->status == STATUS_READY ? 0 : -1;\n}\n\n \"answer\": \"STATUS_READY\"\n }\n},\n{\n \"id\": \"BL-LN-S003-V004\",\n \"metadata\": {\n \"name\": \"数据库连接变式\",\n \"category\": \"Block-Level\",\n \"subcategory\": \"Linear\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"hard\",\n \"intervention\": 2,\n \"variant_type\": \"database_connection\",\n \"execution_pattern\": \"dependent\"\n },\n \"task\": {\n \"description\": \"Given the following database connection setup, what is the\n return value when connection succeeds?\",\n \"code\": \"int connect_database(struct db_context *ctx, const char *conn_string)\n{\n int result = -1;\n ctx->connection = db_connect(conn_string);\n if (ctx->connection != NULL) {\n ctx->prepared_stmt = db_prepare(ctx->connection, \"SELECT\n 1\");\n result = (ctx->prepared_stmt != NULL) ? 0 : -1;\n }\n return\n result;\n}\n\n \"answer\": 0\n }\n},\n{\n \"id\": \"BL-LN-S003-V005\",\n \"metadata\": {\n \"name\": \"网络套接字变式\",\n \"category\": \"Block-Level\",\n \"subcategory\": \"Linear\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"hard\",\n \"intervention\": 2,\n \"variant_type\": \"network_socket\",\n \"execution_pattern\": \"dependent\"\n },\n \"task\": {\n \"description\": \"Given the following socket setup, what is the return value when\n socket creation and binding succeed?\",
```

```
 "code": "int setup_server_socket(int port) {\n int sockfd = socket(AF_INET,\nSOCK_STREAM, 0);\n if (sockfd < 0) return -1;\n struct sockaddr_in addr;\n addr.sin_family = AF_INET;\n addr.sin_port = htons(port);\n addr.sin_addr.s_addr =\nINADDR_ANY;\n int bind_result = bind(sockfd, (struct sockaddr*)&addr, sizeof(addr));\n return bind_result == 0 ? sockfd : -1;\n }",\n "answer": "sockfd"\n },\n {\n "id": "BL-LN-S003-V006",\n "metadata": {\n "name": "Java资源管理变式",\n "category": "Block-Level",\n "subcategory": "Linear",\n "type": "variant",\n "source": "Generated",\n "language": "java",\n "difficulty": "medium",\n "intervention": 1,\n "variant_type": "cross_language",\n "execution_pattern": "dependent"\n },\n "task": {\n "description": "Given the following Java resource management, what is the return value when file processing succeeds?",\n "code": "public int processFile(String filename) {\n int result = -1;\n try (FileInputStream fis = new FileInputStream(filename)) {\n byte[] buffer = new byte[1024];\n int bytesRead = fis.read(buffer);\n result = (bytesRead > 0) ? 0 : -1;\n } catch (IOException e) {\n result = -1;\n }\n return result;\n }",\n "answer": 0\n },\n {\n "id": "BL-LN-S003-V007",\n "metadata": {\n "name": "内存映射文件变式",\n "category": "Block-Level",\n "subcategory": "Linear",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "memory_mapping",\n "execution_pattern": "dependent"\n },\n "task": {\n "description": "Given the following memory mapping setup, what is the return value when mapping succeeds?",\n "code": "int map_file_to_memory(const char *filename, struct file_map *map) {\n int fd = open(filename, O_RDONLY);\n if (fd < 0) return -1;\n map->size = lseek(fd, 0, SEEK_END);\n map->data = mmap(NULL, map->size, PROT_READ, MAP_PRIVATE, fd, 0);\n close(fd);\n return (map->data != MAP_FAILED) ? 0 : -1;\n }",\n "answer": 0\n }\n }\n }\n}
```

```
 "answer": 0
 }
},
{
 "id": "BL-LN-S003-V008",
 "metadata": {
 "name": "动态库加载变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "dynamic_loading",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following dynamic library loading, what is the return value when loading and symbol resolution succeed?",
 "code": "int load_plugin(const char *lib_path, struct plugin *p) {\n p->handle = dlopen(lib_path, RTLD_LAZY);\n if (!p->handle) return -1;\n p->init_func = \n dlsym(p->handle, \"plugin_init\");\n if (!p->init_func) {\n dlclose(p->handle);\n return -1;\n }\n return p->init_func() == 0 ? 0 : -1;\n}",
 "answer": 0
 }
},
{
 "id": "BL-LN-S003-V009",
 "metadata": {
 "name": "线程同步变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "thread_synchronization",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following thread synchronization setup, what is the return value when mutex initialization succeeds?",
 "code": "int init_thread_sync(struct sync_context *ctx) {\n int result =\n pthread_mutex_init(&ctx->mutex, NULL);\n if (result != 0) return -1;\n result =\n pthread_cond_init(&ctx->condition, NULL);\n if (result != 0) {\n pthread_mutex_destroy(&ctx->mutex);\n return -1;\n }\n ctx->initialized = 1;\n return 0;\n}",
 "answer": 0
 }
},
{

```

```
"id": "BL-LN-S003-V010",
"metadata": {
 "name": "JavaScript Promise变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "dependent"
},
"task": {
 "description": "Given the following JavaScript async setup, what is the return value when file reading succeeds?",
 "code": "async function processFile(filename) {\n let result = -1;\n try {\n const fs = require('fs').promises;\n const data = await fs.readFile(filename, 'utf8');\n const parsed = JSON.parse(data);\n result = parsed.valid ? 0 : -1;\n } catch (error) {\n result = -1;\n }\n return result;\n}",
 "answer": 0
},
{
 "id": "BL-LN-S003-V011",
 "metadata": {
 "name": "信号处理变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "signal_handling",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following signal handler setup, what is the return value when signal registration succeeds?",
 "code": "int setup_signal_handlers(struct signal_context *ctx) {\n struct sigaction sa;\n sa.sa_handler = handle_signal;\n sigemptyset(&sa.sa_mask);\n sa.sa_flags = 0;\n int result = sigaction(SIGTERM, &sa, &ctx->old_term);\n if (result != 0) return -1;\n result = sigaction(SIGINT, &sa, &ctx->old_int);\n return result == 0 ? 0 : -1;\n}",
 "answer": 0
 }
},
{
 "id": "BL-LN-S003-V012",
 "metadata": {
 "name": "Python上下文管理器变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "dependent"
 }
}
```

```
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "context_manager",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following Python context manager usage, what is the return value when database operations succeed?",
 "code": "def process_database_transaction(query):\n result = -1\n with database_connection() as conn:\n cursor = conn.cursor()\n cursor.execute(query)\n rows = cursor.fetchall()\n result = 0 if len(rows) > 0\n else -1\n conn.commit()\n return result",
 "answer": 0
 }
},
{
 "id": "BL-LN-S004-V001",
 "metadata": {
 "name": "数学计算链扩展变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "mathematical_chain",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following extended calculation chain, what is the final value of final_result?",
 "code": "initial = 5\nstep1 = initial * 2\nstep2 = step1 + 3\nstep3 = step2 ** 2\nstep4 = step3 // 4\nfinal_result = step4 - 1",
 "answer": 41
 }
},
{
 "id": "BL-LN-S004-V002",
 "metadata": {
 "name": "类型转换链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
```

```
 "variant_type": "type_conversion",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following type conversion chain, what is the final value of result?",
 "code": "double input = 3.7;\nint step1 = (int)input;\nfloat step2 = step1 *\n2.5f;\nlong step3 = (long)step2;\nunsigned int result = (unsigned int)step3;",
 "answer": 7
 }
},
{
 "id": "BL-LN-S004-V003",
 "metadata": {
 "name": "字符串变换链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "string_transformation",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following string transformation chain, what is the final length of processed_text?",
 "code": "original = \" Hello World \ntrimmed = original.strip()\nlowercased = trimmed.lower()\nreplaced = lowercased.replace(\" \", \"_\")\nprocessed_text = replaced.upper()",

 "answer": 11
 }
},
{
 "id": "BL-LN-S004-V004",
 "metadata": {
 "name": "科学计算链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "scientific_computation",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following scientific calculation chain, what is the final value of normalized_result?",
```



```
{
 "id": "BL-LN-S004-V007",
 "metadata": {
 "name": "JavaScript数值链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following JavaScript number chain, what is the final value of result?",
 "code": "let base = 100;\nlet percentage = 0.15;\nlet increase = base * percentage;\nlet intermediate = base + increase;\nlet discount = intermediate * 0.1;\nlet result = Math.round(intermediate - discount);",
 "answer": 104
 },
},
{
 "id": "BL-LN-S004-V008",
 "metadata": {
 "name": "数组索引计算变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "array_indexing",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following array indexing calculation, what is the final value of selected_value?",
 "code": "data = [10, 20, 30, 40, 50]\nindex_base = 2\noffset = 1\nselected_index = (index_base + offset) % len(data)\nselected_value = data[calculated_index]",
 "answer": 40
 },
},
{
 "id": "BL-LN-S004-V009",
 "metadata": {
 "name": "温度转换链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 },
}
```

```
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "unit_conversion",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following temperature conversion chain, what is the final value of kelvin?",
 "code": "celsius = 25\nfahrenheit = celsius * 9/5 + 32\nback_to_celsius = (fahrenheit - 32) * 5/9\nkelvin = back_to_celsius + 273.15\nrounded_kelvin = round(kelvin, 1)",
 "answer": 298.1
 }
},
{
 "id": "BL-LN-S004-V010",
 "metadata": {
 "name": "复数计算链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "complex_numbers",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following complex number calculation, what is the final value of magnitude?",
 "code": "z1 = complex(3, 4)\nz2 = complex(1, 2)\nz_sum = z1 + z2\nz_product = z_sum * complex(0, 1)\nmagnitude = abs(z_product)",
 "answer": 7.211102550927978
 }
},
{
 "id": "BL-LN-S004-V011",
 "metadata": {
 "name": "金融计算链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "financial_calculation",
 "execution_pattern": "dependent"
 },
 "task": {
```

```

 "description": "Given the following financial calculation chain, what is the
final value of final_amount?",
 "code": "principal = 1000\nannual_rate = 0.05\ncompounding_periods = 4\nyears =
2\nperiodic_rate = annual_rate / compounding_periods\n\ttotal_periods = compounding_periods *
years\nfinal_amount = round(principal * (1 + periodic_rate) ** total_periods, 2)",
 "answer": 1104.49
 }
},
{
 "id": "BL-LN-S004-V012",
 "metadata": {
 "name": "C浮点精度链变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "floating_precision",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following floating point precision chain, what is the
final value of result when cast to int?",
 "code": "float base = 10.0f;\ndouble step1 = base / 3.0;\nfloat step2 =
(float)step1;\ndouble step3 = step2 * 3.0;\nfloat step4 = (float)step3;\nint result = (int)
(step4 + 0.5f);",
 "answer": 10
 }
},
{
 "id": "BL-LN-S005-V001",
 "metadata": {
 "name": "浮点累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "floating_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following floating point accumulation, what is the
final value of total (rounded to 1 decimal)?",
 "code": "float total = 0.0f;\n\ttotal += 10.5f;\n\ttotal += 15.3f;\n\ttotal +=
7.2f;\n\ttotal *= 1.5f;\n\tprintf(\"Total: %.1f\\n\", total);",
 "answer": 49.5
 }
},
{

```



```
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "bitwise_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following bit flag accumulation, what is the final value of flags?",
 "code": "unsigned int flags = 0;\nflags |= 0x01;\nflags |= 0x04;\nflags |= 0x10;\nflags &= ~0x04;\nflags |= 0x08;",
 "answer": 25
 }
},
{
 "id": "BL-LN-S005-V005",
 "metadata": {
 "name": "Java StringBuilder累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "java",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following Java StringBuilder accumulation, what is the final length of the result?",
 "code": "StringBuilder sb = new\nStringBuilder();\nfor (int i = 0; i < 10; i++) {\n sb.append(\"Java\");\n sb.append(\" is\");\n sb.append(\"awesome\");\n}\nfor (int i = 4; i < 7; i++) {\n sb.delete(4, 7);\n}\nString result = sb.toString();",
 "answer": 11
 }
},
{
 "id": "BL-LN-S005-V006",
 "metadata": {
 "name": "数组累积求和变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "array_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
```

```
 "description": "Given the following array accumulation, what is the final value of running_sum?",
 "code": "numbers = [5, 10, 15, 20]\nrunning_sum = 0\nfor num in numbers:\n running_sum += num\naverage = running_sum / len(numbers)",
 "answer": 50
 },
 {
 "id": "BL-LN-S005-V007",
 "metadata": {
 "name": "内存块累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "memory_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following memory accumulation, what is the final value of total_allocated?",
 "code": "size_t total_allocated = 0;\ntotal_allocated += malloc_size(64);\ntotal_allocated += malloc_size(128);\ntotal_allocated += malloc_size(256);\ntotal_allocated *= 2;\nsize_t overhead = total_allocated / 10;",
 "answer": 896
 },
 },
 {
 "id": "BL-LN-S005-V008",
 "metadata": {
 "name": "递增计数器变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "counter_increment",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following counter increment operations, what is the final value of counter?",
 "code": "int counter = 0;\ncounter++;\ncounter += 5;\ncounter *= 2;\ncounter -= 3;\nprintf(\"Counter: %d\\n\", counter);",
 "answer": 9
 },
 },
 {
```

```
"id": "BL-LN-S005-V009",
"metadata": {
 "name": "JavaScript对象累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "accumulative"
},
"task": {
 "description": "Given the following JavaScript object accumulation, what is the final value of stats.total?",
 "code": "const stats = { total: 0, count: 0 };\\nstats.total += 100;\\nstats.count++;\\nstats.total += 200;\\nstats.count++;\\nstats.average = stats.total / stats.count;",
 "answer": 300
},
{
 "id": "BL-LN-S005-V010",
 "metadata": {
 "name": "缓冲区填充累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "buffer_filling",
 "execution_pattern": "accumulative"
},
 "task": {
 "description": "Given the following buffer filling operations, what is the final value of buf->used?",
 "code": "struct buffer *buf = malloc(sizeof(struct buffer));\\nbuf->size = 100;\\nbuf->data = malloc(buf->size);\\nbuf->used = 0;\\nbuf->used += write_data(buf, \\\"Hello\\\", 5);\\nbuf->used += write_data(buf, \\\"World\\\", 5);\\nbuf->remaining = buf->size - buf->used;",
 "answer": 10
},
{
 "id": "BL-LN-S005-V011",
 "metadata": {
 "name": "统计累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
```

```
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "statistics_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following statistics accumulation, what is the final value of stats['sum']?",
 "code": "stats = {'sum': 0, 'count': 0, 'max': 0}\nvalues = [3, 7, 2, 9, 1]\nfor val in values:\n stats['sum'] += val\n stats['count'] += 1\n stats['max'] = max(stats['max'], val)",
 "answer": 22
 }
},
{
 "id": "BL-LN-S005-V012",
 "metadata": {
 "name": "哈希累积变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "hash_accumulation",
 "execution_pattern": "accumulative"
 },
 "task": {
 "description": "Given the following hash accumulation, what is the final value of hash % 1000?",
 "code": "unsigned long hash = 5381;\nconst char *str = \"test\";\nfor (int i = 0; str[i] != '\\0'; i++) {\n hash = ((hash << 5) + hash) + str[i];\n}\nunsigned int result = hash % 1000;",
 "answer": 178
 }
},
{
 "id": "BL-LN-S006-V001",
 "metadata": {
 "name": "简化屏幕初始化变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "simplified_initialization",
 "execution_pattern": "sequential"
 },
}
```

```
"task": {
 "description": "Given the following simplified screen initialization, what is the final value of s->cursor_color?",
 "code": "void init_screen(struct screen *s) {\n s->width = 80;\n s->height = 25;\n s->cursor_color = -1;\n s->background_color = 0;\n}",
 "answer": -1
},
{
 "id": "BL-LN-S006-V002",
 "metadata": {
 "name": "Python显示配置变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following Python display configuration, what is the final value of display['cursor_color']?",
 "code": "def setup_display(display):\n display['grid'] = create_grid(80, 24)\n display['saved_state'] = None\n display['cursor_style'] = 'DEFAULT'\n display['cursor_color'] = -1\n display['background'] = 'BLACK'",
 "answer": -1
 }
},
{
 "id": "BL-LN-S006-V003",
 "metadata": {
 "name": "扩展属性初始化变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "extended_initialization",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following extended screen initialization, what is the final value of s->total_colors?",
 "code": "void init_extended_screen(struct ext_screen *s) {\n s->primary_grid = create_grid(100, 50);\n s->backup_grid = NULL;\n s->cursor_style = CURSOR_BLOCK;\n s->cursor_color = -1;\n s->palette_size = 256;\n s->total_colors = s->palette_size + 16;\n}",
 "answer": 272
}
```

```
 },
 },
 {
 "id": "BL-LN-S006-V004",
 "metadata": {
 "name": "图形上下文变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "graphics_context",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following graphics context initialization, what is the final value of ctx->effective_alpha?",
 "code": "void init_graphics_context(struct gfx_context *ctx) {\n ctx->surface = create_surface(800, 600, 32);\n ctx->blend_mode = BLEND_ALPHA;\n ctx->global_alpha = 0.8;\n ctx->local_alpha = 0.5;\n ctx->effective_alpha = ctx->global_alpha * ctx->local_alpha;\n ctx->color_depth = 24;\n}",
 "answer": 0.4
 }
 },
 {
 "id": "BL-LN-S006-V005",
 "metadata": {
 "name": "Java Swing组件变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "java",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following Java Swing component initialization, what is the final value of panel.getCursorColor()?",
 "code": "public void initPanel(CustomPanel panel) {\n panel.setLayout(new GridLayout(25, 80));\n panel.setSavedState(null);\n panel.setCursorStyle(CursorStyle.DEFAULT);\n panel.setCursorColor(-1);\n panel.setDoubleBuffered(true);\n}",
 "answer": -1
 }
 },
 {
 "id": "BL-LN-S006-V006",
 "metadata": {
```

```
 "name": "窗口管理器变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "window_manager",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following window manager initialization, what is the final value of wm->active_windows?",
 "code": "void init_window_manager(struct window_mgr *wm) {\n wm->desktop =\n create_desktop(1920, 1080);\n wm->window_list = create_list();\n wm->focus_window =\n NULL;\n wm->max_windows = 64;\n wm->active_windows = 0;\n wm->z_order_counter =\n 1;\n}",
 "answer": 0
 }
},
{
 "id": "BL-LN-S006-V007",
 "metadata": {
 "name": "颜色管理变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "color_management",
 "execution_pattern": "transform"
 },
 "task": {
 "description": "Given the following color management setup, what is the final value of cm->default_bg?",
 "code": "void setup_color_manager(struct color_mgr *cm) {\n cm->palette =\n create_palette(256);\n cm->default_fg = RGB(255, 255, 255);\n cm->default_bg = RGB(0,\n 0, 0);\n cm->highlight_color = RGB(255, 255, 0);\n cm->transparency = 255;\n}",
 "answer": "RGB(0, 0, 0)"
 }
},
{
 "id": "BL-LN-S006-V008",
 "metadata": {
 "name": "JavaScript Canvas变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "javascript",
 }
}
```

```
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_language",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following JavaScript Canvas initialization, what is the final value of context.cursorColor?",
 "code": "function initCanvas(canvas, context) {\n context.canvas = canvas;\n context.savedImageData = null;\n context.cursorStyle = 'default';\n context.cursorColor = -1;\n context.lineWidth = 1;\n}",
 "answer": -1
 }
},
{
 "id": "BL-LN-S006-V009",
 "metadata": {
 "name": "终端仿真器变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "terminal_emulator",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following terminal emulator initialization, what is the final value of term->escape_state?",
 "code": "void init_terminal(struct terminal *term) {\n term->screen_buffer =\n malloc(80 * 25 * sizeof(char));\n term->cursor_x = 0;\n term->cursor_y = 0;\n term->cursor_visible = 1;\n term->escape_state = ESC_NORMAL;\n term->color_mode =\n COLOR_16;\n}",
 "answer": "ESC_NORMAL"
 }
},
{
 "id": "BL-LN-S006-V010",
 "metadata": {
 "name": "字体渲染器变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "font_renderer",
 "execution_pattern": "dependent"
 },
 "task": {
```

```
 "description": "Given the following font renderer initialization, what is the final value of fr->glyph_cache_size?",
 "code": "void init_font_renderer(struct font_renderer *fr) {\n fr->font_face = load_font(\"default.ttf\");\n fr->font_size = 12;\n fr->dpi = 96;\n fr->cache_capacity = 1024;\n fr->glyph_cache_size = fr->cache_capacity * sizeof(struct glyph);\n fr->antialias = 1;\n }",
 "answer": "1024 * sizeof(struct glyph)"
 },
 {
 "id": "BL-LN-S006-V011",
 "metadata": {
 "name": "Python GUI框架变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "gui_framework",
 "execution_pattern": "sequential"
 },
 "task": {
 "description": "Given the following Python GUI initialization, what is the final value of widget.cursor_blink_rate?",
 "code": "def init_text_widget(widget):\n widget.buffer = TextBuffer()\n widget.cursor_position = 0\n widget.cursor_visible = True\n widget.cursor_blink_rate = 500\n widget.selection_start = -1\n widget.selection_end = -1",
 "answer": 500
 },
 },
 {
 "id": "BL-LN-S006-V012",
 "metadata": {
 "name": "OpenGL上下文变式",
 "category": "Block-Level",
 "subcategory": "Linear",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "opengl_context",
 "execution_pattern": "dependent"
 },
 "task": {
 "description": "Given the following OpenGL context initialization, what is the final value of ctx->max_texture_units?",
 "code": "void init_gl_context(struct gl_context *ctx) {\n ctx->version_major = 3;\n ctx->version_minor = 3;\n ctx->profile = GL_CORE_PROFILE;\n glGetIntegerv(GL_MAX_TEXTURE_IMAGE_UNITS, &ctx->max_texture_units);\n ctx->current_program = 0;\n ctx->vertex_array_binding = 0;\n }",
 "answer": 32
 }
 }
}
```

```
 "answer": "implementation_dependent"
 }
}
```

## 2B - 条件代码块 [ Conditional ] (96)

# 条件代码块推理变式生成提示词

## 任务目标

基于给定的条件代码块种子任务，生成多样化的变式任务来全面测试大模型对条件分支逻辑、控制流分析、路径选择和分支执行结果预测的理解能力。

## 条件代码块特征分析

条件代码块是指包含`if-else`、`switch-case`、三元运算符等分支控制结构的代码段，重点测试模型对条件判断、分支选择、执行路径预测和结果状态推理的理解。

## 变式生成维度

### 1. 条件结构类型变式

- \*\*简单`if-else`变式\*\*：基础的二分支条件判断
- \*\*多重`if-else`变式\*\*：`if-elif-else`的多分支条件链
- \*\*嵌套条件变式\*\*：条件语句的多层嵌套结构
- \*\*`switch-case`变式\*\*：多路分支的`switch`语句
- \*\*三元运算符变式\*\*：条件表达式的简化形式
- \*\*`guard`子句变式\*\*：早期返回的条件保护语句

### 2. 条件表达式复杂度变式

- \*\*简单条件变式\*\*：单一变量的直接比较
- \*\*复合条件变式\*\*：使用`&&`、`||`、`!`的复合逻辑表达式
- \*\*范围判断变式\*\*：数值范围的边界条件判断
- \*\*类型检查变式\*\*：对象类型或`NULL`的条件检查
- \*\*位运算条件变式\*\*：位操作结果的条件判断
- \*\*函数返回值条件变式\*\*：基于函数调用结果的条件判断

### 3. 分支执行路径变式

- \*\*单路径执行变式\*\*：只有一个分支会被执行的情况
- \*\*多路径可能变式\*\*：根据不同输入可能执行不同分支
- \*\*死分支变式\*\*：由于条件恒定而永不执行的分支
- \*\*必然分支变式\*\*：由于条件恒定而必然执行的分支
- \*\*短路执行变式\*\*：逻辑短路导致的分支跳过
- \*\*`fallthrough`变式\*\*：`switch`语句中的贯穿执行

### 4. 条件依赖关系变式

- \*\*独立条件变式\*\*：条件判断之间相互独立
- \*\*依赖条件变式\*\*：后续条件依赖前面条件的结果
- \*\*互斥条件变式\*\*：条件之间具有互斥关系
- \*\*包含条件变式\*\*：一个条件包含于另一个条件
- \*\*重叠条件变式\*\*：条件之间存在部分重叠
- \*\*递进条件变式\*\*：条件判断具有递进关系

### 5. 变量状态变化变式

- \*\*条件内赋值变式\*\*：在条件分支内修改变量值

- **累积状态变式**: 多个分支对同一变量的累积修改
- **状态重置变式**: 某个分支重置变量到初始状态
- **条件状态变式**: 变量状态影响后续条件判断
- **副作用变式**: 条件判断本身产生的副作用
- **状态传播变式**: 变量状态在分支间的传播

### ### 6. 错误处理和边界变式

- **空值检查变式**: `NULL`、`None`、`undefined`的条件检查
- **边界值变式**: 数值边界条件的处理
- **溢出检查变式**: 数值溢出的条件检测
- **资源有效性变式**: 资源分配成功与否的条件检查
- **参数验证变式**: 输入参数有效性的条件验证
- **异常路径变式**: 异常情况的条件处理路径

### ### 7. 控制流复杂度变式

- **线性分支变式**: 简单的线性分支结构
- **树形分支变式**: 层次化的树形分支结构
- **网状分支变式**: 交叉引用的复杂分支网络
- **循环内分支变式**: 循环体内的条件分支
- **递归分支变式**: 递归函数中的条件分支
- **跳转分支变式**: `goto`、`break`、`continue`的条件跳转

### ### 8. 数据类型特化变式

- **数值比较变式**: 整数、浮点数的条件比较
- **字符串比较变式**: 字符串相等性和大小比较
- **指针比较变式**: 指针地址和指向内容的比较
- **对象比较变式**: 对象引用和内容的比较
- **数组比较变式**: 数组长度和元素的条件判断
- **结构体比较变式**: 结构体字段的条件比较

### ### 9. 优化和性能变式

- **分支预测变式**: 可预测和不可预测的分支模式
- **条件消除变式**: 编译器可优化消除的条件
- **常量折叠变式**: 编译时可确定的条件表达式
- **跳转表变式**: `switch`语句的跳转表优化
- **分支消除变式**: 永不执行分支的消除优化
- **条件合并变式**: 多个条件的合并优化

### ### 10. 语言特性变式

- **Python特性变式**: `elif`、布尔表达式、条件表达式
- **C语言特性变式**: `switch-case`、条件运算符、宏条件编译
- **Java特性变式**: `switch`表达式、`instanceof`检查
- **JavaScript特性变式**: 真值判断、可选链操作符
- **跨语言移植变式**: 相同逻辑在不同语言中的条件实现

## ## 复杂度层次设计

### ### 简单条件 (Easy)

- 单层`if-else`或简单`switch-case`
- 基础的比较运算符条件
- 明确的二分支或三分支选择
- 直观的条件判断逻辑

### ### 中等条件 (Medium)

- 2-3层的嵌套条件或中等复杂的条件链
- 包含逻辑运算符的复合条件
- 涉及函数调用或简单计算的条件
- 需要仔细分析分支选择的情况

### ### 复杂条件 (Hard)

- 3层以上的深度嵌套或复杂条件网络
- 复杂的逻辑表达式和多重条件依赖
- 涉及指针、内存管理或复杂数据结构
- 需要深入理解控制流和执行语义

### ### 专家级条件 (Expert)

- 极度复杂的嵌套和条件组合
- 涉及高级语言特性和优化考虑
- 多重边界条件和异常处理
- 需要专业级别的代码理解能力

## ## 生成策略

### ### 种子分析策略

1. \*\*识别条件模式\*\*: 分析种子的条件结构类型和复杂度
2. \*\*提取判断逻辑\*\*: 识别条件表达式和分支选择逻辑
3. \*\*分析执行路径\*\*: 追踪不同输入下的执行路径
4. \*\*确定关键变量\*\*: 找出影响条件判断的关键变量

### ### 变式设计原则

1. \*\*路径完整性\*\*: 确保覆盖所有可能的执行路径
2. \*\*边界条件\*\*: 重点测试边界值和特殊情况
3. \*\*逻辑一致性\*\*: 保证条件逻辑的正确性和一致性
4. \*\*现实相关性\*\*: 优先设计实际编程中常见的条件模式

### ### 质量保证

1. \*\*路径验证\*\*: 验证每个分支路径的执行正确性
2. \*\*条件覆盖\*\*: 确保所有条件分支都得到测试
3. \*\*边界测试\*\*: 包含各种边界和异常条件
4. \*\*语义正确\*\*: 保证条件逻辑符合语言语义

## ## 输出格式要求

```
```json
[
  {
    "id": "BL-CD-S00X-V001",
    "metadata": {
      "name": "条件代码块变式名称",
      "category": "Block-Level",
      "subcategory": "Conditional",
      "type": "variant",
      "source": "Generated",
      "language": "目标语言",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
    }
  }
]
```

```

        "variant_type": "变式类型标签",
        "condition_pattern":
"simple_if/nested_if/switch_case/ternary/complex_logic",
        "branch_count": "分支数量"
    },
    "task": {
        "description": "条件代码块推理任务描述",
        "code": "包含条件分支的代码块",
        "answer": "特定输入下的执行结果"
    }
},
{下一个变式...}
]

```

特殊字段说明

condition_pattern: 标识条件结构的模式类型

branch_count: 记录分支的数量

variant_type: 描述具体的变式类型

生成目标

为每个提供的种子任务生成12-18个条件代码块变式，确保：

每种难度等级至少包含3-4个变式

涵盖至少8-10种不同的条件模式

包含2-3个跨语言的变式

至少包含4-5个边界条件和错误处理的变式

包含不同分支数量的变式（2分支到5+分支）

重点测试场景

分支选择准确性：给定特定输入，能否正确选择执行分支

嵌套逻辑理解：对多层嵌套条件的正确理解

短路逻辑：逻辑运算符短路特性的理解

边界条件处理：边界值和特殊情况的正确处理

状态变化追踪：条件分支对程序状态的影响

请基于此提示词，为给定的条件代码块种子任务生成全面的变式集合，输出格式为包含所有变式的JSON数组。

```

[
{
    "id": "BL-CD-S001-V001",
    "metadata": {
        "name": "简单条件-边界值测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_condition",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {

```

```
        "description": "Given the following conditional block where sysconf returns 0, what is the return value?",  
        "code": "size_t crypt_getpagesize(void) {\n            long r = sysconf(_SC_PAGESIZE);\n            if (r <= 0) {\n                return DEFAULT_MEM_ALIGNMENT;\n            } else {\n                return (size_t)r;\n            }\n        }  
    },  
    {  
        "id": "BL-CD-S001-V002",  
        "metadata": {  
            "name": "简单条件-负值测试",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "c",  
            "difficulty": "easy",  
            "intervention": 0,  
            "variant_type": "negative_value",  
            "condition_pattern": "simple_if",  
            "branch_count": "2"  
        },  
        "task": {  
            "description": "Given the following conditional block where sysconf returns -1, what is the return value?",  
            "code": "size_t crypt_getpagesize(void) {\n                long r = sysconf(_SC_PAGESIZE);\n                if (r <= 0) {\n                    return DEFAULT_MEM_ALIGNMENT;\n                } else {\n                    return (size_t)r;\n                }\n            }  
        }  
    },  
    {  
        "id": "BL-CD-S001-V003",  
        "metadata": {  
            "name": "条件反转变式",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "c",  
            "difficulty": "medium",  
            "intervention": 1,  
            "variant_type": "condition_inversion",  
            "condition_pattern": "simple_if",  
            "branch_count": "2"  
        },  
        "task": {  
            "description": "Given the following conditional block where sysconf returns 8192, what is the return value?",  
            "code": "size_t crypt_getpagesize(void) {\n                long r = sysconf(_SC_PAGESIZE);\n                if (r > 0) {\n                    return (size_t)r;\n                } else {\n                    return DEFAULT_MEM_ALIGNMENT;\n                }\n            }  
        }  
    }  
}
```

```
        "answer": 8192
    }
},
{
    "id": "BL-CD-S001-V004",
    "metadata": {
        "name": "三元运算符变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "ternary_operator",
        "condition_pattern": "ternary",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following ternary conditional where sysconf returns 2048, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    return (r <= 0) ? DEFAULT_MEM_ALIGNMENT : (size_t)r;\n}\n",
        "answer": 2048
    }
},
{
    "id": "BL-CD-S001-V005",
    "metadata": {
        "name": "多重条件检查变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "multiple_conditions",
        "condition_pattern": "complex_if",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following conditional block where sysconf returns 1024, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    if (r <= 0) {\n        return DEFAULT_MEM_ALIGNMENT;\n    } else if (r < 512) {\n        return 512;\n    } else {\n        return (size_t)r;\n    }\n}\n",
        "answer": 1024
    }
},
{
    "id": "BL-CD-S001-V006",
    "metadata": {
```

```
        "name": "范围检查变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "range_check",
        "condition_pattern": "complex_logic",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following conditional block where sysconf returns 16384, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    if (r <= 0) {\n        return DEFAULT_MEM_ALIGNMENT;\n    } else if (r >= 4096 && r <= 8192) {\n        return (size_t)r;\n    } else if (r > 8192) {\n        return 8192;\n    }\n    else {\n        return 4096;\n    }\n}",
        "answer": 8192
    }
},
{
    "id": "BL-CD-S001-V007",
    "metadata": {
        "name": "Python版本转换",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "language_conversion",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following conditional block where get_page_size returns 4096, what is the return value?",
        "code": "def crypt_getpagesize():\n    r = get_page_size()\n    if r <= 0:\n        return DEFAULT_MEM_ALIGNMENT\n    else:\n        return r",
        "answer": 4096
    }
},
{
    "id": "BL-CD-S001-V008",
    "metadata": {
        "name": "状态变量累积变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "state_variable",
        "condition_pattern": "cumulative",
        "branch_count": "2"
    }
}
```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "state_accumulation",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following conditional block where sysconf returns 4096, what is the final value of total_size?",
        "code": "int total_size = 0;\nsize_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n    if (r <= 0) {\n        total_size += DEFAULT_MEM_ALIGNMENT;\n    }\n    return DEFAULT_MEM_ALIGNMENT;\n} else {\n    total_size += (size_t)r;\n}\nreturn (size_t)r;\n}",
        "answer": 4096
    }
},
{
    "id": "BL-CD-S001-V009",
    "metadata": {
        "name": "嵌套函数调用条件",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_function_calls",
        "condition_pattern": "complex_logic",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following conditional block where sysconf returns 4096 and validate_page_size returns 1, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    if (r <= 0 || !validate_page_size(r)) {\n        return DEFAULT_MEM_ALIGNMENT;\n    }\n\n    else if (is_power_of_two(r)) {\n        return (size_t)r;\n    }\n\n    else {\n        return round_to_power_of_two(r);\n    }\n}",
        "answer": 4096
    }
},
{
    "id": "BL-CD-S001-V010",
    "metadata": {
        "name": "位运算条件检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
    }
}
```

```
        "variant_type": "bitwise_conditions",
        "condition_pattern": "complex_logic",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following conditional block where sysconf returns 4096, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    if (r <= 0) {\n        return DEFAULT_MEM_ALIGNMENT;\n    } else if ((r & (r - 1)) == 0)\n        return (size_t)r;\n    } else {\n        return (size_t)(1 << (int)log2(r));\n    }\n}",
        "answer": 4096
    }
},
{
    "id": "BL-CD-S001-V011",
    "metadata": {
        "name": "Guard子句变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "guard_clause",
        "condition_pattern": "guard_clause",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following guard clause where sysconf returns 4096, what is the return value?",
        "code": "size_t crypt_getpagesize(void) {\n    long r = sysconf(_SC_PAGESIZE);\n\n    if (r <= 0)\n        return DEFAULT_MEM_ALIGNMENT;\n    } else\n        return (size_t)r;\n}",
        "answer": 4096
    }
},
{
    "id": "BL-CD-S001-V012",
    "metadata": {
        "name": "溢出检查变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "overflow_check",
        "condition_pattern": "complex_logic",
        "branch_count": "4"
    },
    "task": {

```

```
        "description": "Given the following conditional block where sysconf returns 4096, what is the return value?",  
        "code": "size_t crypt_getpagesize(void) {\n            long r = sysconf(_SC_PAGESIZE);\\n  
            if (r <= 0) {\\n                return DEFAULT_MEM_ALIGNMENT;\\n            } else if (r > SIZE_MAX) {\\n                return SIZE_MAX;\\n            } else if (r < 512) {\\n                return 512;\\n            } else {\\n                return (size_t)r;\\n            }\\n        }\\n    },  
  
{  
    "id": "BL-CD-S002-V001",  
    "metadata": {  
        "name": "嵌套条件-边界分数",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 1,  
        "variant_type": "boundary_condition",  
        "condition_pattern": "nested_if",  
        "branch_count": "4"  
    },  
    "task": {  
        "description": "Given the following nested conditional block where score is 90, what is the final value of grade?",  
        "code": "score = 90\\n if score >= 90:\\n    if score >= 95:\\n        grade = 'A+'\\n    else:\\n        grade = 'A'\\nelse:\\n    if score >= 80:\\n        grade = 'B'\\n    else:\\n        grade = 'C'\\nprint(f\"Grade: {grade}\")",  
        "answer": "A"  
    },  
},  
{  
    "id": "BL-CD-S002-V002",  
    "metadata": {  
        "name": "嵌套条件-最高分数",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 1,  
        "variant_type": "maximum_boundary",  
        "condition_pattern": "nested_if",  
        "branch_count": "4"  
    },  
    "task": {  
        "description": "Given the following nested conditional block where score is 98, what is the final value of grade?",  
    }
```

```
        "code": "score = 98\nif score >= 90:\n    if score >= 95:\n        grade =\n'A+'\n    else:\n        grade = 'A'\nelse:\n    if score >= 80:\n        grade = 'B'\nelse:\n    grade = 'C'\nprint(f\"Grade: {grade}\")",
        "answer": "A+"
    },
},
{
    "id": "BL-CD-S002-V003",
    "metadata": {
        "name": "嵌套条件-低分数段",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "low_score_range",
        "condition_pattern": "nested_if",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following nested conditional block where score is 75, what is the final value of grade?",
        "code": "score = 75\nif score >= 90:\n    if score >= 95:\n        grade =\n'A+'\n    else:\n        grade = 'A'\nelse:\n    if score >= 80:\n        grade = 'B'\nelse:\n    grade = 'C'\nprint(f\"Grade: {grade}\")",
        "answer": "C"
    }
},
{
    "id": "BL-CD-S002-V004",
    "metadata": {
        "name": "三层嵌套变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "three_level_nesting",
        "condition_pattern": "deep_nested_if",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the following three-level nested conditional where score is 92, what is the final value of grade?",
        "code": "score = 92\nif score >= 90:\n    if score >= 95:\n        if score >=\n98:\n            grade = 'A++'\n        else:\n            grade = 'A+'\n    else:\n        grade = 'A'\nelse:\n    if score >= 80:\n        grade = 'B'\n    else:\n        grade =\n'C'\nprint(f\"Grade: {grade}\")",
        "answer": "A"
    }
}
```

```

        }
    },
    {
        "id": "BL-CD-S002-V005",
        "metadata": {
            "name": "扩展分级系统",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "extended_grading",
            "condition_pattern": "nested_if",
            "branch_count": "8"
        },
        "task": {
            "description": "Given the following extended grading system where score is 85, what is the final value of grade?",
            "code": "score = 85\nif score >= 90:\n    if score >= 95:\n        grade = 'A+'\n    else:\n        grade = 'A'\nelse:\n    if score >= 80:\n        if score >= 85:\n            grade = 'B+'\n        else:\n            grade = 'B'\n    else:\n        if score >= 70:\n            grade = 'C'\n        else:\n            grade = 'D'\nprint(f\"Grade: {grade}\")",
            "answer": "B+"
        }
    },
    {
        "id": "BL-CD-S002-V006",
        "metadata": {
            "name": "C语言版本转换",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "language_conversion",
            "condition_pattern": "nested_if",
            "branch_count": "4"
        },
        "task": {
            "description": "Given the following nested conditional block where score is 85, what is the final value of grade?",
            "code": "int score = 85;\nchar grade;\nif (score >= 90) {\n    if (score >= 95)\n        grade = 'A';\n    else {\n        grade = 'B';\n    }\n} else {\n    if (score >= 80)\n        grade = 'C';\n    else {\n        grade = 'D';\n    }\n}\nprintf(\"Grade: %c\\n\", grade);",
            "answer": "C"
        }
    },
    {
        "id": "BL-CD-S002-V007",
        "metadata": {
            "name": "嵌套循环与条件语句",
            "category": "Block-Level",
            "subcategory": "Loops",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "loop",
            "condition_pattern": "nested_loops",
            "branch_count": "4"
        },
        "task": {
            "description": "Given the following nested loop block where score is 85, what is the final value of grade?",
            "code": "score = 85\nfor i in range(1, 4):\n    for j in range(1, 4):\n        if i > j:\n            grade = 'A'\n        else:\n            grade = 'B'\n    else:\n        if i > j:\n            grade = 'C'\n        else:\n            grade = 'D'\nprint(f\"Grade: {grade}\")",
            "answer": "D"
        }
    }
]

```

```
{  
    "id": "BL-CD-S002-V007",  
    "metadata": {  
        "name": "状态累积变式",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "python",  
        "difficulty": "hard",  
        "intervention": 2,  
        "variant_type": "state_accumulation",  
        "condition_pattern": "nested_if",  
        "branch_count": "4"  
    },  
    "task": {  
        "description": "Given the following nested conditional with state tracking where score is 85, what is the final value of bonus_points?",  
        "code": "score = 85\nbonus_points = 0\nif score >= 90:\n    bonus_points += 10\nif score >= 95:\n    bonus_points += 5\n    grade = 'A+'\nelse:\ngrade = 'A'\nelse:\n    if score >= 80:\n        bonus_points += 3\n        grade = 'B'\nelse:\n    grade = 'C'\nprint(f\"Grade: {grade}, Bonus: {bonus_points}\")",  
        "answer": 3  
    },  
},  
{  
    "id": "BL-CD-S002-V008",  
    "metadata": {  
        "name": "复合条件嵌套",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "python",  
        "difficulty": "expert",  
        "intervention": 3,  
        "variant_type": "compound_conditions",  
        "condition_pattern": "complex_logic",  
        "branch_count": "6"  
    },  
    "task": {  
        "description": "Given the following compound conditional where score is 85 and attendance is 0.9, what is the final value of grade?",  
        "code": "score = 85\nattendance = 0.9\nif score >= 90 and attendance >= 0.8:\nif score >= 95 or attendance >= 0.95:\n    grade = 'A+'\nelse:\n    grade = 'A'\nelse:\n    if score >= 80 and attendance >= 0.75:\n        grade = 'B'\n    elif score >= 70:\n        grade = 'C'\n    else:\n        grade = 'D'\nprint(f\"Grade: {grade}\")",  
        "answer": "B"  
    },  
},  
{  
    "id": "BL-CD-S002-V009",  
    "metadata": {
```

```
        "name": "异常处理嵌套",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "exception_handling",
        "condition_pattern": "nested_if",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following nested conditional with validation where score is 85, what is the final value of grade?",
        "code": "score = 85\nif score < 0 or score > 100:\n    grade =\n'Invalid'\nelse:\n    if score >= 90:\n        if score >= 95:\n            grade = 'A+'\n        else:\n            grade = 'A'\n    else:\n        if score >= 80:\n            grade = 'C'\n        else:\n            grade = 'B'\nprint(f\"Grade: {grade}\")",
        "answer": "B"
    }
},
{
    "id": "BL-CD-S002-V010",
    "metadata": {
        "name": "短路逻辑嵌套",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "short_circuit_logic",
        "condition_pattern": "complex_logic",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following nested conditional with short-circuit logic where score is None, what is the final value of grade?",
        "code": "score = None\nif score is not None and score >= 90:\n    if score >= 95:\n        grade = 'A+'\n    else:\n        grade = 'A'\nelse:\n    if score is not None and score >= 80:\n        grade = 'B'\n    else:\n        grade =\n'Incomplete'\nprint(f\"Grade: {grade}\")",
        "answer": "Incomplete"
    }
},
{
    "id": "BL-CD-S002-V011",
    "metadata": {
        "name": "多变量嵌套条件",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "multiple_variables",
        "condition_pattern": "multiple_if",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following nested conditional with validation where score is 85, what is the final value of grade?",
        "code": "score = 85\nif score < 0 or score > 100:\n    grade =\n'Invalid'\nelse:\n    if score >= 90:\n        if score >= 95:\n            grade = 'A+'\n        else:\n            grade = 'A'\n    else:\n        if score >= 80:\n            grade = 'C'\n        else:\n            grade = 'B'\nprint(f\"Grade: {grade}\")",
        "answer": "B"
    }
}
```

```
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "multi_variable",
        "condition_pattern": "nested_if",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the following multi-variable nested conditional where exam_score is 85 and project_score is 90, what is the final value of grade?",
        "code": "exam_score = 85\nproject_score = 90\naverage = (exam_score + project_score) / 2\nif average >= 90:\n    if exam_score >= 85 and project_score >= 85:\n        grade = 'A+'\n    else:\n        grade = 'A'\nelse:\n    if average >= 80:\n        if min(exam_score, project_score) >= 75:\n            grade = 'B+'\n        else:\n            grade = 'B'\n    else:\n        grade = 'C'\nprint(f\"Grade: {grade}\")",
        "answer": "A+"
    }
},
{
    "id": "BL-CD-S002-V012",
    "metadata": {
        "name": "递归嵌套模式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "recursive_pattern",
        "condition_pattern": "deep_nested_if",
        "branch_count": "8"
    },
    "task": {
        "description": "Given the following deep nested conditional where score is 85, what is the final value of detailed_grade?",
        "code": "score = 85\nif score >= 90:\n    if score >= 95:\n        if score >= 98:\n            detailed_grade = 'Excellent+'\n        else:\n            detailed_grade = 'Very Good+'\n    else:\n        if score >= 92:\n            detailed_grade = 'Very Good'\n        else:\n            detailed_grade = 'Good'\n    else:\n        if score >= 87:\n            detailed_grade = 'Good+'\n        else:\n            detailed_grade = 'Satisfactory'\nelse:\n    if score >= 80:\n        detailed_grade = 'Needs Improvement'\n    else:\n        detailed_grade = 'Needs Improvement'\nprint(f\"Detailed Grade: {detailed_grade}\")",
        "answer": "Good"
    }
},
{
    "id": "BL-CD-S003-V001",
    "metadata": {
```

```
        "name": "边界值-10",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_condition",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following else-if chain where value is 10, what is the final value of category?",
        "code": "int value = 10;\nchar *category;\nif (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "medium"
    }
},
{
    "id": "BL-CD-S003-V002",
    "metadata": {
        "name": "边界值-20",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_condition",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following else-if chain where value is 20, what is the final value of category?",
        "code": "int value = 20;\nchar *category;\nif (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "high"
    }
},
{
    "id": "BL-CD-S003-V003",
    "metadata": {
        "name": "极值测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "boundary_condition",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    }
}
```

```
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "extreme_value",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following else-if chain where value is 35, what is the final value of category?",
        "code": "int value = 35;\nchar *category;\nif (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "extreme"
    }
},
{
    "id": "BL-CD-S003-V004",
    "metadata": {
        "name": "扩展分类链",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "extended_chain",
        "condition_pattern": "else_if_chain",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the following extended else-if chain where value is 15, what is the final value of category?",
        "code": "int value = 15;\nchar *category;\nif (value < 5) {\n    category = \"very_low\";\n} else if (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else if (value < 50) {\n    category = \"very_high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "medium"
    }
},
{
    "id": "BL-CD-S003-V005",
    "metadata": {
        "name": "条件顺序变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "order_variation",
        "condition_pattern": "else_if_chain",
        "branch_count": "5"
    }
}
```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "condition_reordering",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following reordered else-if chain where value is 15, what is the final value of category?",
        "code": "int value = 15;\nchar *category;\nif (value >= 30) {\n    category = \"extreme\";\n} else if (value >= 20) {\n    category = \"high\";\n} else if (value >= 10)\n{\n    category = \"medium\";\n} else {\n    category = \"low\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "medium"
    }
},
{
    "id": "BL-CD-S003-V006",
    "metadata": {
        "name": "Python版本转换",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "language_conversion",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following elif chain where value is 15, what is the final value of category?",
        "code": "value = 15\nif value < 10:\n    category = \"low\"\nelif value < 20:\n    category = \"medium\"\nelif value < 30:\n    category = \"high\"\nelse:\n    category = \"extreme\"\nprint(f\"Category: {category}\")",
        "answer": "medium"
    }
},
{
    "id": "BL-CD-S003-V007",
    "metadata": {
        "name": "浮点数比较变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "floating_point",
    }
}
```

```
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following else-if chain where value is 15.5, what is the final value of category?",
        "code": "double value = 15.5;\nchar *category;\nif (value < 10.0) {\n    category = \"low\";\n} else if (value < 20.0) {\n    category = \"medium\";\n} else if (value < 30.0) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
        "answer": "medium"
    }
},
{
    "id": "BL-CD-S003-V008",
    "metadata": {
        "name": "状态累积变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "state_accumulation",
        "condition_pattern": "else_if_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following else-if chain with scoring where value is 15, what is the final value of score?",
        "code": "int value = 15;\nint score = 0;\nchar *category;\nif (value < 10) {\n    category = \"low\";\n    score = 1;\n} else if (value < 20) {\n    category = \"medium\";\n    score = 2;\n} else if (value < 30) {\n    category = \"high\";\n    score = 3;\n} else {\n    category = \"extreme\";\n    score = 4;\n}\nprintf(\"category: %s, Score: %d\\n\", category, score);",
        "answer": 2
    }
},
{
    "id": "BL-CD-S003-V009",
    "metadata": {
        "name": "复合条件变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "compound_conditions",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    }
}
```

```
        },
        "task": {
            "description": "Given the following complex else-if chain where value is 15 and flag is 1, what is the final value of category?",
            "code": "int value = 15;\nint flag = 1;\nchar *category;\nif (value < 10 && flag == 0) {\n    category = \"low_special\";\n} else if (value < 10) {\n    category = \"low\";\n} else if (value < 20 && flag == 1) {\n    category = \"medium_enhanced\";\n} else if (value < 20) {\n    category = \"medium\";\n} else {\n    category = \"high\";\n}\nprintf(\"Category: %s\\n\", category);",
            "answer": "medium_enhanced"
        }
    },
    {
        "id": "BL-CD-S003-V010",
        "metadata": {
            "name": "函数调用条件",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "function_call_conditions",
            "condition_pattern": "else_if_chain",
            "branch_count": "4"
        },
        "task": {
            "description": "Given the following else-if chain where get_threshold() returns 15 and value is 15, what is the final value of category?",
            "code": "int value = 15;\nchar *category;\nint threshold = get_threshold();\nif (value < threshold - 5) {\n    category = \"low\";\n} else if (value < threshold + 5) {\n    category = \"medium\";\n} else if (value < threshold + 15) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",
            "answer": "medium"
        }
    },
    {
        "id": "BL-CD-S003-V011",
        "metadata": {
            "name": "负值处理变式",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "negative_value_handling",
            "condition_pattern": "else_if_chain",
            "branch_count": "5"
        },
        "task": {

```

```
        "description": "Given the following else-if chain where value is -5, what is the final value of category?",  
        "code": "int value = -5;\nchar *category;\nif (value < 0) {\n    category = \"negative\";\n} else if (value < 10) {\n    category = \"low\";\n} else if (value < 20) {\n    category = \"medium\";\n} else if (value < 30) {\n    category = \"high\";\n} else {\n    category = \"extreme\";\n}\nprintf(\"Category: %s\\n\", category);",  
        "answer": "negative"  
    },  
},  
{  
    "id": "BL-CD-S003-V012",  
    "metadata": {  
        "name": "范围重叠检查",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "expert",  
        "intervention": 3,  
        "variant_type": "range_overlap",  
        "condition_pattern": "complex_logic",  
        "branch_count": "6"  
    },  
    "task": {  
        "description": "Given the following range overlap else-if chain where value is 15, what is the final value of category?",  
        "code": "int value = 15;\nchar *category;\nif (value >= 0 && value <= 5) {\n    category = \"range_a\";\n} else if (value > 5 && value <= 15) {\n    category = \"range_b\";\n} else if (value > 10 && value <= 25) {\n    category = \"range_c\";\n} else if (value > 20 && value <= 35) {\n    category = \"range_d\";\n} else {\n    category = \"out_of_range\";\n}\nprintf(\"Category: %s\\n\", category);",  
        "answer": "range_b"  
    },  
},  
{  
    "id": "BL-CD-S004-V001",  
    "metadata": {  
        "name": "周末测试",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "easy",  
        "intervention": 0,  
        "variant_type": "default_case",  
        "condition_pattern": "switch_case",  
        "branch_count": "6"  
    },  
    "task": {
```

```
        "description": "Given the following switch block where day is 7, what is the final value of day_name?",  
        "code": "int day = 7;\nchar *day_name;\nswitch (day) {\n    case 1: day_name =\n        \"Monday\"; break;\n    case 2: day_name = \"Tuesday\"; break;\n    case 3: day_name =\n        \"Wednesday\"; break;\n    case 4: day_name = \"Thursday\"; break;\n    case 5: day_name =\n        \"Friday\"; break;\n    default: day_name = \"weekend\";\n}\nprintf(\"Day: %s\\n\", day_name);",  
        "answer": "weekend"  
    }  
},  
{  
    "id": "BL-CD-S004-V002",  
    "metadata": {  
        "name": "星期一测试",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "easy",  
        "intervention": 0,  
        "variant_type": "first_case",  
        "condition_pattern": "switch_case",  
        "branch_count": "6"  
    },  
    "task": {  
        "description": "Given the following switch block where day is 1, what is the final value of day_name?",  
        "code": "int day = 1;\nchar *day_name;\nswitch (day) {\n    case 1: day_name =\n        \"Monday\"; break;\n    case 2: day_name = \"Tuesday\"; break;\n    case 3: day_name =\n        \"Wednesday\"; break;\n    case 4: day_name = \"Thursday\"; break;\n    case 5: day_name =\n        \"Friday\"; break;\n    default: day_name = \"weekend\";\n}\nprintf(\"Day: %s\\n\", day_name);",  
        "answer": "Monday"  
    }  
},  
{  
    "id": "BL-CD-S004-V003",  
    "metadata": {  
        "name": "字符switch变式",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "medium",  
        "intervention": 1,  
        "variant_type": "character_switch",  
        "condition_pattern": "switch_case",  
        "branch_count": "5"  
    },  
    "task": {
```



```
        "code": "int category = 1;\nint subcategory = 2;\nchar *result;\nswitch\n(category) {\n    case 1:\n        switch (subcategory) {\n            case 1: result =\n                \"Type A1\"; break;\n            case 2: result = \"Type A2\"; break;\n            default:\n                result = \"Type A Unknown\";\n        }\n        break;\n    case 2:\n        switch (subcategory) {\n            case 1: result = \"Type B1\"; break;\n            case 2: result = \"Type B Unknown\";\n            default: result = \"Unknown Category\";\n        }\n        break;\n    default: result = \"Unknown Category\";\n}\nprintf(\"Result: %s\\n\", result);",
        "answer": "Type A2"
    },
    {
        "id": "BL-CD-S004-V006",
        "metadata": {
            "name": "缺失break的switch",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "missing_break",
            "condition_pattern": "switch_case",
            "branch_count": "6"
        },
        "task": {
            "description": "Given the following switch with missing break where day is 2, what is the final value of day_name?",
            "code": "int day = 2;\nchar *day_name;\nswitch (day) {\n    case 1: day_name =\n        \"Monday\";\n    case 2: day_name = \"Tuesday\";\n    case 3: day_name = \"Wednesday\";\n    break;\n    case 4: day_name = \"Thursday\"; break;\n    case 5: day_name = \"Friday\";\n    break;\n    default: day_name = \"weekend\";\n}\nprintf(\"Day: %s\\n\", day_name);",
            "answer": "Wednesday"
        }
    },
    {
        "id": "BL-CD-S004-V007",
        "metadata": {
            "name": "Python match变式",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "language_conversion",
            "condition_pattern": "match_case",
            "branch_count": "6"
        },
        "task": {
```

```

        "description": "Given the following match statement where day is 3, what is the
final value of day_name?",
        "code": "day = 3\nmatch day:\n    case 1:\n        day_name = \"Monday\"\n    case 2:\n        day_name = \"Tuesday\"\n    case 3:\n        day_name = \"Wednesday\"\n    case 4:\n        day_name = \"Thursday\"\n    case 5:\n        day_name = \"Friday\"\n    case _:\n        day_name = \"weekend\"\nprintf(f\"Day: {day_name}\")",
        "answer": "Wednesday"
    },
},
{
    "id": "BL-CD-S004-V008",
    "metadata": {
        "name": "范围case变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "range_case",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following switch with ranges where score is 85, what
is the final value of grade using GNU C extension?",
        "code": "int score = 85;\nchar *grade;\nswitch (score / 10) {\n    case 10:\n        grade = \"A\";\n        break;\n    case 8:\n        grade = \"B\";\n        break;\n    case 7:\n        grade = \"C\";\n        break;\n    default:\n        grade = \"F\";\n}\nprintf(\"Grade: %s\\n\", grade);",
        "answer": "B"
    }
},
{
    "id": "BL-CD-S004-V009",
    "metadata": {
        "name": "状态机switch变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "state_machine",
        "condition_pattern": "switch_case",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following state machine switch where state is 1 and
input is 'A', what is the final value of next_state?",
    }
}

```

```
        "code": "int state = 1;\nchar input = 'A';\nint next_state;\nswitch (state) {\n    case 0:\n        next_state = (input == 'A') ? 1 : 0;\n        break;\n    case 1:\n        next_state = (input == 'B') ? 2 : ((input == 'A') ? 1 : 0);\n        break;\n    case 2:\n        next_state = (input == 'C') ? 3 : 0;\n        break;\n    case 3:\n        next_state = 0;\n        break;\n    default:\n        next_state = 0;\n}\nprintf(\"Next state: %d\\n\", next_state);",
        "answer": 1
    },
},
{
    "id": "BL-CD-S004-V010",
    "metadata": {
        "name": "函数指针switch变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "function_pointer",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following function pointer switch where operation is 2, what is the final value of result when add(5,3) returns 8?",
        "code": "int operation = 2;\nint a = 5, b = 3;\nint result;\nint (*func_ptr)(int, int);\nswitch (operation) {\n    case 1: func_ptr = add; break;\n    case 2: func_ptr = subtract; break;\n    case 3: func_ptr = multiply; break;\n    default: func_ptr = NULL;\n}\nif (func_ptr != NULL) {\n    result = func_ptr(a, b);\n} else {\n    result = 0;\n}\nprintf(\"Result: %d\\n\", result);",
        "answer": 2
    },
},
{
    "id": "BL-CD-S004-V011",
    "metadata": {
        "name": "多case标签变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "multiple_case_labels",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following switch with multiple case labels where day is 6, what is the final value of day_type?",
    }
}
```

```
        "code": "int day = 6;\nchar *day_type;\nswitch (day) {\n    case 1:\n    case 2:\n    case 3:\n    case 4:\n    case 5:\n        day_type = \"weekday\";\n        break;\n    case 6:\n    case 7:\n        day_type = \"weekend\";\n        break;\n    default:\n        day_type = \"Invalid\";\n}\nprintf(\"Day type: %s\\n\", day_type);",
        "answer": "Weekend"
    },
},
{
    "id": "BL-CD-S004-V012",
    "metadata": {
        "name": "宏定义case变式",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "macro_case",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following switch with macro cases where cmd is READ_CMD (value 1), what is the final value of action?",
        "code": "#define READ_CMD 1\n#define WRITE_CMD 2\n#define DELETE_CMD 3\nint cmd = READ_CMD;\nchar *action;\nswitch (cmd) {\n    case READ_CMD: action = \"Reading\";\n    break;\n    case WRITE_CMD: action = \"Writing\";\n    break;\n    case DELETE_CMD: action = \"Deleting\";\n    break;\n    default: action = \"Unknown\";\n}\nprintf(\"Action: %s\\n\", action);",
        "answer": "Reading"
    }
},
{
    "id": "BL-CD-S005-V001",
    "metadata": {
        "name": "fallthrough起始case",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "fallthrough_start",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following switch block with fallthrough where input is 1, what is the final value of flags?",
    }
}
```

```
        "code": "int input = 1;\nint flags = 0;\nswitch (input) {\n    case 1:\n        flags |= 0x01;\n    case 2:\n        flags |= 0x02;\n    case 3:\n        flags |= 0x04;\n    break;\n    default:\n        flags = -1;\n}\nprintf(\"Flags: 0x%02X\\n\", flags);",
        "answer": 7
    },
    {
        "id": "BL-CD-S005-V002",
        "metadata": {
            "name": "fallthrough最后case",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "fallthrough_end",
            "condition_pattern": "switch_case",
            "branch_count": "4"
        },
        "task": {
            "description": "Given the following switch block with fallthrough where input is 3, what is the final value of flags?",
            "code": "int input = 3;\nint flags = 0;\nswitch (input) {\n    case 1:\n        flags |= 0x01;\n    case 2:\n        flags |= 0x02;\n    case 3:\n        flags |= 0x04;\n    break;\n    default:\n        flags = -1;\n}\nprintf(\"Flags: 0x%02X\\n\", flags);",
            "answer": 4
        }
    },
    {
        "id": "BL-CD-S005-V003",
        "metadata": {
            "name": "default case fallthrough",
            "category": "Block-Level",
            "subcategory": "Conditional",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "default_fallthrough",
            "condition_pattern": "switch_case",
            "branch_count": "4"
        },
        "task": {
            "description": "Given the following switch block with fallthrough where input is 5, what is the final value of flags?",
            "code": "int input = 5;\nint flags = 0;\nswitch (input) {\n    case 1:\n        flags |= 0x01;\n    case 2:\n        flags |= 0x02;\n    case 3:\n        flags |= 0x04;\n    break;\n    default:\n        flags = -1;\n}\nprintf(\"Flags: 0x%02X\\n\", flags);",
            "answer": -1
        }
    }
]
```

```
},
{
  "id": "BL-CD-S005-V004",
  "metadata": {
    "name": "复杂位运算fallthrough",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "complex_bitwise",
    "condition_pattern": "switch_case",
    "branch_count": "5"
  },
  "task": {
    "description": "Given the following complex bitwise switch where input is 2, what is the final value of flags?",
    "code": "int input = 2;\nint flags = 0x10;\nswitch (input) {\n    case 1:\n        flags |= 0x01;\n        flags &= ~0x10;\n    case 2:\n        flags |= 0x02;\n        flags ^= 0x08;\n    case 3:\n        flags |= 0x04;\n        break;\n    case 4:\n        flags = 0;\n        break;\n    default:\n        flags = -1;\n}\nprintf(\"Flags: 0x%02x\\n\", flags);",
    "answer": 30
  }
},
{
  "id": "BL-CD-S005-V005",
  "metadata": {
    "name": "累加器fallthrough",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "accumulator",
    "condition_pattern": "switch_case",
    "branch_count": "4"
  },
  "task": {
    "description": "Given the following accumulator switch where input is 2, what is the final value of total?",
    "code": "int input = 2;\nint total = 0;\nswitch (input) {\n    case 1:\n        total += 10;\n    case 2:\n        total += 20;\n    case 3:\n        total += 30;\n        break;\n    default:\n        total = 0;\n}\nprintf(\"Total: %d\\n\", total);",
    "answer": 50
  }
},
{
  "id": "BL-CD-S005-V006",

```

```
"metadata": {
    "name": "字符串累积fallthrough",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "string_accumulation",
    "condition_pattern": "switch_case",
    "branch_count": "4"
},
"task": {
    "description": "Given the following string building switch where input is 2, what is the final value of result after strcat operations?",
    "code": "int input = 2;\nchar result[100] = \"\";\nswitch (input) {\n    case 1:\n        strcat(result, \"A\");\n    case 2:\n        strcat(result, \"B\");\n    case 3:\n        strcat(result, \"C\");\n        break;\n    default:\n        strcpy(result, \"INVALID\");\n}\nprintf(\"Result: %s\\n\", result);",
    "answer": "BC"
},
{
    "id": "BL-CD-S005-V007",
    "metadata": {
        "name": "计数器fallthrough",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "counter",
        "condition_pattern": "switch_case",
        "branch_count": "4"
},
    "task": {
        "description": "Given the following counter switch where input is 2, what is the final value of count?",
        "code": "int input = 2;\nint count = 0;\nswitch (input) {\n    case 1:\n        count++;\n    case 2:\n        count++;\n    case 3:\n        count++;\n        break;\n    default:\n        count = -1;\n}\nprintf(\"Count: %d\\n\", count);",
        "answer": 2
},
{
    "id": "BL-CD-S005-V008",
    "metadata": {
        "name": "权限累积fallthrough",
        "category": "Block-Level",
        "subcategory": "Conditional",
```

```
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "permission_accumulation",
        "condition_pattern": "switch_case",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following permission switch where user_level is 2, what is the final value of permissions?",
        "code": "int user_level = 2;\nint permissions = 0;\nswitch (user_level) {\n    case 3: // Admin\n        permissions |= 0x04; // DELETE\n    case 2: // Manager\n        permissions |= 0x02; // WRITE\n    case 1: // User\n        permissions |= 0x01; // READ\n    break;\n    case 0: // Guest\n        permissions = 0;\n    break;\n    default:\n        permissions = -1;\n}\nprintf(\"Permissions: 0x%02X\\n\", permissions);",
        "answer": 3
    }
},
{
    "id": "BL-CD-S005-V009",
    "metadata": {
        "name": "状态修改fallthrough",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "state_modification",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following state modification switch where phase is 2, what is the final value of system_state?",
        "code": "int phase = 2;\nint system_state = 0x00;\nswitch (phase) {\n    case 1:\n        system_state |= 0x10; // INIT\n    case 2:\n        system_state |= 0x20; // READY\n    case 3:\n        system_state |= 0x40; // ACTIVE\n    break;\n    default:\n        system_state = 0xFF; // ERROR\n}\nprintf(\"System state: 0x%02X\\n\", system_state);",
        "answer": 96
    }
},
{
    "id": "BL-CD-S005-V010",
    "metadata": {
        "name": "嵌套循环fallthrough",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
```

```
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "nested_loop",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following nested loop switch where input is 2, what is the final value of sum?",
        "code": "int input = 2;\nint sum = 0;\nint i;\nswitch (input) {\n    case 1:\n        for (i = 0; i < 1; i++) sum += i + 1;\n    case 2:\n        for (i = 0; i < 2; i++) sum += i + 1;\n    case 3:\n        for (i = 0; i < 3; i++) sum += i + 1;\n        break;\n    default:\n        sum = -1;\n}\nprintf(\"Sum: %d\\n\", sum);",
        "answer": 9
    }
},
{
    "id": "BL-CD-S005-V011",
    "metadata": {
        "name": "Python模拟fallthrough",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "language_conversion",
        "condition_pattern": "if_elif_chain",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following Python fallthrough simulation where input is 2, what is the final value of flags?",
        "code": "input_val = 2\nflags = 0\nfallthrough = False\nif input_val == 1 or fallthrough:\n    flags |= 0x01\n    fallthrough = True\nif input_val == 2 or fallthrough:\n    flags |= 0x02\n    fallthrough = True\nif input_val == 3 or fallthrough:\n    flags |= 0x04\n    fallthrough = False\nif input_val not in [1, 2, 3] and not fallthrough:\n    flags = -1\n\nprint(f\"Flags: 0x{flags:02X}\")",
        "answer": 6
    }
},
{
    "id": "BL-CD-S005-V012",
    "metadata": {
        "name": "多重fallthrough路径",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "nested_loop",
        "condition_pattern": "switch_case",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following nested loop switch where input is 2, what is the final value of sum?",
        "code": "int input = 2;\nint sum = 0;\nint i;\nswitch (input) {\n    case 1:\n        for (i = 0; i < 1; i++) sum += i + 1;\n    case 2:\n        for (i = 0; i < 2; i++) sum += i + 1;\n    case 3:\n        for (i = 0; i < 3; i++) sum += i + 1;\n        break;\n    default:\n        sum = -1;\n}\nprintf(\"Sum: %d\\n\", sum);",
        "answer": 9
    }
}
```

```
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "multiple_fallthrough",
        "condition_pattern": "switch_case",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the following multi-path fallthrough switch where input is 2, what is the final value of result?",
        "code": "int input = 2;\nint result = 0;\nswitch (input) {\n    case 1:\n        result += 1;\n    case 2:\n        result += 2;\n        if (result > 2) {\n            result *= 2;\n            break;\n        }\n    case 3:\n        result += 4;\n    case 4:\n        result += 8;\n        break;\n    default:\n        result = -1;\n}\nprintf(\"Result: %d\\n\", result);",
        "answer": 4
    }
},
{
    "id": "BL-CD-S006-V001",
    "metadata": {
        "name": "简化三元运算",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simple_ternary",
        "condition_pattern": "ternary",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following simple ternary where x is 7 and y is 5, what is the final value of result?",
        "code": "int x = 7, y = 5;\nint result = (x > y) ? x : y;\nprintf(\"Result: %d\\n\", result);",
        "answer": 7
    }
},
{
    "id": "BL-CD-S006-V002",
    "metadata": {
        "name": "条件反转三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "condition_inversion",
    }
}
```

```
        "condition_pattern": "ternary",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following inverted ternary where x is 7 and y is 5, what is the final value of result?",
        "code": "int x = 7, y = 5;\nint result = (x <= y) ? x * 2 : x + 10;\nprintf(\"Result: %d\\n\", result);",
        "answer": 17
    }
},
{
    "id": "BL-CD-S006-V003",
    "metadata": {
        "name": "四层嵌套三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "deep_nesting",
        "condition_pattern": "ternary",
        "branch_count": "8"
    },
    "task": {
        "description": "Given the following four-level nested ternary where x is 7 and y is 5, what is the final value of result?",
        "code": "int x = 7, y = 5;\nint result = (x > y) ? \n    ((x % 2 == 0) ? \n        ((x > 10) ? x / 2 : x * 2) : \n            ((x > 8) ? x - 3 : x + 10)) : \n                ((y % 2 == 0) ? \n                    ((y > 3) ? y * 3 : y + 5) : \n                        ((y > 6) ? y - 2 : y - 2));\nprintf(\"Result: %d\\n\", result);",
        "answer": 17
    }
},
{
    "id": "BL-CD-S006-V004",
    "metadata": {
        "name": "函数调用三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "function_calls",
        "condition_pattern": "ternary",
        "branch_count": "4"
    },
    "task": {
```

```
        "description": "Given the following function call ternary where x is 7, y is 5, and abs() returns absolute value, what is the final value of result?",  
        "code": "int x = 7, y = 5;\nint result = (x > y) ? \n    ((abs(x - y) > 1) ?  
    max(x, y) + abs(x - y) : min(x, y)) : \n    ((abs(y - x) > 1) ? max(x, y) - abs(y - x) :  
    min(x, y));\nprintf(\"Result: %d\\n\", result);",  
        "answer": 9  
    }  
,  
{  
    "id": "BL-CD-S006-V005",  
    "metadata": {  
        "name": "位运算三元",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "hard",  
        "intervention": 2,  
        "variant_type": "bitwise_operations",  
        "condition_pattern": "ternary",  
        "branch_count": "4"  
    },  
    "task": {  
        "description": "Given the following bitwise ternary where x is 7 and y is 5, what is the final value of result?",  
        "code": "int x = 7, y = 5;\nint result = (x & 1) ? \n    ((y & 1) ? (x | y) : (x  
    ^ y)) : \n    ((y & 1) ? (x & y) : (x << 1));\nprintf(\"Result: %d\\n\", result);",  
        "answer": 7  
    }  
,  
{  
    "id": "BL-CD-S006-V006",  
    "metadata": {  
        "name": "Python版本三元",  
        "category": "Block-Level",  
        "subcategory": "Conditional",  
        "type": "variant",  
        "source": "Generated",  
        "language": "python",  
        "difficulty": "medium",  
        "intervention": 1,  
        "variant_type": "language_conversion",  
        "condition_pattern": "ternary",  
        "branch_count": "4"  
    },  
    "task": {  
        "description": "Given the following Python ternary where x is 7 and y is 5, what is the final value of result?",  
        "code": "x, y = 7, 5\nresult = (x * 2 if x % 2 == 0 else x + 10) if x > y else  
(y * 3 if y % 2 == 0 else y - 2)\nprint(f\"Result: {result}\")",  
        "answer": 17  
    }  
}
```

```
},
{
  "id": "BL-CD-S006-V007",
  "metadata": {
    "name": "数组访问三元",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "array_access",
    "condition_pattern": "ternary",
    "branch_count": "4"
  },
  "task": {
    "description": "Given the following array access ternary where x is 7, y is 5, and arr = {1,2,3,4,5,6,7,8}, what is the final value of result?",
    "code": "int x = 7, y = 5;\nint arr[] = {1, 2, 3, 4, 5, 6, 7, 8};\nint result =\n(x > y) ? \n    ((x < 8) ? arr[x-1] : arr[0]) : \n    ((y < 8) ? arr[y-1] :\narr[0]);\nprintf(\"Result: %d\\n\", result);",
    "answer": 7
  }
},
{
  "id": "BL-CD-S006-V008",
  "metadata": {
    "name": "指针操作三元",
    "category": "Block-Level",
    "subcategory": "Conditional",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "pointer_operations",
    "condition_pattern": "ternary",
    "branch_count": "4"
  },
  "task": {
    "description": "Given the following pointer ternary where x is 7, y is 5, and ptr points to x, what is the final value of result?",
    "code": "int x = 7, y = 5;\nint *ptr = &x;\nint result = (ptr != NULL) ? \n    ((*ptr > y) ? (*ptr + *ptr) : (*ptr - y)) : \n    ((y > 0) ? y * 2 : 0);\nprintf(\"Result: %d\\n\", result);",
    "answer": 14
  }
},
{
  "id": "BL-CD-S006-V009",
  "metadata": {
    "name": "类型转换三元",
  }
}
```

```
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "type_casting",
        "condition_pattern": "ternary",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following type casting ternary where x is 7 and y is 5, what is the final value of result?",
        "code": "int x = 7, y = 5;\ndouble result = (x > y) ? \n    ((x % 2 == 0) ?\n(double)x / 2.0 : (double)x + 10.5) : \n    ((y % 2 == 0) ? (double)y * 3.0 : (double)y -\n2.5);\nprintf(\"Result: %.1f\\n\", result);",
        "answer": 17.5
    }
},
{
    "id": "BL-CD-S006-V010",
    "metadata": {
        "name": "短路逻辑三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "short_circuit",
        "condition_pattern": "ternary",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following short-circuit ternary where x is 7, y is 5, and ptr is NULL, what is the final value of result?",
        "code": "int x = 7, y = 5;\nint *ptr = NULL;\nint result = (ptr != NULL && *ptr > 0) ? \n    ((x % 2 == 0) ? x * 2 : x + 10) : \n    ((y > 0 && y < 10) ? y + x : 0);\nprintf(\"Result: %d\\n\", result);",
        "answer": 12
    }
},
{
    "id": "BL-CD-S006-V011",
    "metadata": {
        "name": "宏定义三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
    }
}
```

```
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "macro_usage",
        "condition_pattern": "ternary",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following macro ternary where x is 7 and y is 5, what is the final value of result?",
        "code": "#define MAX(a,b) ((a) > (b) ? (a) : (b))\n#define MIN(a,b) ((a) < (b) ? (a) : (b))\nint x = 7, y = 5;\nint result = (x > y) ? \n    ((x % 2 == 0) ? MAX(x, 10) : MIN(x + 10, 20)) : \n    ((y % 2 == 0) ? MAX(y, 8) : MIN(y + 5, 12));\nprintf(\"Result: %d\\n\", result);",
        "answer": 17
    }
},
{
    "id": "BL-CD-S006-v012",
    "metadata": {
        "name": "结构体字段三元",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "struct_fields",
        "condition_pattern": "ternary",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following struct field ternary where point.x is 7 and point.y is 5, what is the final value of result?",
        "code": "struct Point { int x, y; };\nstruct Point point = {7, 5};\nint result = (point.x > point.y) ? \n    ((point.x % 2 == 0) ? point.x * point.y : point.x + point.y + 5) : \n    ((point.y % 2 == 0) ? point.y * 3 : point.y - 2);\nprintf(\"Result: %d\\n\", result);",
        "answer": 17
    }
},
{
    "id": "BL-CD-S007-v001",
    "metadata": {
        "name": "malloc失败检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
    }
}
```

```
        "variant_type": "null_failure",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following null check block where malloc fails and returns NULL, what is the return value?",
        "code": "ssh_poll_handle ssh_poll_new(socket_t fd, short events) {\n    ssh_poll_handle p;\n    p = malloc(sizeof(struct ssh_poll_handle_struct));\n    if (p == NULL) {\n        return NULL;\n    }\n    p->x.fd = fd;\n    p->events = events;\n    return p;\n}",
        "answer": "NULL"
    }
},
{
    "id": "BL-CD-S007-V002",
    "metadata": {
        "name": "多重指针检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "multiple_null_checks",
        "condition_pattern": "complex_if",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following multiple null checks where both malloc calls succeed, what is the final value of result?",
        "code": "int process_data(void) {\n    char *buffer = malloc(100);\n    char *temp = malloc(50);\n    int result;\n\n    if (buffer == NULL || temp == NULL) {\n        if (buffer) free(buffer);\n        if (temp) free(temp);\n        return -1;\n    }\n\n    // Process data\n    result = 42;\n    free(buffer);\n    free(temp);\n    return result;\n}",
        "answer": 42
    }
},
{
    "id": "BL-CD-S007-V003",
    "metadata": {
        "name": "嵌套内存分配",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "nested_allocation",
        "condition_pattern": "nested_if",
    }
}
```

```
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following nested allocation where first malloc succeeds but second fails, what is the return value?",
        "code": "struct Node* create_list(int size) {\n    struct Node* head =\n    malloc(sizeof(struct Node));\n    if (head == NULL) {\n        return NULL;\n    }\n    head->data = malloc(size * sizeof(int));\n    if (head->data == NULL) {\n        free(head);\n        return NULL;\n    }\n    head->next = NULL;\n    return head;\n}",
        "answer": "NULL"
    }
},
{
    "id": "BL-CD-S007-v004",
    "metadata": {
        "name": "文件指针检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "file_pointer_check",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {
        "description": "Given the following file pointer check where fopen succeeds, what is the return value?",
        "code": "int read_config(const char* filename) {\n    FILE* fp = fopen(filename,\n    \"r\");\n    if (fp == NULL) {\n        return -1;\n    }\n    // Read configuration\n    int config_value = 100;\n    fclose(fp);\n    return config_value;\n}",
        "answer": 100
    }
},
{
    "id": "BL-CD-S007-v005",
    "metadata": {
        "name": "Python None检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "language_conversion",
        "condition_pattern": "simple_if",
        "branch_count": "2"
    },
    "task": {
```

```
        "description": "Given the following None check where data is not None, what is the return value?",  
        "code": "def process_data(data):\n            if data is None:\n                return None\n            result = {\n                'processed': True,\n                'value': data * 2\n            }\n            return result",  
        "answer": "{'processed': True, 'value': 10}"  
    },  
    {  
        "id": "BL-CD-S007-V006",  
        "metadata": {  
            "name": "双重检查模式",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "c",  
            "difficulty": "hard",  
            "intervention": 2,  
            "variant_type": "double_check",  
            "condition_pattern": "nested_if",  
            "branch_count": "3"  
        },  
        "task": {  
            "description": "Given the following double-check pattern where ptr is initially NULL and malloc succeeds, what is the final state of ptr?",  
            "code": "static void* shared_ptr = NULL;\nvoid* get_shared_resource(void) {\nif (shared_ptr == NULL) {\n    shared_ptr = malloc(1024);\n    if (shared_ptr ==\nNULL) {\n        return NULL;\n    }\n    // Initialize resource\n    memset(shared_ptr, 0, 1024);\n}\n    return shared_ptr;\n}",  
            "answer": "valid_pointer"  
        },  
    },  
    {  
        "id": "BL-CD-S007-V007",  
        "metadata": {  
            "name": "错误码检查",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "c",  
            "difficulty": "medium",  
            "intervention": 1,  
            "variant_type": "error_code_check",  
            "condition_pattern": "simple_if",  
            "branch_count": "2"  
        },  
        "task": {  
            "description": "Given the following error code check where init_system() returns 0 (success), what is the return value?",  
        }  
    }  
}
```

```
        "code": "int initialize_application(void) {\n    int result = init_system();\n\n    if (result != 0) {\n        return result;\n    }\n\n    // Continue initialization\n\n    return setup_complete();\n}",
        "answer": "setup_complete_return_value"
    },
},
{
    "id": "BL-CD-S007-V008",
    "metadata": {
        "name": "资源链式检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "resource_chain",
        "condition_pattern": "complex_if",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following resource chain where all allocations succeed, what is the return value?",
        "code": "int setup_resources(void) {\n    void *mem1 = malloc(100);\n    void *mem2 = malloc(200);\n    void *mem3 = malloc(300);\n\n    if (!mem1 || !mem2 || !mem3)\n        if (mem1) free(mem1);\n        if (mem2) free(mem2);\n        if (mem3)\n            free(mem3);\n\n    return -1;\n}\n\n// Setup successful\n\n    global_mem1 = mem1;\n    global_mem2 = mem2;\n    global_mem3 = mem3;\n\n    return 0;\n}",
        "answer": 0
    },
},
{
    "id": "BL-CD-S007-V009",
    "metadata": {
        "name": "条件释放检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "conditional_free",
        "condition_pattern": "complex_if",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following conditional free where ptr is valid and should_free is 1, what happens to ptr?",
        "code": "void cleanup_resource(void** ptr, int should_free) {\n    if (ptr ==\n        NULL || *ptr == NULL) {\n        return;\n    }\n\n    if (should_free) {\n        free(*ptr);\n        *ptr = NULL;\n    }\n}",
        "answer": 0
    }
}
```

```
        "answer": "freed_and_nulled"
    },
},
{
    "id": "BL-CD-S007-V010",
    "metadata": {
        "name": "RAII模式检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "raii_pattern",
        "condition_pattern": "complex_if",
        "branch_count": "4"
    },
    "task": {
        "description": "Given the following RAII-style pattern where resource acquisition succeeds, what is the final state?",
        "code": "typedef struct {\n    void* data;\n    int (*cleanup)(void*);\n} Resource;\n\nResource* acquire_resource(size_t size) {\n    Resource* res =\n    malloc(sizeof(Resource));\n    if (res == NULL) {\n        return NULL;\n    }\n    res->data = malloc(size);\n    if (res->data == NULL) {\n        free(res);\n        return\n    }\n    res->cleanup = free;\n    return res;\n}",
        "answer": "valid_resource_with_data"
    }
},
{
    "id": "BL-CD-S007-V011",
    "metadata": {
        "name": "智能指针模拟",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "smart_pointer",
        "condition_pattern": "complex_if",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following smart pointer simulation where ref_count starts at 0, what is the final ref_count after creation?",
        "code": "typedef struct {\n    void* ptr;\n    int ref_count;\n} SmartPtr;\n\nSmartPtr* create_smart_ptr(size_t size) {\n    SmartPtr* smart =\n    malloc(sizeof(SmartPtr));\n    if (smart == NULL) {\n        return NULL;\n    }\n    smart->ptr = malloc(size);\n    if (smart->ptr == NULL) {\n        free(smart);\n        return\n    }\n    smart->ref_count = 1;\n    return smart;\n}",
        "answer": 1
    }
}
```

```
        }
    },
{
    "id": "BL-CD-S007-V012",
    "metadata": {
        "name": "延迟初始化检查",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "lazy_initialization",
        "condition_pattern": "nested_if",
        "branch_count": "3"
    },
    "task": {
        "description": "Given the following lazy initialization where global_resource is NULL and init succeeds, what is the return value?",
        "code": "static void* global_resource = NULL;\n\nvoid* get_global_resource(void)\n{\n    if (global_resource == NULL) {\n        global_resource =\nmalloc(sizeof(ResourceStruct));\n        if (global_resource == NULL) {\n            return\nNULL;\n        }\n        if (init_resource(global_resource) != 0) {\n            free(global_resource);\n            global_resource = NULL;\n            return\nNULL;\n        }\n    }\n    return\n    global_resource;\n}",
        "answer": "valid_initialized_resource"
    }
},
{
    "id": "BL-CD-S008-V001",
    "metadata": {
        "name": "最低年龄测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "age_boundary",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following complex conditional logic where age is 18, income is 50000, and has_degree is True, what is the final value of status?"
    }
}
```

```
        "code": "age = 18\nincome = 50000\nhas_degree = True\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    else:\n        if age >= 21 and has_degree:\n            status = \"approved_basic\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_premium"
    },
},
{
    "id": "BL-CD-S008-V002",
    "metadata": {
        "name": "低收入高学历",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "low_income_degree",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the following complex conditional logic where age is 25, income is 25000, and has_degree is True, what is the final value of status?",
        "code": "age = 25\nincome = 25000\nhas_degree = True\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    else:\n        if age >= 21 and has_degree:\n            status = \"approved_basic\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_basic"
    },
},
{
    "id": "BL-CD-S008-V003",
    "metadata": {
        "name": "年龄过高测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "age_overflow",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    },
    "task": {
```

```
        "description": "Given the following complex conditional logic where age is 70, income is 50000, and has_degree is True, what is the final value of status?",  
        "code": "age = 70\nincome = 50000\nhas_degree = True\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            if age >= 21 and has_degree:\n                status = \"approved_standard\"\n            else:\n                status = \"approved_basic\"\n    else:\n        status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",  
        "answer": "not_eligible"  
    },  
    {  
        "id": "BL-CD-S008-V004",  
        "metadata": {  
            "name": "年轻无学历低收入",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "python",  
            "difficulty": "medium",  
            "intervention": 1,  
            "variant_type": "young_no_degree_low_income",  
            "condition_pattern": "complex_logic",  
            "branch_count": "5"  
        },  
        "task": {  
            "description": "Given the following complex conditional logic where age is 20, income is 25000, and has_degree is False, what is the final value of status?",  
            "code": "age = 20\nincome = 25000\nhas_degree = False\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            if age >= 21 and has_degree:\n                status = \"approved_standard\"\n            else:\n                status = \"approved_basic\"\n    else:\n        status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",  
            "answer": "pending_review"  
        },  
    },  
    {  
        "id": "BL-CD-S008-V005",  
        "metadata": {  
            "name": "C语言版本转换",  
            "category": "Block-Level",  
            "subcategory": "Conditional",  
            "type": "variant",  
            "source": "Generated",  
            "language": "c",  
            "difficulty": "hard",  
            "intervention": 2,  
            "variant_type": "language_conversion",  
            "condition_pattern": "complex_logic",  
            "branch_count": "5"  
        },  
    },
```

```
"task": {
    "description": "Given the following C version where age is 25, income is 50000, and has_degree is 1, what is the final value of status_code?",
    "code": "int age = 25;\nint income = 50000;\nint has_degree = 1;\nint\nstatus_code;\n\nif (age >= 18 && age <= 65) {\n    if (income > 30000) {\n        if\n        (has_degree) {\n            status_code = 1; // approved_premium\n        } else {\n            status_code = 2; // approved_standard\n        }\n    } else {\n        if (age >= 21 &&\n        has_degree) {\n            status_code = 3; // approved_basic\n        } else {\n            status_code = 4; // pending_review\n        }\n    }\n} else {\n    status_code = 0; //\n    not_eligible\n}\n\nprintf(\"status code: %d\\n\", status_code);",
    "answer": 1
},
{
    "id": "BL-CD-S008-V006",
    "metadata": {
        "name": "扩展条件系统",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "extended_conditions",
        "condition_pattern": "complex_logic",
        "branch_count": "8"
    },
    "task": {
        "description": "Given the extended system where age is 25, income is 50000, has_degree is True, and credit_score is 750, what is the final value of status?",
        "code": "age = 25\nincome = 50000\nhas_degree = True\ncredit_score = 750\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 50000 and credit_score >= 700:\n        if has_degree:\n            status = \"approved_platinum\"\n        else:\n            status = \"approved_gold\"\n    elif income > 30000:\n        if has_degree and credit_score >= 650:\n            status = \"approved_premium\"\n        elif has_degree:\n            status = \"approved_standard\"\n        else:\n            status = \"approved_basic\"\n    else:\n        if age >= 21 and has_degree and credit_score >= 600:\n            status = \"approved_conditional\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_platinum"
    }
},
{
    "id": "BL-CD-S008-V007",
    "metadata": {
        "name": "短路逻辑测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
    }
}
```

```
        "intervention": 2,
        "variant_type": "short_circuit_logic",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the short-circuit logic where age is None, income is 50000, and has_degree is True, what is the final value of status?",
        "code": "age = None\nincome = 50000\nhas_degree = True\nstatus = None\n\nif age is not None and 18 <= age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    else:\n        if age >= 21 and has_degree:\n            status = \"approved_basic\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "not_eligible"
    }
},
{
    "id": "BL-CD-S008-V008",
    "metadata": {
        "name": "边界值组合测试",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "boundary_combinations",
        "condition_pattern": "complex_logic",
        "branch_count": "5"
    },
    "task": {
        "description": "Given the boundary test where age is 21, income is 30000, and has_degree is True, what is the final value of status?",
        "code": "age = 21\nincome = 30000\nhas_degree = True\nstatus = None\n\nif age >= 18 and age <= 65:\n    if income > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            if age >= 21 and has_degree:\n                status = \"approved_basic\"\n            else:\n                status = \"pending_review\"\n    else:\n        status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_basic"
    }
},
{
    "id": "BL-CD-S008-V009",
    "metadata": {
        "name": "函数调用条件",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
```

```

        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "function_call_conditions",
        "condition_pattern": "complex_logic",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the function-based conditions where get_age() returns 25, get_income() returns 50000, and has_valid_degree() returns True, what is the final value of status?",
        "code": "def get_age(): return 25\n\ndef get_income(): return 50000\n\ndef has_valid_degree(): return True\n\ndef get_experience(): return 3\n\nstatus = None\n\nif 18 <= get_age() <= 65:\n    if get_income() > 30000:\n        if has_valid_degree() and get_experience() >= 2:\n            status = \"approved_premium_plus\"\n        elif has_valid_degree():\n            status = \"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    else:\n        if get_age() >= 21 and has_valid_degree():\n            status = \"approved_basic\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_premium_plus"
    }
},
{
    "id": "BL-CD-S008-V010",
    "metadata": {
        "name": "异常处理条件",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "exception_handling",
        "condition_pattern": "complex_logic",
        "branch_count": "6"
    },
    "task": {
        "description": "Given the exception handling where age is 25, income is 'invalid', and has_degree is True, what is the final value of status?",
        "code": "age = 25\n\nincome = 'invalid'\n\nhas_degree = True\n\nstatus = None\n\ntry:\n    income_val = int(income)\nexcept (ValueError, TypeError):\n    income_val = 0\n\nif isinstance(age, int) and 18 <= age <= 65:\n    if income_val > 30000:\n        if has_degree:\n            status = \"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    else:\n        if age >= 21 and has_degree:\n            status = \"approved_basic\"\n        else:\n            status = \"pending_review\"\nelse:\n    status = \"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_basic"
    }
},
{
    "id": "BL-CD-S008-V011",
    "metadata": {

```

```
        "name": "多重验证条件",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "multiple_validations",
        "condition_pattern": "complex_logic",
        "branch_count": "7"
    },
    "task": {
        "description": "Given the multi-validation system where all inputs are valid and age is 25, income is 50000, has_degree is True, what is the final value of status?",
        "code": "age = 25\nincome = 50000\nhas_degree = True\nemployment_status =\n'full_time'\nstatus = None\n\n# Input validation\nif not isinstance(age, int) or not\nisinstance(income, int):\n    status = \"invalid_input\"\nelif age < 0 or income < 0:\n    status = \"invalid_values\"\nelif 18 <= age <= 65:\n    if income > 30000 and\nemployment_status == 'full_time':\n        if has_degree:\n            status =\n\"approved_premium\"\n        else:\n            status = \"approved_standard\"\n    elif\nincome > 20000:\n        if age >= 21 and has_degree:\n            status =\n\"approved_basic\"\n        else:\n            status = \"pending_review\"\n    else:\n        status = \"insufficient_income\"\nelse:\n    status =\n\"not_eligible\"\n\nprint(f\"Status: {status}\")",
        "answer": "approved_premium"
    }
},
{
    "id": "BL-CD-S008-V012",
    "metadata": {
        "name": "状态机条件系统",
        "category": "Block-Level",
        "subcategory": "Conditional",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "state_machine",
        "condition_pattern": "complex_logic",
        "branch_count": "8"
    },
    "task": {
        "description": "Given the state machine where current_state is 'initial', age is 25, income is 50000, and has_degree is True, what is the final value of status?"
    }
}
```

```

    "code": "age = 25\nincome = 50000\nhas_degree = True\ncurrent_state =\n'initial'\nstatus = None\nif current_state == 'initial':\n    if 18 <= age <= 65:\n        current_state = 'age_verified'\n        status = \"not_eligible\"\n    else:\n        current_state = 'rejected'\n        if income > 30000:\n            current_state = 'income_verified'\n            if income > 15000:\n                current_state = 'income_low'\n            else:\n                status = \"insufficient_income\"\n            if has_degree:\n                status = \"approved_premium\"\n            else:\n                status = \"approved_standard\"\n        elif current_state == 'income_low':\n            if age >= 21 and has_degree:\n                status = \"approved_basic\"\n            else:\n                status = \"pending_review\"\n        print(f\"Status: {status}\")",
    "answer": "approved_premium"
}
}
]

```

2C - 迭代代码块 [Iterative] (115)

迭代代码块推理变式生成提示词

任务目标

基于给定的迭代代码块种子任务，生成多样化的变式任务来全面测试大模型对循环结构、迭代过程、状态累积和终止条件的理解能力，包括各种循环类型、嵌套迭代、控制流跳转和递归调用等复杂迭代场景。

迭代代码块特征分析

迭代代码块是指包含**for**、**while**、**do-while**循环以及递归调用等重复执行结构的代码段，重点测试模型对迭代次数计算、状态变化追踪、终止条件判断和累积结果预测的理解。

变式生成维度

1. 循环结构类型变式

- **for循环变式**：传统的计数循环和范围循环
- **while循环变式**：条件驱动的前置判断循环
- **do-while循环变式**：后置判断的至少执行一次循环
- **foreach循环变式**：基于集合遍历的循环
- **递归循环变式**：函数递归调用形成的迭代
- **goto循环变式**：使用**goto**构造的循环结构

2. 迭代模式变式

- **计数迭代变式**：基于计数器的固定次数迭代
- **条件迭代变式**：基于条件判断的动态次数迭代
- **遍历迭代变式**：遍历数组、列表、字符串等数据结构
- **搜索迭代变式**：查找特定元素或条件的迭代
- **累积迭代变式**：逐步累积计算结果的迭代
- **变换迭代变式**：逐步变换数据状态的迭代

3. 终止条件变式

- **计数终止变式**：基于迭代次数的终止条件
- **数值终止变式**：基于数值达到阈值的终止
- **状态终止变式**：基于程序状态变化的终止
- **外部终止变式**：基于外部输入或事件的终止
- **异常终止变式**：因异常或错误导致的提前终止
- **复合终止变式**：多个条件组合的复杂终止判断

4. 嵌套和组合变式

- **单层循环变式**: 简单的单层迭代结构
- **双层嵌套变式**: 两层嵌套的循环结构
- **多层嵌套变式**: 三层及以上的深度嵌套
- **循环组合变式**: 多个独立循环的顺序组合
- **交错循环变式**: 循环之间有交互和依赖关系
- **递归嵌套变式**: 递归调用中包含循环结构

5. 控制流跳转变式

- **break跳出变式**: 使用`break`提前退出循环
- **continue跳过变式**: 使用`continue`跳过当前迭代
- **return返回变式**: 在循环中使用`return`直接返回
- **goto跳转变式**: 使用`goto`跳转改变循环流程
- **异常跳出变式**: 通过异常机制跳出循环
- **多层跳出变式**: 从深层嵌套中跳出到外层

6. 状态累积变式

- **数值累积变式**: 数值的加法、乘法等累积运算
- **字符串累积变式**: 字符串的拼接和构建
- **数组累积变式**: 向数组或列表添加元素
- **对象累积变式**: 对象属性的逐步设置和修改
- **状态机变式**: 状态在迭代中的转换和更新
- **资源累积变式**: 内存、文件等资源的逐步分配

7. 边界和特殊情况变式

- **空迭代变式**: 迭代次数为零的边界情况
- **单次迭代变式**: 只执行一次的迭代
- **无限循环变式**: 理论上的无限循环及其检测
- **溢出变式**: 迭代过程中的数值溢出情况
- **内存限制变式**: 受内存限制的迭代过程
- **性能边界变式**: 大数据量的迭代性能考虑

8. 递归模式变式

- **线性递归变式**: 简单的线性递归结构
- **分支递归变式**: 多分支的递归调用
- **尾递归变式**: 尾递归优化的递归模式
- **相互递归变式**: 函数间的相互递归调用
- **间接递归变式**: 通过中间函数的间接递归
- **递归终止变式**: 递归的基础情况和终止条件

9. 数据结构遍历变式

- **数组遍历变式**: 一维和多维数组的遍历
- **链表遍历变式**: 单链表、双链表的遍历
- **树遍历变式**: 二叉树、多叉树的遍历
- **图遍历变式**: 图结构的深度和广度遍历
- **哈希表遍历变式**: 哈希表键值对的遍历
- **字符串遍历变式**: 字符串字符的逐个处理

10. 并发和同步变式

- **并行迭代变式**: 多线程并行执行的迭代
- **同步迭代变式**: 需要同步的多线程迭代
- **原子操作变式**: 迭代中的原子操作累积
- **锁保护变式**: 使用锁保护的共享状态迭代

- **无锁迭代变式**: 无锁数据结构的迭代访问
- **分布式迭代变式**: 分布式环境下的迭代处理

复杂度层次设计

简单迭代 (Easy)

- 单层`for/while`循环, 固定次数迭代
- 简单的计数或累积操作
- 明确的终止条件, 无控制流跳转
- 基础数据类型的简单操作

中等迭代 (Medium)

- 2层嵌套循环或包含简单条件的迭代
- 涉及`break/continue`的控制流
- 数组或字符串的遍历处理
- 简单的递归调用

复杂迭代 (Hard)

- 3层及以上嵌套或复杂的迭代逻辑
- 复杂的终止条件和状态管理
- 多重控制流跳转和异常处理
- 复杂数据结构的遍历

专家级迭代 (Expert)

- 极度复杂的嵌套和递归结构
- 涉及高级算法和数据结构
- 并发和同步的迭代处理
- 性能关键的迭代优化

生成策略

种子分析策略

1. **识别迭代模式**: 分析种子的循环类型和迭代特征
2. **提取核心逻辑**: 识别迭代的核心计算和状态变化
3. **分析终止条件**: 理解循环的终止机制和边界条件
4. **追踪状态变化**: 跟踪变量在迭代过程中的状态演变

变式设计原则

1. **迭代完整性**: 确保能够正确模拟完整的迭代过程
2. **状态一致性**: 保证每次迭代的状态变化逻辑正确
3. **边界准确性**: 重点测试边界条件和特殊情况
4. **性能意识**: 考虑迭代的时间和空间复杂度

质量保证

1. **迭代验证**: 手工验证关键迭代步骤的正确性
2. **边界测试**: 确保边界情况得到正确处理
3. **终止验证**: 验证所有循环都能正确终止
4. **状态追踪**: 确保状态变化的准确跟踪

输出格式要求

```
```json
[
```

```

{
 "id": "BL-IT-S00X-V001",
 "metadata": {
 "name": "迭代代码块变式名称",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "目标语言",
 "difficulty": "easy/medium/hard/expert",
 "intervention": 0,
 "variant_type": "变式类型标签",
 "loop_pattern": "for/while/do_while/foreach/recursive",
 "nesting_depth": "嵌套层数",
 "iteration_count": "预期迭代次数"
 },
 "task": {
 "description": "迭代代码块推理任务描述",
 "code": "包含迭代结构的代码块",
 "answer": "迭代执行后的最终结果"
 }
},
{下一个变式...}
]

```

特殊字段说明

**loop\_pattern:** 标识主要的循环模式类型

**nesting\_depth:** 记录循环嵌套的深度

**iteration\_count:** 预期的迭代执行次数

**variant\_type:** 描述具体的变式类型

生成目标

为每个提供的种子任务生成12-18个迭代代码块变式，确保：

每种难度等级至少包含3-4个变式

涵盖至少6-8种不同的循环模式

包含不同嵌套深度的变式（1-4层）

至少包含3-4个包含控制流跳转的变式

包含2-3个递归相关的变式

重点测试场景

迭代次数计算：给定初始条件，准确计算循环执行次数

状态变化追踪：正确追踪变量在迭代过程中的状态变化

终止条件判断：准确判断循环的终止时机和条件

嵌套循环理解：对多层嵌套循环的正确理解和计算

控制流处理：对**break**、**continue**等控制流的正确处理

递归深度理解：对递归调用深度和返回值的正确计算

常见迭代计算模式

累加求和： $\sum_{i=1}^n i = n(n+1)/2$

阶乘计算： $n! = n \times (n-1)!$

斐波那契数列： $F(n) = F(n-1) + F(n-2)$

幂次计算： $a^n$  的迭代计算

最大公约数：欧几里得算法的迭代实现

素数判断：试除法的迭代实现

请基于此提示词，为给定的迭代代码块种子任务生成全面的变式集合，输出格式为包含所有变式的JSON数组。

```
[
 {
 "id": "BL-IT-S001-V001",
 "metadata": {
 "name": "简单for循环反向计数",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "reverse_counting",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following reverse for loop block, what is the final value of product?",
 "code": "int product = 1;\nfor (int i = 5; i >= 1; i--) {\n product *= i;\n}\nprintf(\"Product: %d\\n\", product);",
 "answer": 120
 }
 },
 {
 "id": "BL-IT-S001-V002",
 "metadata": {
 "name": "步长为2的for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "step_increment",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following for loop with step 2, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 6; i += 2) {\n sum += i *\n i;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 35
 }
 },
 {
 "id": "BL-IT-S001-V003",
 "metadata": {
 "name": "累积乘法运算",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "cumulative_product",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following for loop block, what is the final value of product?",
 "code": "int product = 1;\nfor (int i = 1; i <= 5; i++) {\n product *= i;\n}\nprintf(\"Product: %d\\n\", product);",
 "answer": 120
 }
 }]
```

```
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "multiplicative_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following for loop with multiplicative accumulation, what is the final value of result?",
 "code": "int result = 2;\nfor (int i = 1; i <= 4; i++) {\n result *= (i + 1);\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 240
 }
},
{
 "id": "BL-IT-S001-V004",
 "metadata": {
 "name": "条件累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following for loop with conditional accumulation, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 10; i++) {\n if (i % 3 == 0) {\n sum += i;\n }\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 18
 }
},
{
 "id": "BL-IT-S001-V005",
 "metadata": {
 "name": "负数范围for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
```

```
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "negative_range",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following for loop with negative range, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = -2; i <= 2; i++) {\n sum += i *\ni;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 10
 }
},
{
 "id": "BL-IT-S001-V006",
 "metadata": {
 "name": "浮点数累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "float_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following for loop with float accumulation, what is the final value of sum (rounded to 1 decimal)?",
 "code": "float sum = 0.0;\nfor (int i = 1; i <= 4; i++) {\n sum += 1.0 /\ni;\n}\nprintf(\"Sum: %.1f\\n\", sum);",
 "answer": 2.1
 }
},
{
 "id": "BL-IT-S001-V007",
 "metadata": {
 "name": "字符ASCII累加",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "char_ascii_sum",
 "loop_pattern": "for",
 "nesting_depth": 1,
```

```
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following for loop with character ASCII accumulation, what is the final value of sum?",
 "code": "int sum = 0;\nfor (char c = 'A'; c <= 'E'; c++) {\n sum +=\n (int)c;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 335
 }
},
{
 "id": "BL-IT-S001-V008",
 "metadata": {
 "name": "复合赋值运算for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "compound_assignment",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following for loop with compound assignment, what is the final value of result?",
 "code": "int result = 10;\nfor (int i = 1; i <= 4; i++) {\n result += i;\n result *= 2;\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 120
 }
},
{
 "id": "BL-IT-S001-V009",
 "metadata": {
 "name": "位运算累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "bitwise_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following for loop with bitwise operations, what is the final value of result?",
 "code": "int result = 0;\nfor (int i = 1; i <= 4; i++) {\n result |= i;\n}\nprintf(\"Result: %d\\n\", result);"
 }
}
```

```
 "code": "int result = 0;\nfor (int i = 1; i <= 4; i++) {\n result ^= (1 << i);\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 30
 }
},
{
 "id": "BL-IT-S001-V010",
 "metadata": {
 "name": "数组元素累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "array_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following for loop with array element accumulation, what is the final value of sum?",
 "code": "int arr[] = {2, 4, 6, 8, 10};\nint sum = 0;\nfor (int i = 0; i < 5; i++) {\n sum += arr[i];\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 30
 }
},
{
 "id": "BL-IT-S001-V011",
 "metadata": {
 "name": "最大值查找for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "max_finding",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following for loop to find maximum value, what is the final value of max?",
 "code": "int arr[] = {3, 7, 2, 9, 1, 5};\nint max = arr[0];\nfor (int i = 1; i < 6; i++) {\n if (arr[i] > max) {\n max = arr[i];\n }\n}\nprintf(\"Max: %d\\n\", max);",
 "answer": 9
 }
}
```

```
},
{
 "id": "BL-IT-S001-V012",
 "metadata": {
 "name": "计数器for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "counting",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following for loop counting positive numbers, what is the final value of count?",
 "code": "int arr[] = {-1, 3, -2, 5, 0, 7, -4, 2};\nint count = 0;\nfor (int i = 0; i < 8; i++) {\n if (arr[i] > 0) {\n count++;\n }\n}\nprintf(\"Count: %d\\n\", count);",
 "answer": 4
 }
},
{
 "id": "BL-IT-S001-V013",
 "metadata": {
 "name": "交替累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "alternating_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following for loop with alternating accumulation, what is the final value of result?",
 "code": "int result = 0;\nfor (int i = 1; i <= 6; i++) {\n if (i % 2 == 1)\n result += i;\n else\n result -= i;\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": -3
 }
},
{
 "id": "BL-IT-S001-V014",
 "metadata": {
 "name": "交替累加for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "alternating_accumulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following for loop with alternating accumulation, what is the final value of result?",
 "code": "int result = 0;\nfor (int i = 1; i <= 6; i++) {\n if (i % 2 == 1)\n result += i;\n else\n result -= i;\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": -3
 }
}
```

```
"metadata": {
 "name": "平方和for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "square_sum",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
},
"task": {
 "description": "Given the following for loop calculating sum of squares, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 4; i++) {\n sum += i * i;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 30
},
{
 "id": "BL-IT-S001-v015",
 "metadata": {
 "name": "边界条件for循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "boundary_condition",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 0
},
 "task": {
 "description": "Given the following for loop with boundary condition, what is the final value of sum?",
 "code": "int sum = 5;\nfor (int i = 1; i < 1; i++) {\n sum += i;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 5
},
{
 "id": "BL-IT-S002-v001",
 "metadata": {
 "name": "字符数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
```

```
"source": "Generated",
"language": "c",
"difficulty": "medium",
"intervention": 1,
"variant_type": "char_array_init",
"loop_pattern": "for",
"nesting_depth": 1,
"iteration_count": 8
},
"task": {
 "description": "Given the following character array initialization loop where BUFFER_SIZE is 8, what is the value of buffer[5] after execution?",
 "code": "char buffer[8];\nfor (int i = 0; i < 8; i++) {\n buffer[i] = 'A' + i;\n}\nprintf(\"buffer[5]: %c\\n\", buffer[5]);",
 "answer": "F"
},
{
 "id": "BL-IT-S002-V002",
 "metadata": {
 "name": "整数数组递增初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "int_array_increment",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following integer array initialization where SIZE is 6, what is the value of arr[4] after execution?",
 "code": "int arr[6];\nfor (int i = 0; i < 6; i++) {\n arr[i] = (i + 1) * 3;\n}\nprintf(\"arr[4]: %d\\n\", arr[4]);",
 "answer": 15
 }
},
{
 "id": "BL-IT-S002-V003",
 "metadata": {
 "name": "布尔数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "bool_array_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 }
}
```

```
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 7
 },
 "task": {
 "description": "Given the following boolean array initialization where SIZE is 7, what is the value of flags[3] after execution?",
 "code": "bool flags[7];\nfor (int i = 0; i < 7; i++) {\n flags[i] = (i % 2 == 0);\n}\nprintf(\"flags[3]: %s\\n\", flags[3] ? \"true\" : \"false\");",
 "answer": "false"
 }
},
{
 "id": "BL-IT-S002-V004",
 "metadata": {
 "name": "二维数组行初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "2d_array_init",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 12
 },
 "task": {
 "description": "Given the following 2D array initialization where ROWS is 3 and COLS is 4, what is the value of matrix[1][2] after execution?",
 "code": "int matrix[3][4];\nfor (int i = 0; i < 3; i++) {\n for (int j = 0; j < 4; j++) {\n matrix[i][j] = i * 4 + j + 1;\n }\n}",
 "answer": 7
 }
},
{
 "id": "BL-IT-S002-V005",
 "metadata": {
 "name": "浮点数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "float_array_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
```

```
 "description": "Given the following float array initialization where SIZE is 5, what is the value of arr[2] after execution?",
 "code": "float arr[5];\nfor (int i = 0; i < 5; i++) {\n arr[i] = (i + 1) * 0.5;\n}\nprintf(\"arr[2]: %.1f\\n\", arr[2]);",
 "answer": 1.5
 },
 {
 "id": "BL-IT-S002-V006",
 "metadata": {
 "name": "指针数组条件初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "pointer_conditional_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following pointer array conditional initialization where SIZE is 8, how many pointers are set to NULL?",
 "code": "int* ptr_arr[8];\nint dummy = 10;\nfor (int i = 0; i < 8; i++) {\n if (i % 3 == 0) {\n ptr_arr[i] = NULL;\n } else {\n ptr_arr[i] = &dummy;\n }\n}",
 "answer": 3
 },
 {
 "id": "BL-IT-S002-V007",
 "metadata": {
 "name": "字符串数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "string_array_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following string array initialization where SIZE is 4, what is the length of strings[2] after execution?",
 "code": "char strings[4][10];\nfor (int i = 0; i < 4; i++) {\n for (int j = 0; j <= i; j++) {\n strings[i][j] = 'x';\n }\n strings[i][i+1] = '\\0';\n}",
 "answer": 5
 }
 }
 }
}
```

```
 "answer": 3
 }
},
{
 "id": "BL-IT-S002-V008",
 "metadata": {
 "name": "结构体数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "struct_array_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following struct array initialization where SIZE is 5, what is the value of points[3].y after execution?",
 "code": "typedef struct { int x, y; } Point;\nPoint points[5];\nfor (int i = 0; i < 5; i++) {\n points[i].x = i * 2;\n points[i].y = i * i;\n}",
 "answer": 9
 }
},
{
 "id": "BL-IT-S002-V009",
 "metadata": {
 "name": "矩阵对角线初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "diagonal_init",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 16
 },
 "task": {
 "description": "Given the following diagonal matrix initialization where SIZE is 4, what is the value of matrix[2][2] after execution?",
 "code": "int matrix[4][4];\nfor (int i = 0; i < 4; i++) {\n for (int j = 0; j < 4; j++) {\n matrix[i][j] = (i == j) ? i + 1 : 0;\n }\n}",
 "answer": 3
 }
},
{
 "id": "BL-IT-S002-V010",
 "metadata": {
 "name": "矩阵对角线初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "diagonal_init",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 16
 },
 "task": {
 "description": "Given the following diagonal matrix initialization where SIZE is 4, what is the value of matrix[2][2] after execution?",
 "code": "int matrix[4][4];\nfor (int i = 0; i < 4; i++) {\n for (int j = 0; j < 4; j++) {\n matrix[i][j] = (i == j) ? i + 1 : 0;\n }\n}",
 "answer": 3
 }
}
```

```
"metadata": {
 "name": "数组反向初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "reverse_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
},
"task": {
 "description": "Given the following reverse array initialization where SIZE is 6, what is the value of arr[1] after execution?",
 "code": "int arr[6];\nfor (int i = 0; i < 6; i++) {\n arr[i] = 6 - i;\n}\nprintf(\"arr[1]: %d\\n\", arr[1]);",
 "answer": 5
},
{
 "id": "BL-IT-S002-V011",
 "metadata": {
 "name": "累积初始化数组",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cumulative_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following cumulative array initialization where SIZE is 5, what is the value of arr[3] after execution?",
 "code": "int arr[5];\narr[0] = 1;\nfor (int i = 1; i < 5; i++) {\n arr[i] = arr[i-1] + i;\n}\nprintf(\"arr[3]: %d\\n\", arr[3]);",
 "answer": 7
 }
},
{
 "id": "BL-IT-S002-V012",
 "metadata": {
 "name": "位模式数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
```

```
"source": "Generated",
"language": "c",
"difficulty": "hard",
"intervention": 2,
"variant_type": "bit_pattern_init",
"loop_pattern": "for",
"nesting_depth": 1,
"iteration_count": 8
},
"task": {
 "description": "Given the following bit pattern array initialization where SIZE is 8, what is the value of arr[5] after execution?",
 "code": "int arr[8];\nfor (int i = 0; i < 8; i++) {\n arr[i] = 1 << i;\n}\nprintf(\"arr[5]: %d\\n\", arr[5]);",
 "answer": 32
},
{
 "id": "BL-IT-S002-V013",
 "metadata": {
 "name": "三维数组初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "3d_array_init",
 "loop_pattern": "for",
 "nesting_depth": 3,
 "iteration_count": 24
},
 "task": {
 "description": "Given the following 3D array initialization where dimensions are 2x3x4, what is the value of cube[1][1][2] after execution?",
 "code": "int cube[2][3][4];\nfor (int i = 0; i < 2; i++) {\n for (int j = 0; j < 3; j++) {\n for (int k = 0; k < 4; k++) {\n cube[i][j][k] = i * 12 + j * 4 + k;\n }\n }\n}",
 "answer": 18
},
 {
 "id": "BL-IT-S002-V014",
 "metadata": {
 "name": "条件跳过初始化",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 }
 }
]
```

```
 "variant_type": "conditional_skip_init",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following conditional skip initialization where SIZE is 10, how many elements are initialized to -1?",
 "code": "int arr[10];\nfor (int i = 0; i < 10; i++) {\n if (i % 3 == 0) {\n continue;\n }\n arr[i] = -1;\n}",
 "answer": 6
 }
},
{
 "id": "BL-IT-S003-V001",
 "metadata": {
 "name": "条件递增while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_increment",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 7
 },
 "task": {
 "description": "Given the following while loop with conditional increment, what is the final value of sum?",
 "code": "int value = 1;\nint sum = 0;\nwhile (value < 100) {\n sum += value;\n value *= 2;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 127
 }
},
{
 "id": "BL-IT-S003-V002",
 "metadata": {
 "name": "字符串长度计算while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "string_length",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 11
 },
}
```

```
"task": {
 "description": "Given the following while loop to calculate string length, what is the final value of length?",
 "code": "char str[] = \"Hello world\";\\nint length = 0;\\nwhile (str[length] != '\\\\0') {\\n length++;\\n}\\nprintf(\"Length: %d\\n\", length);",
 "answer": 11
},
{
 "id": "BL-IT-S003-V003",
 "metadata": {
 "name": "数字反转while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "number_reversal",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following while loop to reverse a number, what is the final value of reversed?",
 "code": "int num = 1234;\\nint reversed = 0;\\nwhile (num > 0) {\\n reversed = reversed * 10 + num % 10;\\n num /= 10;\\n}\\nprintf(\"Reversed: %d\\n\", reversed);",
 "answer": 4321
 }
},
{
 "id": "BL-IT-S003-V004",
 "metadata": {
 "name": "最大公约数while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "gcd_calculation",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following while loop to calculate GCD using Euclidean algorithm, what is the final value of a?",
 "code": "int a = 48, b = 18;\\nwhile (b != 0) {\\n int temp = b;\\n b = a % b;\\n a = temp;\\n}\\nprintf(\"GCD: %d\\n\", a);",
 "answer": 6
 }
}
```

```

 "answer": 6
 }
},
{
 "id": "BL-IT-S003-V005",
 "metadata": {
 "name": "平方根逼近while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "sqrt_approximation",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following while loop for square root approximation, what is the final value of x (rounded to nearest integer)?",
 "code": "double x = 10.0;\ndouble target = 25.0;\nwhile (x * x - target > 0.1 || target - x * x > 0.1) {\n x = (x + target / x) / 2.0;\n}\nprintf(\"Result: %.0f\\n\", x);",
 "answer": 5
 }
},
{
 "id": "BL-IT-S003-V006",
 "metadata": {
 "name": "斐波那契while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "fibonacci_sequence",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following while loop to generate Fibonacci sequence, what is the final value of current?",
 "code": "int prev = 0, current = 1, next;\nint limit = 20;\nwhile (current <= limit) {\n next = prev + current;\n prev = current;\n current = next;\n}\nprintf(\"Current: %d\\n\", current);",
 "answer": 34
 }
},
{
 "id": "BL-IT-S003-V007",
 "metadata": {
 "name": "斐波那契while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "fibonacci_sequence",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 7
 },
 "task": {
 "description": "Given the following while loop to generate Fibonacci sequence, what is the final value of current?",
 "code": "int prev = 0, current = 1, next;\nint limit = 20;\nwhile (current <= limit) {\n next = prev + current;\n prev = current;\n current = next;\n}\nprintf(\"Current: %d\\n\", current);",
 "answer": 34
 }
}

```



```
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cumulative_product",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following while loop for cumulative product, what is the final value of product?",
 "code": "int i = 1;\nint product = 1;\nwhile (i <= 5) {\n product *= i;\n i++;\n}\nprintf(\"Product: %d\\n\", product);",
 "answer": 120
 }
},
{
 "id": "BL-IT-S003-V010",
 "metadata": {
 "name": "条件累加while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_sum",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following while loop with conditional sum, what is the final value of sum?",
 "code": "int i = 1;\nint sum = 0;\nwhile (i <= 10) {\n if (i % 2 == 0) {\n sum += i;\n }\n i++;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 30
 }
},
{
 "id": "BL-IT-S003-V011",
 "metadata": {
 "name": "搜索目标while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
```

```
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "target_search",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following while loop to search for target, what is the final value of index?",
 "code": "int arr[] = {10, 20, 30, 40, 50};\nint target = 40;\nint index = 0;\nwhile (index < 5 && arr[index] != target) {\n index++;\n}\nprintf(\"Index: %d\\n\", index);",
 "answer": 3
 }
},
{
 "id": "BL-IT-S003-V012",
 "metadata": {
 "name": "幂运算while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "power_calculation",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following while loop to calculate 3^4 , what is the final value of result?",
 "code": "int base = 3;\nint exp = 4;\nint result = 1;\nwhile (exp > 0) {\n result *= base;\n exp--;\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 81
 }
},
{
 "id": "BL-IT-S003-V013",
 "metadata": {
 "name": "数组最小值while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "min_finding",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 4
 }
}
```

```
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following while loop to find minimum value, what is the final value of min?",
 "code": "int arr[] = {8, 3, 9, 1, 5, 7};\nint min = arr[0];\nint i = 1;\nwhile (i < 6) {\n if (arr[i] < min) {\n min = arr[i];\n }\n i++;\n}\nprintf(\"Min: %d\\n\", min);",
 "answer": 1
 }
},
{
 "id": "BL-IT-S003-V014",
 "metadata": {
 "name": "阶乘计算while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "factorial_calculation",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following while loop to calculate 6!, what is the final value of factorial?",
 "code": "int n = 6;\nint factorial = 1;\nwhile (n > 0) {\n factorial *= n;\n n--;\n}\nprintf(\"Factorial: %d\\n\", factorial);",
 "answer": 720
 }
},
{
 "id": "BL-IT-S003-V015",
 "metadata": {
 "name": "边界条件while循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "boundary_condition",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 0
 },
 "task": {
```

```
 "description": "Given the following while loop with boundary condition, what is the final value of count?",
 "code": "int count = 5;\nint condition = 0;\nwhile (condition > 0) {\n count++;\n condition--;\n}\nprintf(\"Count: %d\\n\", count);",
 "answer": 5
 },
 {
 "id": "BL-IT-S004-V001",
 "metadata": {
 "name": "三层嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "triple_nested",
 "loop_pattern": "for",
 "nesting_depth": 3,
 "iteration_count": 24
 },
 "task": {
 "description": "Given the following triple nested loop, what is the final value of total?",
 "code": "total = 0\nfor i in range(2):\n for j in range(3):\n for k in range(4):\n total += (i + 1) * (j + 1) * (k + 1)\nprintf(f\"Total: {total}\")",
 "answer": 150
 },
 },
 {
 "id": "BL-IT-S004-V002",
 "metadata": {
 "name": "不等长嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "unequal_nested",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following unequal nested loop, what is the final value of total?",
 "code": "total = 0\nfor i in range(4):\n for j in range(i + 1):\n total += (i + j)\nprintf(f\"Total: {total}\")",
 "answer": 20
 },
 },
}
```

```
 },
 },
 {
 "id": "BL-IT-S004-V003",
 "metadata": {
 "name": "条件嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_nested",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following conditional nested loop, what is the final value of count?",
 "code": "count = 0\nfor i in range(3):\n for j in range(3):\n if i + j\n >= 2:\n count += 1\nprint(f\"Count: {count}\")",
 "answer": 6
 }
 },
 {
 "id": "BL-IT-S004-V004",
 "metadata": {
 "name": "矩阵乘法嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "matrix_multiplication",
 "loop_pattern": "for",
 "nesting_depth": 3,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following matrix multiplication nested loop for 2x2 matrices, what is the value of result[1][0]?",
 "code": "A = [[1, 2], [3, 4]]\nB = [[2, 0], [1, 3]]\nresult = [[0, 0], [0, 0]]\nfor i in range(2):\n for j in range(2):\n for k in range(2):\n result[i][j] += A[i][k] * B[k][j]\nprint(f\"result[1][0]: {result[1][0]}\")",
 "answer": 10
 }
 },
 {
 "id": "BL-IT-S004-V005",
 "metadata": {
 "name": "矩阵乘法嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "matrix_multiplication",
 "loop_pattern": "for",
 "nesting_depth": 3,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following matrix multiplication nested loop for 2x2 matrices, what is the value of result[1][0]?",
 "code": "A = [[1, 2], [3, 4]]\nB = [[2, 0], [1, 3]]\nresult = [[0, 0], [0, 0]]\nfor i in range(2):\n for j in range(2):\n for k in range(2):\n result[i][j] += A[i][k] * B[k][j]\nprint(f\"result[1][0]: {result[1][0]}\")",
 "answer": 10
 }
 }
]
```

```
"metadata": {
 "name": "对角线累加嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "diagonal_sum",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
},
"task": {
 "description": "Given the following nested loop for diagonal sum, what is the final value of diagonal_sum?",
 "code": "matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]\ndiagonal_sum = 0\nfor i in range(3):\n for j in range(3):\n if i == j:\n diagonal_sum += matrix[i][j]\nprint(f\"Diagonal sum: {diagonal_sum}\")",
 "answer": 15
}
},
{
 "id": "BL-IT-S004-V006",
 "metadata": {
 "name": "上三角矩阵嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "upper_triangle",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 6
},
 "task": {
 "description": "Given the following nested loop for upper triangle sum, what is the final value of sum_upper?",
 "code": "matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]\nsum_upper = 0\nfor i in range(3):\n for j in range(i, 3):\n sum_upper += matrix[i][j]\nprint(f\"Upper triangle sum: {sum_upper}\")",
 "answer": 21
}
},
{
 "id": "BL-IT-S004-V007",
 "metadata": {
 "name": "字符串模式嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "string",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
}
}
```

```
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "string_pattern",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following nested loop creating string pattern, what is the length of result?",
 "code": "result = \"\"\nfor i in range(4):\n for j in range(i + 1):\n result += \"*\n result += \"\\n\\n\\n\\nprint(f\"Length: {len(result)}\")",
 "answer": 14
 }
},
{
 "id": "BL-IT-S004-V008",
 "metadata": {
 "name": "列表推导式等价嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "list_comprehension_equivalent",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 12
 },
 "task": {
 "description": "Given the following nested loop equivalent to list comprehension, what is the sum of all elements in result?",
 "code": "result = []\nfor i in range(3):\n for j in range(4):\n result.append(i * j)\n total = sum(result)\n nprint(f\"Total: {total}\")",
 "answer": 18
 }
},
{
 "id": "BL-IT-S004-V009",
 "metadata": {
 "name": "嵌套循环最大值查找",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
```

```
 "intervention": 1,
 "variant_type": "nested_max_finding",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop to find maximum, what is the final value of max_val?",
 "code": "matrix = [[3, 7, 1], [9, 2, 5], [4, 8, 6]]\nmax_val = matrix[0][0]\nfor i in range(3):\n for j in range(3):\n if matrix[i][j] > max_val:\n max_val = matrix[i][j]\nprint(f\"Max: {max_val}\")",
 "answer": 9
 }
},
{
 "id": "BL-IT-S004-V010",
 "metadata": {
 "name": "嵌套循环计数器",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "nested_counter",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop counter, what is the final value of even_count?",
 "code": "matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]\neven_count = 0\nfor i in range(3):\n for j in range(3):\n if matrix[i][j] % 2 == 0:\n even_count += 1\nprint(f\"Even count: {even_count}\")",
 "answer": 4
 }
},
{
 "id": "BL-IT-S004-V011",
 "metadata": {
 "name": "乘积表嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "multiplication_table",
 "loop_pattern": "for",
```

```
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop creating multiplication table, what is the value of table[2][1]?",
 "code": "table = [[0 for _ in range(3)] for _ in range(3)]\nfor i in range(3):\n for j in range(3):\n table[i][j] = (i + 1) * (j + 1)\nprint(f\"table[2][1]: {table[2][1]}\")",
 "answer": 6
 }
},
{
 "id": "BL-IT-S004-V012",
 "metadata": {
 "name": "坐标距离嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "coordinate_distance",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop calculating Manhattan distances, what is the final value of total_distance?",
 "code": "total_distance = 0\nfor i in range(3):\n for j in range(3):\n distance = abs(i - 1) + abs(j - 1)\n total_distance += distance\nprint(f\"Total distance: {total_distance}\")",
 "answer": 8
 }
},
{
 "id": "BL-IT-S004-V013",
 "metadata": {
 "name": "帕斯卡三角嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "pascal_triangle",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 6
 },
}
```

```

"task": {
 "description": "Given the following nested loop generating Pascal's triangle, what is the value of triangle[2][1]?",
 "code": "triangle = [[1], [1, 1], [1, 0, 1]]\nfor i in range(1, 3):\n for j in range(1, i):\n triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j]\nprint(f\"triangle[2][1]: {triangle[2][1]}\")",
 "answer": 2
},
{

 "id": "BL-IT-S004-V014",
 "metadata": {
 "name": "螺旋矩阵嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "spiral_matrix",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop for spiral pattern sum, what is the final value of center_sum?",
 "code": "matrix = [[1, 2, 3], [8, 9, 4], [7, 6, 5]]\ncenter_sum = 0\nfor i in range(1, 2):\n for j in range(1, 2):\n center_sum += matrix[i][j]\n for di in [-1, 0, 1]:\n for dj in [-1, 0, 1]:\n if 0 <= i+di < 3 and 0 <= j+dj < 3 and (di != 0 or dj != 0):\n center_sum += matrix[i+di][j+dj]\nprint(f\"Center sum: {center_sum}\")",
 "answer": 45
 }
},
{

 "id": "BL-IT-S005-V001",
 "metadata": {
 "name": "多重break嵌套循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multiple_break",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 8
 },
 "task": {

```



```
"code": "int count = 0;\nfor (int i = 1; i <= 3; i++) {\n for (int j = 1; j\n<= 3; j++) {\n count++;\n if (i + j == 4) goto end;\n }\n}\nend:\nprintf(\"Count: %d\\n\", count);",
"answer": 4
}
},
{
"id": "BL-IT-S005-V004",
"metadata": {
"name": "函数返回中断循环",
"category": "Block-Level",
"subcategory": "Iterative",
"type": "variant",
"source": "Generated",
"language": "c",
"difficulty": "hard",
"intervention": 2,
"variant_type": "function_return",
"loop_pattern": "for",
"nesting_depth": 1,
"iteration_count": 3
},
"task": {
"description": "Given the following function with return in loop, what value does the function return?",
"code": "int find_first_multiple(int target) {\n for (int i = 1; i <= 10;\n i++) {\n if (i * 3 >= target) {\n return i * 3;\n }\n }\n return -1;\n}\nint result = find_first_multiple(8);",
"answer": 9
}
},
{
"id": "BL-IT-S005-V005",
"metadata": {
"name": "标志变量控制循环",
"category": "Block-Level",
"subcategory": "Iterative",
"type": "variant",
"source": "Generated",
"language": "c",
"difficulty": "hard",
"intervention": 2,
"variant_type": "flag_control",
"loop_pattern": "for",
"nesting_depth": 1,
"iteration_count": 6
},
"task": {
"description": "Given the following loop with flag control, what is the final value of i?",
"code": "int found = 0;\nint i;\nfor (i = 1; i <= 10 && !found; i++) {\n if (i * i > 30) {\n found = 1;\n }\n}\nprintf(\"i: %d\\n\", i);",
"answer": 7
}
}
```

```

 }
 },
 {
 "id": "BL-IT-S005-V006",
 "metadata": {
 "name": "switch控制流循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "switch_control",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following loop with switch control flow, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 5; i++) {\n switch (i % 3) {\n case 0:\n sum += i * 2;\n break;\n case 1:\n sum += i;\n break;\n case 2:\n continue;\n }\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 7
 }
 },
 {
 "id": "BL-IT-S005-V007",
 "metadata": {
 "name": "多条件break循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "multi_condition_break",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following loop with multiple break conditions, what is the final value of product?",
 "code": "int product = 1;\nfor (int i = 1; i <= 10; i++) {\n product *= i;\n if (product > 20 || i >= 4) {\n break;\n }\n}\nprintf(\"Product: %d\\n\", product);",
 "answer": 24
 }
 },
}

```

```
{
 "id": "BL-IT-S005-V008",
 "metadata": {
 "name": "异常模拟控制流",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "exception_simulation",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following loop simulating exception handling, what is the final value of count?",
 "code": "int count = 0;\nint error_flag = 0;\nfor (int i = 1; i <= 10; i++) {\n if (i == 8) {\n error_flag = 1;\n break;\n }\n count++;\n}\nif (error_flag) {\n count = -1;\n}\nprintf(\"Count: %d\\n\", count);",
 "answer": -1
 }
},
{
 "id": "BL-IT-S005-V009",
 "metadata": {
 "name": "条件continue累加",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_continue",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 12
 },
 "task": {
 "description": "Given the following loop with conditional continue, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 12; i++) {\n if (i % 3 == 0 || i % 4 == 0) {\n continue;\n }\n sum += i;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 30
 }
},
{
 "id": "BL-IT-S005-V010",
 "metadata": {
 "name": "嵌套continue控制",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "nested_continue",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following nested loop with continue statements, what is the final value of sum?",
 "code": "int sum = 0;\nfor (int i = 1; i <= 10; i++) {\n for (int j = 1; j <= 5; j++) {\n if (j % 2 == 0) {\n continue;\n }\n sum += i;\n }\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 50
 }
}
```

```
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_continue",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following nested loop with continue, what is the final value of count?",
 "code": "int count = 0;\nfor (int i = 1; i <= 3; i++) {\n for (int j = 1; j <= 3; j++) {\n if (i == j) {\n continue;\n }\n count++;\n }\n}\nprintf(\"Count: %d\\n\", count);",
 "answer": 6
 }
},
{
 "id": "BL-IT-S005-V011",
 "metadata": {
 "name": "do-while中的break",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "do_while_break",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following do-while loop with break, what is the final value of sum?",
 "code": "int sum = 0;\ndo {\n sum += i;\n i++;\n if (sum > 5) {\n break;\n }\n} while (i <= 10);\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 6
 }
},
{
 "id": "BL-IT-S005-V012",
 "metadata": {
 "name": "while中的continue",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
```

```
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "while_continue",
 "loop_pattern": "while",
 "nesting_depth": 1,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following while loop with continue, what is the final value of sum?",
 "code": "int sum = 0;\nint i = 1;\nwhile (i <= 10) {\n if (i % 3 == 0) {\n i++;\n continue;\n }\n sum += i;\n i++;\n}\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 37
 }
},
{
 "id": "BL-IT-S005-V013",
 "metadata": {
 "name": "标签跳转循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "label_jump",
 "loop_pattern": "for",
 "nesting_depth": 2,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following nested loop with label jump, what is the final value of total?",
 "code": "int total = 0;\nouter:\nfor (int i = 1; i <= 3; i++) {\n for (int j = 1; j <= 3; j++) {\n total += i + j;\n if (total > 10) {\n goto outer_end;\n }\n }\n}\nouter_end:\nprintf(\"Total: %d\\n\", total);",
 "answer": 12
 }
},
{
 "id": "BL-IT-S005-V014",
 "metadata": {
 "name": "复合控制流循环",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
```

```
 "variant_type": "compound_control_flow",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following loop with compound control flow, what is the final value of result?",
 "code": "int result = 0;\nfor (int i = 1; i <= 15; i++) {\n if (i % 2 == 0)\n continue;\n if (i > 10) break;\n if (i % 3 == 0) {\n result += i * 2;\n }\n else {\n result += i;\n }\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 34
 }
},
{
 "id": "BL-IT-S006-V001",
 "metadata": {
 "name": "do-while递增到阈值",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "threshold_increment",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following do-while loop incrementing to threshold, what is the final value of value?",
 "code": "int value = 2;\ndo {\n value *= 2;\n} while (value < 20);\nprintf(\"value: %d\\n\", value);",
 "answer": 32
 }
},
{
 "id": "BL-IT-S006-V002",
 "metadata": {
 "name": "do-while字符递增",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "char_increment",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 3
 }
}
```

```
 },
 "task": {
 "description": "Given the following do-while loop with character increment, what is the final value of ch?",
 "code": "char ch = 'x';\ndo {\n ch++;\\n} while (ch < 'z');\\nprintf(\"Character: %c\\n\", ch);",
 "answer": "z"
 }
 },
 {
 "id": "BL-IT-S006-V003",
 "metadata": {
 "name": "do-while累积求和",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cumulative_sum",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following do-while loop with cumulative sum, what is the final value of sum?",
 "code": "int sum = 0;\nint i = 1;\ndo {\n sum += i * i;\n i++;\\n} while (i <= 4);\\nprintf(\"Sum: %d\\n\", sum);",
 "answer": 30
 }
 },
 {
 "id": "BL-IT-S006-V004",
 "metadata": {
 "name": "do-while除法递减",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "division_decrement",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following do-while loop with division, what is the final value of num?",
 "code": "int num = 10;\ndo {\n num /= 2;\n} while (num > 0);\\nprintf(\"Final value: %d\\n\", num);",
 "answer": 1
 }
 }
}
```

```
 "code": "int num = 100;\nint count = 0;\ndo {\n num /= 2;\n count++;\n}\nwhile (num > 5);\nprintf(\"Count: %d\\n\", count);",
 "answer": 4
 },
},
{
 "id": "BL-IT-S006-V005",
 "metadata": {
 "name": "do-while条件累加",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_accumulation",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following do-while loop with conditional accumulation, what is the final value of result?",
 "code": "int result = 0;\nint i = 1;\ndo {\n if (i % 2 == 1) {\n result += i;\n } else {\n result -= i;\n }\n i++;\n} while (i <= 6);\nprintf(\"Result: %d\\n\", result);",
 "answer": -3
 }
},
{
 "id": "BL-IT-S006-V006",
 "metadata": {
 "name": "do-while数组填充",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "array_filling",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following do-while loop filling array, what is the value of arr[3]?",
 "code": "int arr[5];\nint i = 0;\ndo {\n arr[i] = (i + 1) * 3;\n i++;\n}\nwhile (i < 5);\nprintf(\"arr[3]: %d\\n\", arr[3]);",
 "answer": 12
 }
}
```

```
},
{
 "id": "BL-IT-S006-V007",
 "metadata": {
 "name": "do-while浮点递增",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "float_increment",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following do-while loop with float increment, what is the final value of sum (rounded to 1 decimal)?",
 "code": "float sum = 0.0;\nfloat x = 0.5;\ndo {\n sum += x;\n x += 0.5;\n}\nwhile (x <= 2.5);\nprintf(\"Sum: %.1f\\n\", sum);",
 "answer": 7.5
 }
},
{
 "id": "BL-IT-S006-V008",
 "metadata": {
 "name": "do-while最大值查找",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "max_finding",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following do-while loop finding maximum, what is the final value of max?",
 "code": "int arr[] = {3, 8, 2, 9, 1};\nint max = arr[0];\nint i = 1;\ndo {\n if (arr[i] > max) {\n max = arr[i];\n }\n i++;\n} while (i < 5);\nprintf(\"Max: %d\\n\", max);",
 "answer": 9
 }
},
{
 "id": "BL-IT-S006-V009",
 "metadata": {
```

```
 "name": "do-while位运算",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "bitwise_operation",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following do-while loop with bitwise operations, what is the final value of result?",
 "code": "int result = 1;\nint shift = 1;\ndo {\n result |= (1 << shift);\n shift++;\n} while (shift <= 3);\nprintf(\"Result: %d\\n\", result);",
 "answer": 15
 }
},
{
 "id": "BL-IT-S006-V010",
 "metadata": {
 "name": "do-while字符串构建",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "string_building",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following do-while loop building string, what is the length of the resulting string?",
 "code": "char str[10] = \"\";\nint len = 0;\nchar ch = 'A';\ndo {\n str[len] = ch;\n len++;\n ch++;\n} while (ch <= 'C');\nstr[len] = '\\0';\nprintf(\"Length: %d\\n\", len);",
 "answer": 3
 }
},
{
 "id": "BL-IT-S006-V011",
 "metadata": {
 "name": "do-while阶乘计算",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
```

```
"source": "Generated",
"language": "c",
"difficulty": "easy",
"intervention": 0,
"variant_type": "factorial_calculation",
"loop_pattern": "do_while",
"nesting_depth": 1,
"iteration_count": 5
},
"task": {
 "description": "Given the following do-while loop calculating factorial, what is the final value of factorial?",
 "code": "int factorial = 1;\nint n = 5;\ndo {\n factorial *= n;\n n--;\n}\nwhile (n > 0);\nprintf(\"Factorial: %d\\n\", factorial);",
 "answer": 120
}
},
{
 "id": "BL-IT-S006-V012",
 "metadata": {
 "name": "do-while倒数累加",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "reciprocal_sum",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following do-while loop with reciprocal sum, what is the final value of sum (rounded to 2 decimals)?",
 "code": "double sum = 0.0;\nint i = 1;\ndo {\n sum += 1.0 / i;\n i++;\n}\nwhile (i <= 4);\nprintf(\"Sum: %.2f\\n\", sum);",
 "answer": 2.08
 }
},
{
 "id": "BL-IT-S006-V013",
 "metadata": {
 "name": "do-while模运算计数",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "modulo_counting",
```

```
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following do-while loop counting with modulo, what is the final value of count?",
 "code": "int count = 0;\nint i = 1;\ndo {\n if (i % 3 == 0) {\n count++;\n }\n i++;\n} while (i <= 12);\nprintf(\"Count: %d\\n\", count);",
 "answer": 4
 }
},
{
 "id": "BL-IT-S006-V014",
 "metadata": {
 "name": "do-while边界测试",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "boundary_test",
 "loop_pattern": "do_while",
 "nesting_depth": 1,
 "iteration_count": 1
 },
 "task": {
 "description": "Given the following do-while loop with false condition, what is the final value of counter?",
 "code": "int counter = 10;\ndo {\n counter += 5;\n} while\n(0);\nprintf(\"Counter: %d\\n\", counter);",
 "answer": 15
 }
},
{
 "id": "BL-IT-S007-V001",
 "metadata": {
 "name": "尾递归阶乘",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "tail_recursive_factorial",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
```

```

 "description": "Given the following tail recursive factorial function called with factorial_tail(5, 1), what is the return value?",
 "code": "int factorial_tail(int n, int acc) {\n if (n <= 1) {\n return acc;\n }\n return factorial_tail(n - 1, n * acc);\n}\nint result = factorial_tail(5, 1);\nprintf(\"Result: %d\\n\", result);",
 "answer": 120
}
},
{
 "id": "BL-IT-S007-V002",
 "metadata": {
 "name": "斐波那契递归",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "fibonacci_recursive",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 9
 },
 "task": {
 "description": "Given the following recursive Fibonacci function called with fibonacci(6), what is the return value?",
 "code": "int fibonacci(int n) {\n if (n <= 1) {\n return n;\n }\n return fibonacci(n - 1) + fibonacci(n - 2);\n}\nint result = fibonacci(6);\nprintf(\"Result: %d\\n\", result);",
 "answer": 8
 }
},
{
 "id": "BL-IT-S007-V003",
 "metadata": {
 "name": "递归幂运算",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "recursive_power",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following recursive power function called with power(2, 3), what is the return value?",

```

```
 "code": "int power(int base, int exp) {\n if (exp == 0) {\n return\n1;\n }\n return base * power(base, exp - 1);\n}\n\nint result = power(2,\n3);\n\nprintf(\"Result: %d\\n\", result);",
 "answer": 8
 }
},
{
 "id": "BL-IT-S007-V004",
 "metadata": {
 "name": "递归数组求和",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "recursive_array_sum",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following recursive array sum function called with array [1,2,3,4,5] and size 5, what is the return value?",
 "code": "int array_sum(int arr[], int size) {\n if (size == 0) {\n return 0;\n }\n return arr[size - 1] + array_sum(arr, size - 1);\n}\n\nint arr[] = {1,\n2, 3, 4, 5};\n\nint result = array_sum(arr, 5);\n\nprintf(\"Result: %d\\n\", result);",
 "answer": 15
 }
},
{
 "id": "BL-IT-S007-V005",
 "metadata": {
 "name": "递归最大公约数",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "recursive_gcd",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive GCD function called with gcd(48, 18), what is the return value?",
 "code": "int gcd(int a, int b) {\n if (b == 0) {\n return a;\n }\n return gcd(b, a % b);\n}\n\nint result = gcd(48, 18);\n\nprintf(\"Result: %d\\n\", result);",
 "answer": 6
 }
}
```

```

 "answer": 6
 }
},
{
 "id": "BL-IT-S007-V006",
 "metadata": {
 "name": "递归数字反转",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_number_reverse",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive number reversal function called with reverse(1234, 0), what is the return value?",
 "code": "int reverse(int num, int rev) {\n if (num == 0) {\n return rev;\n }\n return reverse(num / 10, rev * 10 + num % 10);\n}\n\nint result = reverse(1234, 0);\nprintf(\"Result: %d\\n\", result);",
 "answer": 4321
 }
},
{
 "id": "BL-IT-S007-V007",
 "metadata": {
 "name": "递归数字位数",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "recursive_digit_count",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive digit counting function called with count_digits(1234), what is the return value?",
 "code": "int count_digits(int num) {\n if (num == 0) {\n return 0;\n }\n return 1 + count_digits(num / 10);\n}\n\nint result = count_digits(1234);\nprintf(\"Result: %d\\n\", result);",
 "answer": 4
 }
},
{
 "id": "BL-IT-S007-V008",
 "metadata": {
 "name": "递归字符串反转",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "recursive_string_reverse",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive string reversal function called with reverse(\"abcde\"), what is the return value?",
 "code": "char* reverse(char* str) {\n if (*str == '\\0') {\n return str;\n }\n char* result = reverse(str + 1);\n result[0] = *str;\n return result;\n}\n\nchar* result = reverse(\"abcde\");\nprintf(\"Result: %s\\n\", result);",
 "answer": "edcba"
 }
}

```

```
{
 "id": "BL-IT-S007-V008",
 "metadata": {
 "name": "递归回文检查",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_palindrome",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following recursive palindrome check function called with is_palindrome(\"racecar\", 0, 6), what is the return value?",
 "code": "int is_palindrome(char str[], int start, int end) {\n if (start >= end) {\n return 1;\n }\n if (str[start] != str[end]) {\n return 0;\n }\n return is_palindrome(str, start + 1, end - 1);\n}\nchar str[] = \"racecar\";\nint result = is_palindrome(str, 0, 6);\nprintf(\"Result: %d\\n\", result);",
 "answer": 1
 }
},
{
 "id": "BL-IT-S007-V009",
 "metadata": {
 "name": "递归二进制转换",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "recursive_binary",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive binary conversion function called with count_binary_ones(13), what is the return value?",
 "code": "int count_binary_ones(int num) {\n if (num == 0) {\n return 0;\n }\n return (num & 1) + count_binary_ones(num >> 1);\n}\nint result = count_binary_ones(13);\nprintf(\"Result: %d\\n\", result);",
 "answer": 3
 }
},
{
 "id": "BL-IT-S007-V010",
 "metadata": {
 "name": "递归斐波那契数列",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 2,
 "variant_type": "recursive_fibonacci",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following recursive Fibonacci sequence function called with fibonacci(5), what is the return value?",
 "code": "int fibonacci(int n) {\n if (n == 0) {\n return 0;\n }\n if (n == 1) {\n return 1;\n }\n return fibonacci(n - 1) + fibonacci(n - 2);\n}\nint result = fibonacci(5);\nprintf(\"Result: %d\\n\", result);",
 "answer": 5
 }
}
```

```
"metadata": {
 "name": "相互递归函数",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "mutual_recursion",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 6
},
"task": {
 "description": "Given the following mutually recursive functions called with is_even(6), what is the return value?",
 "code": "int is_odd(int n);\nint is_even(int n) {\n if (n == 0) return 1;\n return is_odd(n - 1);\n}\nint is_odd(int n) {\n if (n == 0) return 0;\n return is_even(n - 1);\n}\nint result = is_even(6);\\nprintf(\"Result: %d\\n\", result);",
 "answer": 1
},
{
 "id": "BL-IT-S007-V011",
 "metadata": {
 "name": "递归汉诺塔",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_hanoi",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 7
},
 "task": {
 "description": "Given the following recursive Hanoi tower function counting moves for hanoi(3), what is the return value?",
 "code": "int hanoi(int n) {\n if (n == 1) {\n return 1;\n }\n return 2 * hanoi(n - 1) + 1;\n}\nint result = hanoi(3);\\nprintf(\"Result: %d\\n\", result);",
 "answer": 7
},
{
 "id": "BL-IT-S007-V012",
 "metadata": {
 "name": "递归字符串长度",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "string_length",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 6
}
}
```

```
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "recursive_string_length",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following recursive string length function called with str_length(\"Hello\"), what is the return value?",
 "code": "int str_length(char str[]) {\n if (str[0] == '\\0') {\n return 0;\n }\n return 1 + str_length(str + 1);\n}\nchar str[] = \"Hello\";\nint result = str_length(str);\nprintf(\"Result: %d\\n\", result);",
 "answer": 5
 }
},
{
 "id": "BL-IT-S007-V013",
 "metadata": {
 "name": "递归树形结构",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_tree_sum",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 3
 },
 "task": {
 "description": "Given the following recursive tree sum function, what is the return value for a tree with root value 5 and children 2 and 3?",
 "code": "typedef struct Node {\n int value;\n struct Node* left;\n struct Node* right;\n} Node;\nint tree_sum(Node* node) {\n if (node == NULL) {\n return 0;\n }\n return node->value + tree_sum(node->left) + tree_sum(node->right);\n}\n// Tree: root(5) -> left(2), right(3)\nNode left = {2, NULL, NULL};\nNode right = {3, NULL, NULL};\nNode root = {5, &left, &right};\nint result = tree_sum(&root);",
 "answer": 10
 }
},
{
 "id": "BL-IT-S007-V014",
 "metadata": {
 "name": "递归快速排序分区",
 "category": "Block-Level",
 "subcategory": "Iterative",
```

```
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_quicksort_partition",
 "loop_pattern": "recursive",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following recursive partition count function called with count_partition([3,1,4,1,5], 5, 3), what is the return value?",
 "code": "int count_partition(int arr[], int size, int pivot) {\n if (size == 0) {\n return 0;\n }\n int count = (arr[0] <= pivot) ? 1 : 0;\n return count + count_partition(arr + 1, size - 1, pivot);\n}\nint arr[] = {3, 1, 4, 1, 5};\nint result = count_partition(arr, 5, 3);\nprintf(\"Result: %d\\n\", result);",
 "answer": 3
 }
},
{
 "id": "BL-IT-S008-V001",
 "metadata": {
 "name": "多重累积过滤器",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multi_accumulator_filter",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 10
 },
 "task": {
 "description": "Given the following complex iterative accumulation with multiple filters, what is the final value of total_score?",
 "code": "numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]\ntotal_score = 0\nbonus_multiplier = 1\npenalty = 0\nfor num in numbers:\n if num % 2 == 0:\n total_score += num * bonus_multiplier\n bonus_multiplier += 0.5\n elif num % 3 == 0:\n total_score += num * 2\n penalty += 1\n else:\n total_score -= penalty\n\nprint(f\"Total Score: {total_score}\")",
 "answer": 51.0
 }
},
{
 "id": "BL-IT-S008-V002",
 "metadata": {
 "name": "状态机累积器",
 "category": "Block-Level",
 "subcategory": "Iterative",
```

```
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "state_machine_accumulator",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following state machine accumulation, what is the final value of result?",
 "code": "data = [1, 3, 2, 4, 1, 5, 2, 3]\nresult = 0\nstate = 'ADD'\nthreshold = 3\n\nfor value in data:\n if state == 'ADD':\n result += value\n if value >= threshold:\n state = 'MULTIPLY'\n elif state == 'MULTIPLY':\n result *= value\n if value < threshold:\n state = 'SUBTRACT'\n elif state == 'SUBTRACT':\n result -= value\n if value >= threshold:\n state = 'ADD'\n\n \nprint(f\"Result: {result}\")",
 "answer": 26
 }
},
{
 "id": "BL-IT-S008-V003",
 "metadata": {
 "name": "动态阈值累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "dynamic_threshold",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following dynamic threshold accumulation, what is the final value of sum?",
 "code": "values = [10, 5, 15, 8, 20, 3]\nsum = 0\nthreshold = 7\nadjustment = 2\n\nfor i, value in enumerate(values):\n if value > threshold:\n sum += value\n threshold += adjustment\n else:\n sum -= value\n threshold = max(1, threshold - adjustment)\n\nprint(f\"sum: {sum}\")",
 "answer": 19
 }
},
{
 "id": "BL-IT-S008-V004",
 "metadata": {
 "name": "窗口滑动累积",
 "category": "Block-Level",
 "subcategory": "Iterative"
 }
}
```

```
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "sliding_window",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following sliding window accumulation with window size 3, what is the final value of max_sum?",
 "code": "numbers = [1, 4, 2, 5, 3, 8, 1, 6]\nmax_sum = 0\nwindow_size = 3\n\nfor i in range(len(numbers) - window_size + 1):\n current_sum = 0\n for j in\n range(window_size):\n current_sum += numbers[i + j]\n max_sum = max(max_sum,\n current_sum)\n\n \nprint(f\"Max Sum: {max_sum}\")",
 "answer": 16
 }
},
{
 "id": "BL-IT-S008-V005",
 "metadata": {
 "name": "优先级权重累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "priority_weighted",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following priority weighted accumulation, what is the final value of weighted_sum?",
 "code": "items = [(10, 1), (5, 3), (15, 2), (8, 1), (12, 4), (6,\n2)]\nweighted_sum = 0\nbonus_threshold = 10\n\nfor value, priority in items:\n weight =\n priority * 2 if value >= bonus_threshold else priority\n weighted_sum += value * weight\n\nprint(f\"Weighted Sum: {weighted_sum}\")",
 "answer": 258
 }
},
{
 "id": "BL-IT-S008-V006",
 "metadata": {
 "name": "条件分支累积器",
 "category": "Block-Level",
 "subcategory": "Iterative",
```

```
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_branching",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 7
 },
 "task": {
 "description": "Given the following conditional branching accumulation, what is the final value of result?",
 "code": "sequence = [2, 7, 1, 9, 4, 6, 3]\nresult = 100\noperations = 0\n\nfor num in sequence:\n if num < 5:\n if operations % 2 == 0:\n result +=\nnum * 3\n else:\n result -= num\n else:\n result = result // 2 +\nnum\noperations += 1\n\nprint(f\"Result: {result}\")",
 "answer": 22
 }
},
{
 "id": "BL-IT-S008-V007",
 "metadata": {
 "name": "累积历史追踪",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "history_tracking",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following accumulation with history tracking, what is the final value of current?",
 "code": "values = [3, 8, 2, 5, 7]\ncurrent = 0\nhistory = []\nmax_history = 3\n\nfor value in values:\n current += value\n history.append(current)\n if len(history) > max_history:\n history.pop(0)\n if len(history) == max_history:\n avg = sum(history) / len(history)\n if current > avg:\n current =\nint(avg)\n\nprint(f\"Current: {current}\")",
 "answer": 16
 }
},
{
 "id": "BL-IT-S008-V008",
 "metadata": {
 "name": "指数衰减累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "exponential_decay",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following exponential decay accumulation, what is the final value of result?",
 "code": "values = [10, 5, 2, 1]\nresult = 0\n\nfor value in values:\n result += value\n result *= 0.5\n\nprint(f\"Result: {result}\")",
 "answer": 1.5
 }
}
```

```
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "exponential_decay",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 4
 },
 "task": {
 "description": "Given the following exponential decay accumulation, what is the final value of total (rounded to nearest integer)?",
 "code": "inputs = [10, 20, 15, 25]\ntotal = 0\ndecay_factor = 0.8\nweight = 1.0\n\nfor value in inputs:\n total += value * weight\n weight *= decay_factor\n\nprint(f\"Total: {round(total)}\")",
 "answer": 46
 }
},
{
 "id": "BL-IT-S008-V009",
 "metadata": {
 "name": "条件重置累积器",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_reset",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following conditional reset accumulation, what is the final value of accumulator?",
 "code": "data = [3, 7, 2, 9, 1, 8, 4, 6]\naccumulator = 0\nreset_threshold = 15\nmax_value = 0\n\nfor value in data:\n accumulator += value\n max_value = max(max_value, value)\n if accumulator >= reset_threshold:\n accumulator = max_value\n max_value = 0\n\nprint(f\"Accumulator: {accumulator}\")",
 "answer": 6
 }
},
{
 "id": "BL-IT-S008-V010",
 "metadata": {
 "name": "多阶段处理器",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
```

```
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multi_stage_processor",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following multi-stage processing, what is the final value of output?",
 "code": "inputs = [4, 9, 2, 7, 1, 5]\\noutput = 0\\nstage = 1\\nstage_counter = 0\\n\\nfor value in inputs:\\n if stage == 1:\\n output += value * 2\\n elif stage == 2:\\n output = output // 10\\n elif stage == 3:\\n output -= value\\n stage_counter += 1\\n if stage_counter >= 2:\\n stage = (stage % 3) + 1\\n stage_counter = 0\\n\\nprint(f\"Output: {output}\")",
 "answer": 8
 }
},
{
 "id": "BL-IT-S008-V011",
 "metadata": {
 "name": "反馈控制累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "feedback_control",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 5
 },
 "task": {
 "description": "Given the following feedback control accumulation, what is the final value of output?",
 "code": "setpoints = [10, 15, 8, 20, 12]\\noutput = 5\\nerror_sum = 0\\nkp, ki = 0.5, 0.1\\n\\nfor setpoint in setpoints:\\n error = setpoint - output\\n error_sum += error\\n control = kp * error + ki * error_sum\\n output += control\\n output = max(0, min(25, output)) # clamp between 0 and 25\\n\\nprint(f\"Output: {round(output, 1)}\")",
 "answer": 16.4
 }
},
{
 "id": "BL-IT-S008-V012",
 "metadata": {
 "name": "分组统计累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
```

```
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "grouped_statistics",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 8
 },
 "task": {
 "description": "Given the following grouped statistics accumulation, what is the final value of max_group_sum?",
 "code": "values = [3, 1, 4, 1, 5, 9, 2, 6]\ngroups = [0, 1, 0, 1, 0, 1, 0, 1]\ngroup_sums = [0, 0]\nmax_group_sum = 0\n\nfor i, value in enumerate(values):\n group = groups[i]\n group_sums[group] += value\n max_group_sum = max(max_group_sum, group_sums[group])\n\n\nprint(f\"Max Group Sum: {max_group_sum}\")",
 "answer": 17
 }
},
{
 "id": "BL-IT-S008-V013",
 "metadata": {
 "name": "时间序列窗口",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "time_series_window",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 7
 },
 "task": {
 "description": "Given the following time series window processing, what is the final value of trend_score?",
 "code": "prices = [100, 105, 98, 110, 95, 115, 102]\ntrend_score = 0\nwindow = []\nwindow_size = 3\n\nfor price in prices:\n window.append(price)\n if len(window) > window_size:\n window.pop(0)\n if len(window) == window_size:\n if window[2] > window[0]: # current > first\n trend_score += 1\n elif window[2] < window[0]:\n trend_score -= 1\n\n\nprint(f\"Trend Score: {trend_score}\")",
 "answer": 0
 }
},
{
 "id": "BL-IT-S008-V014",
 "metadata": {
 "name": "自适应权重累积",
 "category": "Block-Level",
 "subcategory": "Iterative",
 "type": "variant",
```

```

 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "adaptive_weight",
 "loop_pattern": "for",
 "nesting_depth": 1,
 "iteration_count": 6
 },
 "task": {
 "description": "Given the following adaptive weight accumulation, what is the final value of weighted_avg (rounded to 1 decimal)?",
 "code": "measurements = [8.5, 7.2, 9.1, 6.8, 8.9, 7.5]\nweighted_avg =\nmeasurements[0]\nweight = 0.3\nadaptation_rate = 0.05\n\nfor i in range(1,\nlen(measurements)):\n error = abs(measurements[i] - weighted_avg)\n if error > 1.0:\n weight = min(0.8, weight + adaptation_rate)\n else:\n weight = max(0.1,\n weight - adaptation_rate)\n weighted_avg = weight * measurements[i] + (1 - weight)\n* weighted_avg\n\nprint(f\"Weighted Avg: {round(weighted_avg, 1)}\")",
 "answer": 7.9
 }
]

```

## 2D - 大混合 (43)

# 代码块级推理混合变式生成提示词

## 任务目标

基于给定的代码块级推理种子任务，生成融合多种代码块类型的综合变式任务，全面测试大模型对线性代码块、条件代码块和迭代代码块等多种代码块结构的综合理解和交叉应用能力。

## 核心设计理念

不局限于单一代码块类型，而是创造包含多种代码块推理元素的复合场景，测试模型在复杂代码块环境下的综合推理能力，模拟真实编程中常见的混合代码结构。

## 混合变式生成维度

### 1. 线性与条件混合变式

- \*\*顺序+条件分支\*\*：线性语句序列后跟条件判断
- \*\*条件+顺序执行\*\*：条件分支内包含线性语句序列
- \*\*交替混合\*\*：线性语句与条件语句交替出现
- \*\*依赖条件\*\*：线性语句的结果影响后续条件判断
- \*\*条件赋值\*\*：条件分支中的变量赋值影响后续线性计算
- \*\*多路径线性\*\*：不同条件分支包含不同的线性语句序列

### 2. 线性与迭代混合变式

- \*\*循环前初始化\*\*：线性代码为循环做准备和初始化
- \*\*循环后处理\*\*：循环结束后的线性代码处理结果
- \*\*循环内线性\*\*：循环体内包含线性语句序列
- \*\*嵌套计算\*\*：线性计算结果作为循环参数
- \*\*累积线性\*\*：循环的累积结果用于后续线性计算
- \*\*状态传递\*\*：线性代码与循环之间的状态传递

### ### 3. 条件与迭代混合变式

- \*\*条件控制循环\*\*：条件判断决定循环的执行与否
- \*\*循环内条件\*\*：循环体内包含条件分支逻辑
- \*\*条件终止\*\*：复杂条件表达式控制循环终止
- \*\*分支循环\*\*：不同条件分支包含不同的循环结构
- \*\*循环条件更新\*\*：循环过程中更新条件判断的变量
- \*\*条件递归\*\*：条件判断与递归调用的结合

### ### 4. 三重混合变式

- \*\*线性+条件+迭代\*\*：三种代码块类型的完整结合
- \*\*嵌套三重混合\*\*：在循环内包含条件和线性代码
- \*\*序列三重混合\*\*：三种类型按序列顺序组合
- \*\*交错三重混合\*\*：三种类型交错穿插出现
- \*\*递归三重混合\*\*：递归函数内包含条件和线性代码
- \*\*复杂状态管理\*\*：三种类型共同管理复杂程序状态

### ### 5. 控制流复杂混合变式

- \*\*跳转与分支\*\*：`break/continue`与条件分支的组合
- \*\*异常与循环\*\*：异常处理与迭代结构的结合
- \*\*返回与条件\*\*：函数返回与条件判断的混合
- \*\*goto与结构\*\*：`goto`跳转与结构化代码的混合
- \*\*多层跳出\*\*：从深层嵌套结构的复杂跳出
- \*\*状态机模式\*\*：使用混合结构实现状态机

### ### 6. 数据流复杂混合变式

- \*\*多源数据流\*\*：多个数据源通过不同代码块汇聚
- \*\*分流处理\*\*：一个数据源分流到不同的处理路径
- \*\*循环依赖\*\*：不同代码块间形成数据依赖循环
- \*\*状态累积\*\*：跨多种代码块的状态累积过程
- \*\*数据变换管道\*\*：数据经过多种代码块的变换处理
- \*\*反馈循环\*\*：输出结果反馈影响前面的处理逻辑

### ### 7. 内存与资源混合变式

- \*\*资源分配+条件+释放\*\*：资源管理的完整生命周期
- \*\*循环内存管理\*\*：循环中的动态内存分配和释放
- \*\*条件资源分配\*\*：基于条件的选择性资源分配
- \*\*递归资源累积\*\*：递归调用中的资源累积管理
- \*\*内存泄漏检测\*\*：复杂控制流中的内存泄漏问题
- \*\*资源共享\*\*：多个代码块间的资源共享和竞争

### ### 8. 错误处理混合变式

- \*\*多层错误处理\*\*：不同代码块层次的错误处理
- \*\*错误传播\*\*：错误在混合代码结构中的传播
- \*\*恢复策略\*\*：错误发生后的恢复和继续执行
- \*\*资源清理\*\*：异常情况下的资源清理
- \*\*状态一致性\*\*：错误处理中的状态一致性维护
- \*\*回滚机制\*\*：复杂操作的事务性回滚

### ### 9. 算法模式混合变式

- \*\*搜索算法\*\*：结合线性、条件、迭代的搜索实现
- \*\*排序算法\*\*：复杂排序算法的混合代码结构
- \*\*图算法\*\*：图遍历和处理的混合代码实现
- \*\*动态规划\*\*：DP算法的混合代码结构

- **分治算法**: 分治策略的递归和条件混合
- **贪心算法**: 贪心选择的条件和迭代混合

### ### 10. 性能与优化混合变式

- **分支预测**: 条件分支对性能的影响
- **循环展开**: 循环优化与线性代码的关系
- **缓存友好**: 数据访问模式的性能考虑
- **并行化**: 混合代码结构的并行化可能
- **编译器优化**: 编译器对混合结构的优化
- **热点路径**: 性能关键路径的识别和优化

## ## 复杂度层次设计

### ### 简单混合 (Easy)

- 2种代码块类型的基础组合
- 清晰的执行顺序和数据流
- 简单的控制流和状态管理
- 直观的逻辑组合

### ### 中等混合 (Medium)

- 2-3种代码块类型的中度组合
- 包含嵌套结构或中等复杂的控制流
- 涉及状态传递和简单的错误处理
- 需要仔细分析执行路径

### ### 复杂混合 (Hard)

- 3种代码块类型的深度融合
- 复杂的嵌套和控制流结构
- 多层次的状态管理和错误处理
- 涉及算法实现或资源管理

### ### 专家级混合 (Expert)

- 极度复杂的多类型深度嵌套
- 涉及高级算法和性能优化
- 复杂的并发和同步处理
- 需要专业级别的代码理解能力

## ## 生成策略

### ### 种子分析策略

1. **识别主导类型**: 分析种子任务的主要代码块类型 (A/B/C)
2. **提取核心逻辑**: 识别种子中的关键逻辑和计算模式
3. **设计融合点**: 确定可以融合其他代码块类型的切入点
4. **保持核心复杂度**: 在添加新元素时保持原有难度层次

### ### 变式设计原则

1. **渐进式融合**: 从简单混合逐步增加到复杂混合
2. **逻辑连贯性**: 确保混合后的代码逻辑清晰且正确
3. **现实导向**: 优先设计实际编程中常见的混合模式
4. **测试价值**: 每个混合变式都应测试特定的推理能力组合

### ### 质量保证机制

1. **执行路径验证**: 验证所有可能执行路径的正确性

2. **状态一致性检查**: 确保程序状态的一致性维护
3. **边界条件覆盖**: 包含各种边界和异常情况
4. **性能考虑**: 确保代码的合理性和可执行性

## 输出格式要求

```
```json
[
  {
    "id": "BL-MIX-S00X-V001",
    "metadata": {
      "name": "混合代码块变式名称",
      "category": "Block-Level",
      "subcategory": "Mixed",
      "type": "variant",
      "source": "Generated",
      "language": "目标语言",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
      "variant_type": "混合模式标签",
      "mixed_blocks": ["Linear", "Conditional", "Iterative"],
      "primary_pattern": "主导的代码块模式",
      "complexity_factors": ["控制流", "状态管理", "错误处理"]
    },
    "task": {
      "description": "混合代码块推理任务描述",
      "code": "包含多种代码块类型的复合代码",
      "answer": "执行完成后的最终结果"
    }
  },
  {下一个变式...}
]

```

特殊字段说明

mixed_blocks: 标识该变式包含的代码块类型组合

primary_pattern: 标识主导的代码块模式

complexity_factors: 标识增加复杂度的关键因素

生成目标

总共生成80-100个变式

每种难度等级至少包含4-5个变式

涵盖至少10-12种不同的混合模式

包含2-3个跨语言的混合变式

至少包含5-6个三重混合的复杂案例

包含不同复杂度因素的变式组合

重点测试场景

执行路径分析: 在复杂混合结构中正确分析执行路径

状态跟踪能力: 跨多种代码块准确跟踪程序状态

控制流理解: 对复杂控制流的正确理解和预测

边界条件处理: 在混合结构中正确处理边界情况

错误传播理解: 理解错误在混合结构中的传播机制

性能影响分析: 理解不同混合模式对性能的影响

常见混合模式示例

初始化+循环+判断: 经典的算法实现模式

条件+递归+累积：树形算法的常见结构

循环+嵌套条件+状态更新：状态机实现模式

线性预处理+复杂循环+结果处理：数据处理管道

多层条件+循环+异常处理：健壮性代码模式

递归+条件终止+状态回溯：回溯算法模式

请基于此提示词，为给定的代码块级种子任务生成全面的混合变式集合，输出格式为包含所有变式的JSON数组。

```
[  
  {  
    "id": "BL-MIX-S001-V001",  
    "metadata": {  
      "name": "Linear Initialization + Conditional Validation Mixed",  
      "category": "Block-Level",  
      "subcategory": "Mixed",  
      "type": "variant",  
      "source": "Generated",  
      "language": "c",  
      "difficulty": "easy",  
      "intervention": 0,  
      "variant_type": "Linear+Conditional",  
      "mixed_blocks": ["Linear", "Conditional"],  
      "primary_pattern": "Linear",  
      "complexity_factors": ["sequential execution", "conditional branching"]  
    },  
    "task": {  
      "description": "Given the following code block containing linear initialization followed by conditional validation, what is the final value of `result` when `input` is 25?",  
      "code": "int input = 25;\nint result = 0;\nint threshold = 20;\nint multiplier = 3;\n\n// Linear initialization\nresult = input * multiplier;\n\n// Conditional validation\nif (result > threshold) {\n    result = result + 10;\n} else {\n    result = result - 5;\n}\n\nprintf(\"Result: %d\\n\", result);",  
      "answer": 85  
    }  
  },  
  {  
    "id": "BL-MIX-S001-V002",  
    "metadata": {  
      "name": "Linear Computation within Conditional Branches",  
      "category": "Block-Level",  
      "subcategory": "Mixed",  
      "type": "variant",  
      "source": "Generated",  
      "language": "python",  
      "difficulty": "easy",  
      "intervention": 0,  
      "variant_type": "Conditional+Linear",  
      "mixed_blocks": ["Conditional", "Linear"],  
      "primary_pattern": "Conditional",  
      "complexity_factors": ["conditional logic", "linear computation"]  
    },  
  },  
]
```

```
"task": {
    "description": "Given the following code block containing linear computation within conditional branches, what is the final value of `final_grade` when `score` is 88?",
    "code": "score = 88\nbonus_points = 5\npenalty = 2\n\nif score >= 85:\n    # Linear computation sequence\n    adjusted_score = score + bonus_points\n    final_grade = adjusted_score * 1.1\n    final_grade = int(final_grade)\nelse:\n    adjusted_score = score - penalty\n    final_grade = adjusted_score * 0.9\n    final_grade = int(final_grade)\n\nprint(f\"Final grade: {final_grade}\")",
    "answer": 102
},
{
    "id": "BL-MIX-S001-V003",
    "metadata": {
        "name": "Linear Preparation + Loop Processing",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "Linear+Iterative",
        "mixed_blocks": ["Linear", "Iterative"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["linear initialization", "loop accumulation"]
    },
    "task": {
        "description": "Given the following code block containing linear preparation followed by loop processing, what is the final value of `sum`?",
        "code": "int base = 10;\nint increment = 3;\nint limit = 5;\nint sum = base;\n\nfor (int i = 1; i <= limit; i++)\n    sum += i * increment;\n\nprintf(\"Sum: %d\\n\", sum);",
        "answer": 55
    }
},
{
    "id": "BL-MIX-S001-V004",
    "metadata": {
        "name": "Conditional Control within Loop",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "Iterative+Conditional",
        "mixed_blocks": ["Iterative", "Conditional"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["loop control", "conditional branching"]
    },
    "task": {

```

```

    "description": "Given the following code block containing conditional control within a loop, what is the final value of `count`?",
    "code": "numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]\ncount = 0\nntotal = 0\n\nfor num in numbers:\n    if num % 2 == 0:\n        total += num\n        count += 1\n    else:\n        total -= num\n\nprint(f\"Count: {count}, Total: {total}\")",
    "answer": 5
}
},
{
    "id": "BL-MIX-S001-V005",
    "metadata": {
        "name": "Triple Mixed: Initialization + Conditional + Loop",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Linear+Conditional+Iterative",
        "mixed_blocks": ["Linear", "Conditional", "Iterative"],
        "primary_pattern": "Mixed",
        "complexity_factors": ["linear initialization", "conditional validation", "loop processing"]
    },
    "task": {
        "description": "Given the following triple mixed code block containing linear initialization, conditional validation, and loop processing, what is the final value of `result` when `input` is 7?",
        "code": "int input = 7;\nint result = 0;\nint base = 5;\nint factor = 2;\n\n// Linear initialization\nresult = input + base;\n\n// Conditional validation\nif (result > 10)\n    // Loop processing\n    for (int i = 1; i <= 3; i++) {\n        result *= factor;\n    }\nelse {\n    result = result * 10;\n}\n\nprintf(\"Result: %d\\n\", result);",
        "answer": 96
    }
},
{
    "id": "BL-MIX-S001-V006",
    "metadata": {
        "name": "Conditional Dependent Linear Computation Chain",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "Conditional+Linear",
        "mixed_blocks": ["Conditional", "Linear"],
        "primary_pattern": "Linear",
        "complexity_factors": ["conditional selection", "linear dependency"]
    },
    "task": {

```

```

        "description": "Given the following conditional dependent linear computation chain, what is the final value of `final_value` when `x` is 15?",  

        "code": "x = 15\nthreshold = 12\n\n# Conditional base value selection\nif x > threshold:\n    base_value = x * 2\nelse:\n    base_value = x + 10\n\n# Linear computation chain\nstep1 = base_value + 5\nstep2 = step1 * 1.5\nfinal_value = int(step2 - 3)\n\nprint(f\"Final value: {final_value}\")",
        "answer": 49
    }
},
{
    "id": "BL-MIX-S001-V007",
    "metadata": {
        "name": "Nested Loops with Conditional Filtering",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Iterative+Conditional",
        "mixed_blocks": ["Iterative", "Conditional"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["nested loops", "conditional filtering"]
    },
    "task": {
        "description": "Given the following code block with nested loops and conditional filtering, what is the final value of `valid_count`?",  

        "code": "int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};\nint valid_count = 0;\nint threshold = 5;\n\nfor (int i = 0; i < 3; i++) {\n    for (int j = 0; j < 3; j++) {\n        if (matrix[i][j] > threshold) {\n            valid_count++;\n        }\n    }\n}\n\nprintf(\"valid count: %d\\n\", valid_count);",
        "answer": 4
    }
},
{
    "id": "BL-MIX-S001-V008",
    "metadata": {
        "name": "Loop Accumulation with Linear Transformation",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "Iterative+Linear",
        "mixed_blocks": ["Iterative", "Linear"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["loop accumulation", "linear transformation"]
    },
    "task": {

```

```

    "description": "Given the following code block with loop accumulation followed by linear transformation, what is the final value of `final_result`?",

    "code": "data = [2, 4, 6, 8, 10]\naccumulator = 0\n\n# Loop accumulation\nfor value in data:\n    accumulator += value * 2\n\n# Linear transformation\nintermediate = accumulator / 2\nbonus = 15\nfinal_result = int(intermediate + bonus)\n\nprint(f\"Final result: {final_result}\")",
    "answer": 45
}

},
{
    "id": "BL-MIX-S001-V009",
    "metadata": {
        "name": "Multi-Branch Linear Processing",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Conditional+Linear",
        "mixed_blocks": ["Conditional", "Linear"],
        "primary_pattern": "Conditional",
        "complexity_factors": ["multi-way branching", "linear processing"]
    },
    "task": {
        "description": "Given the following multi-branch linear processing code block, what is the final value of `final_score` when `grade` is 78?",

        "code": "int grade = 78;\nint final_score = 0;\nint bonus = 0;\nint penalty = 0;\n\nif (grade >= 90) {\n    bonus = 20;\n    penalty = 0;\n    final_score = grade + bonus;\n} else if (grade >= 80) {\n    bonus = 10;\n    penalty = 2;\n    final_score = grade + bonus - penalty;\n} else if (grade >= 70) {\n    bonus = 5;\n    penalty = 5;\n    final_score = grade + bonus - penalty;\n} else {\n    bonus = 0;\n    penalty = 10;\n    final_score = grade - penalty;\n}\n\nprintf(\"Final score: %d\\n\", final_score);",
        "answer": 78
    }
},
{
    "id": "BL-MIX-S001-V010",
    "metadata": {
        "name": "Conditional Loop Termination Control",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "Conditional+Iterative",
        "mixed_blocks": ["Conditional", "Iterative"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["conditional termination", "loop control"]
    }
},

```

```

"task": {
    "description": "Given the following code block with conditional loop termination control, what is the final value of `iterations`?",
    "code": "int value = 100;\nint iterations = 0;\nint threshold = 10;\n\nwhile (value > threshold) {\n    iterations++;\n    value = value / 2;\n    if (iterations >= 5) {\n        break;\n    }\n}\nprintf(\"Iterations: %d, Final value: %d\\n\", iterations, value);",
    "answer": 4
},
{
    "id": "BL-MIX-S001-V011",
    "metadata": {
        "name": "Linear Preprocessing + Complex Loop",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Linear+Iterative",
        "mixed_blocks": ["Linear", "Iterative"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["linear preprocessing", "complex loop"]
    },
    "task": {
        "description": "Given the following code block with linear preprocessing and complex loop, what is the final value of `processed_count`?",
        "code": "# Linear preprocessing\nraw_data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]\nfilter_threshold = 5\nmultiplier = 2\nadjustment = 3\n\n# Complex loop\nprocessed_count = 0\nsum_processed = 0\nfor item in raw_data:\n    if item > filter_threshold:\n        processed_value = item * multiplier + adjustment\n        sum_processed += processed_value\n        processed_count += 1\n\nprint(f\"Processed count: {processed_count}\")",
        "answer": 5
    }
},
{
    "id": "BL-MIX-S001-V012",
    "metadata": {
        "name": "Recursive Call with Conditional Logic",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "Iterative+Conditional",
        "mixed_blocks": ["Iterative", "Conditional"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["recursive calls", "conditional logic"]
    }
}

```

```

},
"task": {
  "description": "Given the following code block with recursive calls and conditional logic, what is the return value of `fibonacci(6)`?",
  "code": "int fibonacci(int n) {\n    // Base conditions\n    if (n <= 1) {\n        return n;\n    }\n    // Conditional optimization\n    if (n == 2) {\n        return 1;\n    }\n    // Recursive calls\n    return fibonacci(n - 1) + fibonacci(n - 2);\n}\n\nint result = fibonacci(6);\nprintf(\"Fibonacci(6): %d\\n\", result);",
  "answer": 8
}
},
{
  "id": "BL-MIX-S001-V013",
  "metadata": {
    "name": "Alternating Linear and Conditional Processing",
    "category": "Block-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "Linear+Conditional",
    "mixed_blocks": ["Linear", "Conditional"],
    "primary_pattern": "Mixed",
    "complexity_factors": ["alternating patterns", "state transfer"]
  },
  "task": {
    "description": "Given the following code block with alternating linear and conditional processing, what is the final value of `result` when `input` is 12?",
    "code": "input = 12\nresult = input\n\n# First linear processing\nresult = result * 2\n\n# First conditional processing\nif result > 20:\n    result = result - 5\nelse:\n    result = result + 3\n\n# Second linear processing\nresult = result + 10\n\n# Second conditional processing\nif result % 2 == 0:\n    result = result / 2\nelse:\n    result = result * 3\n\nprint(f\"Result: {result}\")",
    "answer": 17.0
}
},
{
  "id": "BL-MIX-S001-V014",
  "metadata": {
    "name": "Loop with Nested Conditionals and Linear Operations",
    "category": "Block-Level",
    "subcategory": "Mixed",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "Iterative+Conditional+Linear",
    "mixed_blocks": ["Iterative", "Conditional", "Linear"],
    "primary_pattern": "Iterative",
    "complexity_factors": ["nested loops", "conditional branching", "linear operations"]
  }
}

```

```

        "complexity_factors": ["nested loops", "conditional branches", "linear
computation"]
    },
    "task": {
        "description": "Given the following code block with loop containing nested
conditionals and linear operations, what is the final value of `final_sum`?",
        "code": "int arr[] = {2, 5, 8, 3, 9, 1, 7, 4};\nint size = 8;\nint final_sum =
0;\nint even_multiplier = 3;\nint odd_multiplier = 2;\n\nfor (int i = 0; i < size; i++) {\n    int current = arr[i];\n    // Conditional logic\n    if (current % 2 == 0) {\n        // Linear computation (even)\n        current = current * even_multiplier;\n    }\n    else {\n        // Linear computation (odd)\n        current = current * odd_multiplier;\n    }\n    final_sum += current;\n}\n\nprintf(\"Final sum: %d\\n\", final_sum);",
        "answer": 119
    }
},
{
    "id": "BL-MIX-S001-V015",
    "metadata": {
        "name": "Branch Selection for Different Loops",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Conditional+Iterative",
        "mixed_blocks": ["Conditional", "Iterative"],
        "primary_pattern": "Conditional",
        "complexity_factors": ["branch selection", "different loops"]
    },
    "task": {
        "description": "Given the following code block with branch selection for
different loops, what is the final value of `result` when `mode` is 'fast'?",
        "code": "mode = 'fast'\n\nfor i in range(len(data)):\n    for j in
range(i + 1):\n        result += data[j]\n\nif mode == 'slow':\n    # Slow processing loop\n    for i in range(len(data)):\n        for j in
range(i + 1):\n            result += data[j]\n\nif mode == 'fast':\n    # Fast processing
loop\n    for item in data:\n        result += item * 2\n\nelse:\n    # Default processing
loop\n    for item in data:\n        result += item\n\nprint(f\"Result: {result}\")",
        "answer": 30
    }
},
{
    "id": "BL-MIX-S001-V016",
    "metadata": {
        "name": "State Machine Pattern Mix",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
    }
}

```

```

    "intervention": 3,
    "variant_type": "Linear+Conditional+Iterative",
    "mixed_blocks": ["Linear", "Conditional", "Iterative"],
    "primary_pattern": "Mixed",
    "complexity_factors": ["state machine", "state transitions", "loop control"]
},
"task": {
    "description": "Given the following state machine pattern mixed code block, what is the final value of `final_state` when the input string is \"ABBC\"?",
    "code": "typedef enum { IDLE = 0, STATE_A = 1, STATE_B = 2, ERROR = -1 } state_t;\n\nstate_t process_string(const char* input) {\n    state_t current_state = IDLE;\n\n    int position = 0;\n\n    // Loop through each character\n    while (input[position] != '\\0') {\n        char ch = input[position];\n\n        // State transition conditions\n        switch (current_state) {\n            case IDLE:\n                if (ch == 'A') current_state = STATE_A;\n                else current_state = ERROR;\n                break;\n            case STATE_A:\n                if (ch == 'B') current_state = STATE_B;\n                else current_state = ERROR;\n                break;\n            case STATE_B:\n                if (ch == 'B' || ch == 'C') current_state = STATE_B;\n                else current_state = ERROR;\n                break;\n            default:\n                current_state = ERROR;\n        }\n\n        // Linear increment\n        position++;\n\n        // Error check\n        if (current_state == ERROR) break;\n    }\n\n    return current_state;\n}\n\nstate_t final_state = process_string(\"ABBC\");\nprintf(\"Final state: %d\\n\", final_state);",
    "answer": 2
}
},
{
    "id": "BL-MIX-S001-V017",
    "metadata": {
        "name": "Data Transformation Pipeline",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Linear+Conditional+Iterative",
        "mixed_blocks": ["Linear", "Conditional", "Iterative"],
        "primary_pattern": "Linear",
        "complexity_factors": ["data pipeline", "multi-stage processing"]
},
    "task": {
        "description": "Given the following data transformation pipeline code block, what is the length of `processed_data`?",
        "code": "raw_data = [10, 25, 8, 33, 15, 42, 7, 28]\n\n# Stage 1: Linear preprocessing\nscale_factor = 1.2\noffset = 5\nstage1_data = [int(x * scale_factor + offset) for x in raw_data]\n\n# Stage 2: Conditional filtering\nthreshold = 20\nstage2_data = []\nfor value in stage1_data:\n    if value >= threshold:\n        stage2_data.append(value)\n\n# Stage 3: Loop transformation\nprocessed_data = []\nfor i, value in enumerate(stage2_data):\n    if i % 2 == 0:\n        processed_data.append(value * 2)\n    else:\n        processed_data.append(value + 10)\n\nprint(f\"Processed data length: {len(processed_data)}\")",
    }
}

```

```

        "answer": 6
    }
},
{
    "id": "BL-MIX-S001-V018",
    "metadata": {
        "name": "Nested Function Call Mix",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "Iterative+Conditional+Linear",
        "mixed_blocks": ["Iterative", "Conditional", "Linear"],
        "primary_pattern": "Iterative",
        "complexity_factors": ["function nesting", "recursive calls", "multiple
conditions"]
    },
    "task": {
        "description": "Given the following nested function call mixed code block, what
is the return value of `calculate_sum(4, 3)`?",
        "code": "int process_value(int x, int depth) {\n    // Linear base calculation\n    int base = x * 2 + 1;\n    // Conditional recursion\n    if (depth > 0) {\n        return base + process_value(x - 1, depth - 1);\n    } else {\n        return base;\n    }\n}\n\nint calculate_sum(int n, int depth) {\n    int total = 0;\n    // Loop calls\n    for (int i = 1; i <= n; i++) {\n        int partial = process_value(i, depth);\n        total += partial;\n    }\n    return total;\n}\n\nint result = calculate_sum(4, 3);\nprintf(\"Result: %d\\n\", result);",
        "answer": 80
    }
},
{
    "id": "BL-MIX-S001-V019",
    "metadata": {
        "name": "Resource Management Mixed Pattern",
        "category": "Block-Level",
        "subcategory": "Mixed",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "Linear+Conditional+Iterative",
        "mixed_blocks": ["Linear", "Conditional", "Iterative"],
        "primary_pattern": "Mixed",
        "complexity_factors": ["resource allocation", "conditional release", "loop
management"]
    },
    "task": {
        "description": "Given the following resource management mixed pattern code
block, what is the final value of `allocated_count` when `capacity` is 5?",
    }
}

```

```

        "code": "typedef struct {\n    int* ptr;\n    int size;\n    int allocated;\n} resource_t;\n\nint manage_resources(int capacity) {\n    resource_t resources[10];\n    int allocated_count = 0;\n\n    // Linear initialization\n    for (int i = 0; i < 10; i++) {\n        resources[i].ptr = NULL;\n        resources[i].size = 0;\n        resources[i].allocated = 0;\n\n        // Loop allocation\n        for (int j = 0; j < capacity && j < 10; j++) {\n            int size = (j + 1) * 16;\n\n            // Conditional allocation\n            if (size <= 64) {\n                resources[i].ptr = malloc(size);\n                if (resources[i].ptr != NULL) {\n                    resources[i].size = size;\n                    resources[i].allocated = 1;\n                    allocated_count++;\n                }\n            }\n\n            // Conditional release (simulated, not actually freeing)\n            for (int k = 0; k < 10; k++) {\n                if (resources[k].allocated && resources[k].size > 32) {\n                    // This is just counting, not actually freeing\n                    free(resources[k].ptr);\n                    //\n                    allocated_count--;\n                }\n            }\n\n            //\n            return allocated_count;\n        }\n    }\n\n    //\n    nprintf(\"Allocated count: %d\\n\", result);\n\n    \"answer\": 4\n}\n},\n{\n    \"id\": \"BL-MIX-S001-V020\",\n    \"metadata\": {\n        \"name\": \"Multi-Level Break Control\",\n        \"category\": \"Block-Level\",\n        \"subcategory\": \"Mixed\",\n        \"type\": \"variant\",\n        \"source\": \"Generated\",\n        \"language\": \"python\",\n        \"difficulty\": \"hard\",\n        \"intervention\": 2,\n        \"variant_type\": \"Iterative+Conditional\",\n        \"mixed_blocks\": [\"Iterative\", \"Conditional\"],\n        \"primary_pattern\": \"Iterative\",\n        \"complexity_factors\": [\"multi-level loops\", \"break control\"]\n    },\n    \"task\": {\n        \"description\": \"Given the following multi-level break control code block, what is the final value of `found_count`?\",\n        \"code\": \"matrix = [\n            [1, 2, 3, 15],\n            [4, 5, 6, 16],\n            [7, 8, 9,\n            17],\n            [10, 11, 12, 18]]\n\n        target = 15\n        found_count = 0\n        found = False\n\n        for i in range(len(matrix)):\n            if found:\n                break\n            for j in range(len(matrix[i])):\n                current = matrix[i][j]\n                if current == target:\n                    found = True\n                found_count += 1\n                break\n                elif current > 10:\n                    found_count\n\n        += 1\n\n        nprintf(f\"Found count: {found_count}\")\", \n\n        \"answer\": 4\n    }\n}\n]

```

3 - 代码属性推理 (283)

3A - 循环属性 [Loop] (52)

```
# 循环属性推理变式生成提示词
```

任务目标

基于给定的循环属性推理种子任务，生成多样化的变式任务来全面测试大模型对循环结构深层属性的理解能力，包括迭代计数分析、变量状态追踪、终止条件判断等核心循环属性推理技能。

循环属性推理特征分析

循环属性推理关注循环结构的内在特性和行为模式，重点测试模型对循环执行过程中数量关系、状态变化和控制逻辑的深层理解，而非仅仅关注最终结果。

变式生成维度

1. 迭代计数分析变式

- **简单计数变式**：基础`for/while`循环的迭代次数计算
- **嵌套计数变式**：多层嵌套循环的总迭代次数分析
- **条件计数变式**：带有条件跳过的实际执行次数
- **动态计数变式**：循环边界动态变化的计数分析
- **递归计数变式**：递归调用的深度和总调用次数
- **部分计数变式**：特定条件下的子集迭代计数

2. 变量状态追踪变式

- **累积器追踪变式**：累积变量在特定迭代的状态值
- **计数器追踪变式**：计数器变量的演变过程
- **状态机追踪变式**：状态变量在迭代中的转换
- **数组索引追踪变式**：数组访问索引的变化追踪
- **指针追踪变式**：指针变量在循环中的移动
- **多变量追踪变式**：多个相关变量的同步状态追踪

3. 终止条件分析变式

- **单一条件变式**：简单终止条件的判断分析
- **复合条件变式**：多个条件组合的终止判断
- **动态条件变式**：终止条件在循环中动态变化
- **收敛条件变式**：数值收敛算法的终止判断
- **多出口条件变式**：多个可能出口的条件分析
- **异常终止变式**：异常情况导致的提前终止

4. 循环不变量变式

- **数学不变量变式**：数学关系在循环中的保持
- **逻辑不变量变式**：逻辑条件在循环中的维持
- **数据结构不变量变式**：数据结构性质的保持
- **资源不变量变式**：资源状态的一致性维护
- **关系不变量变式**：变量间关系的保持
- **边界不变量变式**：边界条件的维护

5. 循环复杂度分析变式

- **时间复杂度变式**：循环的时间复杂度分析
- **空间复杂度变式**：循环中内存使用的分析
- **嵌套复杂度变式**：嵌套循环的复杂度计算
- **最坏情况变式**：最坏执行情况的复杂度

- **平均情况变式**：平均执行情况的复杂度
- **最优情况变式**：最优执行情况的复杂度

6. 循环模式识别变式

- **累积模式变式**：累积计算模式的识别
- **搜索模式变式**：搜索算法模式的分析
- **变换模式变式**：数据变换模式的识别
- **过滤模式变式**：数据过滤模式的分析
- **映射模式变式**：数据映射模式的识别
- **归约模式变式**：数据归约模式的分析

7. 循环边界分析变式

- **边界计算变式**：循环边界的精确计算
- **越界检测变式**：数组越界的可能分析
- **边界调整变式**：循环边界的动态调整
- **边界优化变式**：循环边界的优化分析
- **边界错误变式**：常见边界错误的识别
- **边界特例变式**：特殊边界情况的处理

8. 循环依赖分析变式

- **数据依赖变式**：循环中的数据依赖关系
- **控制依赖变式**：控制流的依赖分析
- **循环依赖变式**：循环迭代间的依赖关系
- **函数依赖变式**：函数调用的依赖分析
- **内存依赖变式**：内存访问的依赖关系
- **资源依赖变式**：资源使用的依赖分析

9. 循环优化分析变式

- **循环展开变式**：循环展开的效果分析
- **循环合并变式**：多个循环的合并可能
- **循环交换变式**：嵌套循环的交换优化
- **循环分裂变式**：循环分裂的优化效果
- **循环向量化变式**：循环向量化的可能
- **循环并行化变式**：循环并行化的分析

10. 高级循环属性变式

- **循环不动点变式**：不动点的识别和分析
- **循环周期性变式**：循环行为的周期性分析
- **循环收敛性变式**：数值算法的收敛性分析
- **循环稳定性变式**：算法稳定性的分析
- **循环正确性变式**：循环正确性的验证
- **循环等价性变式**：不同循环实现的等价性

复杂度层次设计

简单属性 (Easy)

- 单层循环的基础属性分析
- 简单的计数和状态追踪
- 明确的终止条件判断
- 直观的循环模式识别

中等属性 (Medium)

- 2层嵌套循环的属性分析

- 涉及条件判断的复杂计数
- 多变量的状态追踪
- 中等复杂度的终止条件

复杂属性 (Hard)

- 3层以上嵌套或复杂循环结构
- 动态边界和复杂状态管理
- 多重终止条件的分析
- 涉及算法优化的属性分析

专家级属性 (Expert)

- 极度复杂的循环属性分析
- 涉及高级算法和数值计算
- 复杂的不变量和收敛性分析
- 需要深度数学和算法知识

生成策略

种子分析策略

1. **识别核心属性**: 分析种子任务关注的主要循环属性
2. **提取分析模式**: 识别属性分析的方法和思路
3. **确定关键参数**: 找出影响循环属性的关键参数
4. **分析复杂度层次**: 评估当前任务的复杂度水平

变式设计原则

1. **属性聚焦**: 每个变式都应密切关注特定的循环属性
2. **分析深度**: 注重对循环内在机制的深层分析
3. **量化精确**: 提供精确的数量分析和状态描述
4. **实用导向**: 关注实际编程中重要的循环属性

质量保证

1. **计算验证**: 手工验证关键计算步骤的正确性
2. **属性一致性**: 确保循环属性的逻辑一致性
3. **边界检查**: 验证边界条件的处理正确性
4. **复杂度合理**: 确保复杂度分析的合理性

输出格式要求

```
```json
[
 {
 "id": "PL-LP-S00X-V001",
 "metadata": {
 "name": "循环属性变式名称",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "目标语言",
 "difficulty": "easy/medium/hard/expert",
 "intervention": 0,
 "variant_type": "变式类型标签",
 }
 }
]
```

```

 "property_focus": "iteration_count/variable_tracking/termination_condition/invariant/complexity",
 "analysis_type": "quantitative/qualitative/comparative"
},
"task": {
 "description": "循环属性分析任务描述（英文）",
 "code": "待分析的循环代码",
 "answer": "特定属性的分析结果"
}
},
{下一个变式...}
]

```

特殊字段说明

**property\_focus**: 标识主要关注的循环属性类型

**analysis\_type**: 标识分析的类型（定量/定性/比较）

**variant\_type**: 描述具体的变式类型

生成目标

为每个提供的种子任务生成12-18个循环属性变式，确保：

每种难度等级至少包含3-4个变式

涵盖至少8-10种不同的属性分析类型

包含不同的分析深度和复杂度

至少包含3-4个涉及高级循环属性的变式

包含定量和定性分析的平衡组合

重点测试场景

精确计数能力：在复杂循环结构中准确计算迭代次数

状态追踪能力：准确追踪变量在特定迭代的状态

条件分析能力：正确分析复杂终止条件的触发时机

模式识别能力：识别循环中的计算模式和数据流

复杂度分析能力：分析循环的时间和空间复杂度

不变量理解：理解和验证循环不变量

常见循环属性模式

线性累积：`sum += i` (等差数列求和)

指数增长：`value *= 2` (几何级数)

收敛迭代：Newton方法、二分查找等

条件计数：满足特定条件的元素计数

状态转换：有限状态机的状态转换

数据变换：数组元素的逐个变换处理

请基于此提示词，为给定的循环属性推理种子任务生成全面的变式集合，输出格式为包含所有变式的JSON数组。

```

[
{
 "id": "PL-LP-S001-V001",
 "metadata": {
 "name": "Simple Triangle Nested Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 }
}
]
```

```
 "variant_type": "简单计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this simple triangular nested loop structure and calculate the total number of inner loop executions.",
 "code": "int count = 0;\nfor (int i = 1; i <= 3; i++) {\n for (int j = 1; j <= i; j++) {\n count++;\n printf(\"i=%d, j=%d\\n\", i, j);\n }\n}\nprintf(\"Total iterations: %d\\n\", count);",
 "answer": 6
 }
},
{
 "id": "PL-LP-S001-V002",
 "metadata": {
 "name": "Dynamic Boundary Nested Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "动态计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this nested loop with dynamic boundaries, calculate the total number of inner loop executions where the outer loop's upper bound changes after each iteration.",
 "code": "int total = 0;\nint upper_bound = 5;\nfor (int i = 1; i <= upper_bound; i++) {\n for (int j = i; j <= upper_bound; j++) {\n total++;\n printf(\"i=%d, j=%d\\n\", i, j);\n }\n upper_bound--; // Dynamically decrease upper bound\n}\nprintf(\"Total iterations: %d\\n\", total);",
 "answer": 9
 }
},
{
 "id": "PL-LP-S001-V003",
 "metadata": {
 "name": "Conditional skip Counting Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 }
}
```

```

},
"task": {
 "description": "In this nested loop with conditional skipping, calculate the number of times the loop body is actually executed (excluding iterations skipped by continue).",
 "code": "int executed = 0;\nfor (int i = 1; i <= 4; i++) {\n for (int j = 1; j <= 5; j++) {\n if ((i + j) % 3 == 0) {\n continue; // Skip certain iterations\n }\n executed++;\n printf(\"Executed: i=%d, j=%d\\n\", i, j);\n }\n}\nprintf(\"Actually executed: %d\\n\", executed);",
 "answer": 14
}
},
{
 "id": "PL-LP-S001-V004",
 "metadata": {
 "name": "Triple Nested Complex Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "嵌套计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this triple nested loop structure and determine the total number of executions of the innermost loop body.",
 "code": "int total = 0;\nfor (int i = 1; i <= 3; i++) {\n for (int j = i; j <= 4; j++) {\n for (int k = 1; k <= (j - i + 1); k++) {\n total++;\n printf(\"i=%d, j=%d, k=%d\\n\", i, j, k);\n }\n }\n}\nprintf(\"Total innermost iterations: %d\\n\", total);",
 "answer": 20
 }
},
{
 "id": "PL-LP-S001-V005",
 "metadata": {
 "name": "Recursive Iteration Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "递归计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {

```

```
 "description": "Calculate the total number of recursive calls made by this function when called with initial value n=4.",
 "code": "int call_count = 0;\n\nint recursive_sum(int n) {\n call_count++;\n printf(\"Call %d: recursive_sum(%d)\\n\", call_count, n);\n if (n <= 1) {\n return n;\n }\n return n + recursive_sum(n-1) + recursive_sum(n-2);\n}\n\n//\ncalled with n=4\nint result = recursive_sum(4);\nprintf(\"Total recursive calls: %d\\n\", call_count);",
 "answer": 15
 },
 {
 "id": "PL-LP-S001-V006",
 "metadata": {
 "name": "Matrix Diagonal Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "部分计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this matrix traversal loop, how many times are diagonal elements (where i == j) processed?",
 "code": "int diagonal_count = 0;\nint matrix[5][5];\nfor (int i = 0; i < 5; i++) {\n for (int j = 0; j < 5; j++) {\n if (i == j) {\n diagonal_count++;\n printf(\"Diagonal element at (%d,%d)\\n\", i, j);\n }\n }\n}\nprintf(\"Processing element (%d,%d)\\n\", i, j);\nprintf(\"Diagonal elements processed: %d\\n\", diagonal_count);",
 "answer": 5
 },
 },
 {
 "id": "PL-LP-S001-V007",
 "metadata": {
 "name": "Fibonacci Loop Iteration Count",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "简单计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
```

```
 "description": "Calculate how many iterations this Fibonacci generation loop executes to reach the first Fibonacci number greater than 100.",
 "code": "int a = 0, b = 1, temp;\nint iterations = 0;\n\nwhile (b <= 100) {\n iterations++;\n printf(\"Iteration %d: Fibonacci = %d\\n\", iterations, b);\n temp = a + b;\n a = b;\n b = temp;\n}\nprintf(\"Total iterations: %d\\n\", iterations);\nprintf(\"First Fibonacci > 100: %d\\n\", b);",
 "answer": 12
 }
},
{
 "id": "PL-LP-S001-V008",
 "metadata": {
 "name": "Early Break Counting Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this nested loop with early break conditions, how many total iterations of the inner loop are executed before the break occurs?",
 "code": "int total_inner = 0;\nint target_sum = 45;\nint current_sum = 0;\n\nfor (int i = 1; i <= 10; i++) {\n for (int j = 1; j <= 8; j++) {\n total_inner++;\n current_sum += (i * j);\n printf(\"Inner iteration %d: i=%d, j=%d, sum=%d\\n\", total_inner, i, j, current_sum);\n if (current_sum >= target_sum) {\n printf(\"Target reached! Breaking...\\n\");\n goto exit_loops;\n }\n }\n}\n\nexit_loops:\nprintf(\"Total inner iterations: %d\\n\", total_inner);",
 "answer": 8
 }
},
{
 "id": "PL-LP-S001-V009",
 "metadata": {
 "name": "Geometric Series Loop Count",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "动态计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
```

```
 "description": "This loop implements a geometric series with variable step size. Calculate the total number of iterations executed.",
 "code": "double value = 1.0;\ndouble multiplier = 1.5;\nint iterations = 0;\n\ndouble threshold = 1000.0;\n\nwhile (value < threshold) {\n iterations++;\n value *= multiplier;\n multiplier += 0.1; // Increasing multiplier\n printf(\"Iteration %d: value=%.\n2f, multiplier=%.\n2f\\n\", iterations, value, multiplier);\n}\n\nprintf(\"Total iterations: %d\\n\", iterations);\nprintf(\"Final value: %.2f\\n\", value);",
 "answer": 8
 },
},
{
 "id": "PL-LP-S002-v001",
 "metadata": {
 "name": "Simple Accumulator Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the accumulator variable 'sum' and determine its value at the start of the 4th iteration.",
 "code": "int sum = 10;\n\nfor (int i = 1; i <= 6; i++) {\n printf(\"Iteration %d start: sum = %d\\n\", i, sum);\n sum += i * multiplier;\n multiplier++;\n printf(\"Iteration %d end: sum = %d\\n\", i, sum);\n}",
 "answer": 28
 },
},
{
 "id": "PL-LP-S002-v002",
 "metadata": {
 "name": "Multi-Variable State Synchronization",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "多变量追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this complex state tracking loop, what is the value of variable 'balance' at the start of the 5th iteration?",
 "code": "int balance = 0;\n\nfor (int i = 1; i <= 5; i++) {\n balance += i;\n printf(\"Iteration %d start: balance = %d\\n\", i, balance);\n balance -= i;\n printf(\"Iteration %d end: balance = %d\\n\", i, balance);\n}"
 }
}
```

```
 "code": "double balance = 1000.0;\ndouble rate = 0.05;\nint transaction_count = 0;\n\nfor (int month = 1; month <= 8; month++) {\n printf(\"Month %d start: balance = %.2f\\n\", month, balance);\n balance *= (1 + rate); // Apply interest\n transaction_count += month;\n balance -= (transaction_count * fee); // Deduct fees\n rate -= 0.005; // Decrease rate\n printf(\"Month %d end: balance = %.2f\\n\", month, balance);\n}\n\n \"answer\": 1039.48\n }\n},\n{\n \"id\": \"PL-LP-S002-V003\",\n \"metadata\": {\n \"name\": \"Array Index Pointer Tracking\",\n \"category\": \"Property-Level\",\n \"subcategory\": \"Loop\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"medium\",\n \"intervention\": 0,\n \"variant_type\": \"数组索引追踪变式\",\n \"property_focus\": \"variable_tracking\",\n \"analysis_type\": \"quantitative\"\n },\n \"task\": {\n \"description\": \"In this array processing loop, what is the value of the array element pointed to by 'ptr' during the 6th iteration?\",\n \"code\": \"int arr[] = {5, 12, 8, 23, 16, 31, 7, 19, 25, 14};\nint *ptr = arr;\nint step = 1;\n\nfor (int i = 0; i < 8; i++) {\n printf(\"Iteration %d: ptr points to arr[%d] = %d\\n\", i+1, ptr-arr, *ptr);\n ptr += step;\n if (i % 2 == 1) step++;\n}\n// Increase step every two iterations\\n}\",\n \"answer\": 7\n }\n},\n{\n \"id\": \"PL-LP-S002-V004\",\n \"metadata\": {\n \"name\": \"State Machine Variable Tracking\",\n \"category\": \"Property-Level\",\n \"subcategory\": \"Loop\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"expert\",\n \"intervention\": 0,\n \"variant_type\": \"状态机追踪变式\",\n \"property_focus\": \"variable_tracking\",\n \"analysis_type\": \"quantitative\"\n },\n \"task\": {\n \"description\": \"what is the value of the 'state' variable at the start of the 7th iteration in this state machine?\",\n }\n}
```

```

 "code": "int state = 0; // 0=IDLE, 1=ACTIVE, 2=PROCESSING, 3=WAITING\nint
counter = 0;\nint threshold = 3;\n\nfor (int cycle = 1; cycle <= 10; cycle++) {\nprintf(\"Cycle %d start: state = %d, counter = %d\\n\", cycle, state, counter);\n\n
switch(state) {\n case 0: // IDLE\n state = 1; counter = 0; break;\n case 1: // ACTIVE\n counter++; if (counter >= threshold) { state
= 2; counter = 0; }\n break;\n case 2: // PROCESSING\n counter++; if (counter >= 2) { state = 3; counter = 0; } \n break;\n case 3: // WAITING\n counter++; if (counter >= 1) { state =
0; counter = 0; }\n break;\n }\n}\n\n
"answer": 1
}

},
{
 "id": "PL-LP-S002-V005",
 "metadata": {
 "name": "Fibonacci Sequence Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "计数器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the Fibonacci sequence generation and determine the value
of variable 'current' at the start of the 6th iteration.",
 "code": "int prev = 0, current = 1, next;\nint iteration = 1;\n\nwhile
(iteration <= 8) {\n printf(\"Iteration %d start: prev=%d, current=%d\\n\", iteration,
prev, current);\n next = prev + current;\n prev = current;\n current = next;\n iteration++;\n printf(\"Iteration %d end: current=%d\\n\", iteration-1, current);\n}\n",
 "answer": 8
 }
},
{
 "id": "PL-LP-S002-V006",
 "metadata": {
 "name": "Running Average Calculation Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {

```

```
 "description": "In this running average calculation, what is the value of 'running_avg' at the start of the 4th iteration?",
 "code": "int data[] = {10, 25, 15, 30, 20, 35};\n double running_sum = 0.0;\n double running_avg = 0.0;\n for (int i = 0; i < 6; i++) {\n printf(\"Iteration %d start: running_avg = %.2f\\n\", i+1, running_avg);\n running_sum += data[i];\n running_avg = running_sum / (i + 1);\n printf(\"Added %d, new avg = %.2f\\n\", data[i], running_avg);\n }",
 "answer": 16.67
 },
 {
 "id": "PL-LP-S002-V007",
 "metadata": {
 "name": "Factorial Accumulator Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the factorial calculation and determine the value of 'factorial' after processing the 5th iteration.",
 "code": "long long factorial = 1;\n for (int i = 1; i <= 7; i++) {\n printf(\"Iteration %d start: factorial = %lld\\n\", i, factorial);\n factorial *= i;\n }\n printf(\"Iteration %d end: factorial = %lld\\n\", i, factorial);",
 "answer": 120
 },
 },
 {
 "id": "PL-LP-S002-V008",
 "metadata": {
 "name": "Exponential Decay Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this exponential decay simulation, what is the value of 'energy' at the start of the 6th iteration?",
 }
 }
}
```

```
 "code": "double energy = 1000.0;\ndouble decay_rate = 0.85;\nint time_step = 0;\n\nwhile (energy > 10.0 && time_step < 20) {\n time_step++;\n printf(\"Step %d start: energy = %.2f\\n\", time_step, energy);\n energy *= decay_rate;\n decay_rate -= 0.02; // Increasing decay\n printf(\"Step %d end: energy = %.2f, rate = %.3f\\n\", time_step, energy, decay_rate);\n}",
 "answer": 418.21
 }
},
{
 "id": "PL-LP-S002-V009",
 "metadata": {
 "name": "Circular Buffer Index Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "数组索引追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this circular buffer implementation, what is the value of 'write_index' at the start of the 8th iteration?",
 "code": "int buffer[5] = {0};\nint write_index = 0;\nint read_index = 0;\n\nint buffer_size = 5;\nint data[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};\n\nfor (int i = 0; i < 10; i++) {\n printf(\"Iteration %d start: write_index = %d\\n\", i+1, write_index);\n buffer[write_index] = data[i];\n write_index = (write_index + 1) % buffer_size;\n\n if (i >= 2) { // Start reading after 3 writes\n read_index = (read_index + 1) % buffer_size;\n }\n printf(\"wrote %d, write_index now = %d\\n\", data[i], write_index);\n}",
 "answer": 2
 }
},
{
 "id": "PL-LP-S002-V010",
 "metadata": {
 "name": "Digital Root Calculation Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
```

```
 "description": "Track the digital root calculation and determine the value of 'number' at the start of the 3rd iteration.",
 "code": "int number = 9875;\nint iteration = 0;\n\nwhile (number >= 10) {\n iteration++;\n printf(\"Iteration %d start: number = %d\\n\", iteration, number);\n int sum = 0;\n int temp = number;\n while (temp > 0) {\n sum += temp % 10;\n temp /= 10;\n }\n number = sum;\n printf(\"Iteration %d end: number = %d\\n\", iteration, number);\n}\n\nprintf(\"Digital root: %d\\n\", number);",
 "answer": 11
 },
},
{
 "id": "PL-LP-S003-V001",
 "metadata": {
 "name": "Simple Convergence Termination",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "单一条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this simple numerical approximation loop and determine after which iteration number the loop will terminate.",
 "code": "double x = 10.0;\ndouble target = 5.0;\ndouble tolerance = 0.1;\n\nint iteration = 0;\n\nwhile (fabs(x - target) > tolerance) {\n iteration++;\n x = (x + target) / 2.0; // simple averaging approach\n printf(\"Iteration %d: x = %.3f, diff = %.3f\\n\", iteration, x, fabs(x - target));\n}\n\nprintf(\"Converged after %d iterations\\n\", iteration);",
 "answer": 5
 },
},
{
 "id": "PL-LP-S003-V002",
 "metadata": {
 "name": "Compound Termination Condition Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "复合条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
```

```

 "description": "This loop has multiple termination conditions. Analyze which condition will trigger first and determine the iteration number when the loop terminates.",
 "code": "double value = 2.0;\ndouble growth_rate = 1.15;\nint iteration = 0;\nint max_iterations = 20;\ndouble upper_limit = 50.0;\ndouble stability_threshold = 0.01;\ndouble prev_value = 0.0;\ndo {\n prev_value = value;\n value *= growth_rate;\n growth_rate *= 0.98; // Decreasing growth rate\n iteration++;\\n\n printf(\"Iteration %d: value=%f, growth_rate=%f, change=%f\\n\", \niteration, value, growth_rate, fabs(value - prev_value));\\n\n} while (iteration < max_iterations && \n value < upper_limit && \n fabs(value - prev_value) > stability_threshold);\\n\\nprintf(\"Terminated after %d iterations\\n\", iteration);",
 "answer": 15
}
},
{
 "id": "PL-LP-S003-V003",
 "metadata": {
 "name": "Dynamic Termination Condition Change",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "动态条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this loop with dynamic termination conditions, the termination condition itself changes during loop execution. Determine after which iteration the loop will terminate.",
 "code": "int sum = 0;\nint threshold = 30;\nint increment = 5;\nint adjustment_frequency = 3;\nint iteration = 0;\n\nwhile (sum < threshold) {\n iteration++;\n sum += increment;\n // Adjust conditions every 3 iterations\n if (iteration % adjustment_frequency == 0) {\n threshold += 10; // Increase threshold\n increment += 2; // Increase increment\n }\n\n printf(\"Iteration %d: sum=%d, threshold=%d, increment=%d\\n\", \niteration, sum, threshold, increment);\n}\n\nprintf(\"Loop terminated after %d iterations\\n\", iteration);",
 "answer": 7
 }
},
{
 "id": "PL-LP-S003-V004",
 "metadata": {
 "name": "Binary Search Termination Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "二分查找"
 }
}

```

```

 "intervention": 0,
 "variant_type": "收敛条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this binary search loop and determine after which iteration it will terminate when searching for target value 7.",
 "code": "int binary_search(int arr[], int size, int target) {\n int left = 0,\n right = size - 1;\n int iteration = 0;\n while (left <= right) {\n iteration++;\n int mid = left + (right - left) / 2;\n printf(\"Iteration %d:\nleft=%d, right=%d, mid=%d, arr[%d]=%d\\n\", iteration, left, right, mid,\narr[mid]);\n if (arr[mid] == target) {\n printf(\"Found target\nafter %d iterations\\n\", iteration);\n return mid;\n }\n if (arr[mid] < target) {\n left = mid + 1;\n } else {\n right = mid - 1;\n }\n }\n printf(\"Target not found after %d iterations\\n\", iteration);\n return -1;\n}\n// Array: [1, 3, 5, 7, 9, 11, 13, 15]\nint arr[] = {1, 3,\n5, 7, 9, 11, 13, 15};\nint result = binary_search(arr, 8, 7);",
 "answer": 2
 }
},
{
 "id": "PL-LP-S003-V005",
 "metadata": {
 "name": "Square Root Approximation Termination",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "收敛条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This square root approximation using Newton's method will terminate after how many iterations?",
 "code": "double sqrt_newton(double n) {\n double x = n / 2.0; // Initial\n guess\n double tolerance = 0.001;\n int iteration = 0;\n while (1) {\n iteration++;\n double new_x = 0.5 * (x + n / x);\n printf(\"Iteration %d: x =\n%.6f, new_x = %.6f, diff = %.6f\\n\", iteration, x, new_x, fabs(new_x -\nx));\n if (fabs(new_x - x) < tolerance) {\n printf(\"Converged\nafter %d iterations\\n\", iteration);\n return new_x;\n }\n x = new_x;\n }\n}\n// calculate sqrt(25)\ndouble result = sqrt_newton(25.0);",
 "answer": 4
 }
},
{
 "id": "PL-LP-S003-V006",
 "metadata": {
 "name": "Collatz Sequence Termination",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "收敛条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 }
}

```

```

 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "单一条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Calculate how many iterations the Collatz sequence takes to reach 1 when starting with n=27.",
 "code": "int collatz_steps(int n) {\n int steps = 0;\n int original = n;\n\n while (n != 1) {\n steps++; \n\n if (n % 2 == 0) {\n n = n / 2;\n printf(\"Step %d: n = %d\\n\", steps,\n\n n);\n\n if (n % 2 == 0) {\n n = n / 2;\n printf(\" -> %d (even, divide by 2)\\n\", n);\n } else {\n n = 3 * n + 1;\n printf(\" -> %d (odd, 3n+1)\\n\", n);\n }\n }\n\n printf(\"Collatz(%d) takes %d steps\\n\", original, steps);\n }\n\n return steps;\n}\n\n// calculate Collatz(27)\nint result = collatz_steps(27);",
 "answer": 111
 }
},
{
 "id": "PL-LP-S003-V007",
 "metadata": {
 "name": "Power Series Convergence",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "收敛条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This power series approximation of e^x will terminate after how many iterations when x=1 and tolerance=0.0001?",
 "code": "double power_series_exp(double x, double tolerance) {\n double sum = 1.0; // First term\n double term = 1.0;\n int iteration = 0;\n\n while (1) {\n iteration++;\n term = term * x / iteration; // Next term: x^n/n!\n sum += term;\n\n printf(\"Iteration %d: term = %.6f, sum = %.6f\\n\", iteration,\n\n term, sum);\n\n if (fabs(term) < tolerance) {\n printf(\"Converged after %d iterations\\n\", iteration);\n return sum;\n }\n\n if (iteration > 50) { // Safety limit\n printf(\"Reached iteration limit\\n\");\n break;\n }\n }\n\n return sum;\n}\n\n// calculate e^1 with tolerance 0.0001\ndouble result = power_series_exp(1.0, 0.0001);",
 "answer": 7
 }
}

```

```
},
{
 "id": "PL-LP-S003-v008",
 "metadata": {
 "name": "Timeout Based Termination",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "复合条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This loop with timeout and value conditions will terminate after which iteration?",
 "code": "int process_with_timeout() {\n int value = 100;\n int timeout_counter = 0;\n int max_timeout = 15;\n int iteration = 0;\n while (value > 10 && timeout_counter < max_timeout) {\n iteration++;\n value -= (iteration * 2);\n timeout_counter += iteration;\n printf(\"Iteration %d: value=%d, timeout_counter=%d\\n\", iteration, value, timeout_counter);\n }\n printf(\"Terminated after %d iterations\\n\", iteration);\n if (value <= 10) {\n printf(\"Terminated due to value condition\\n\");\n } else {\n printf(\"Terminated due to timeout\\n\");\n }\n return iteration;\n}\nint result = process_with_timeout();",
 "answer": 4
 }
},
{
 "id": "PL-LP-S003-v009",
 "metadata": {
 "name": "Oscillating System Termination",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "动态条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This oscillating system will reach equilibrium and terminate after which iteration?"
 }
}
```

```

 "code": "double oscillating_system()\n double position = 10.0;\n double\n velocity = 0.0;\n double damping = 0.9;\n double spring_constant = 0.8;\n double\n equilibrium_threshold = 0.01;\n int iteration = 0;\n \n while (fabs(position) >\nequilibrium_threshold || fabs(velocity) > equilibrium_threshold) {\n iteration++;\n\n // simple harmonic oscillator with damping\n double acceleration = -\n spring_constant * position;\n velocity = velocity * damping + acceleration;\n\n position += velocity;\n\n printf(\"Iteration %d: pos=%.\n4f, vel=%.\n4f,\nacc=%.\n4f\\n\", iteration, position, velocity, acceleration);\n\n if (iteration > 100) { // safety limit\n printf(\"Reached iteration\nlimit\\n\");\n break;\n }\n }\n\n printf(\"System reached\nequilibrium after %d iterations\\n\", iteration);\n\n return iteration;\n}\n\nint result =\noscillating_system();",
 "answer": 67
 }
},
{
 "id": "PL-LP-S004-V001",
 "metadata": {
 "name": "Basic Conditional Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "简单计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this simple conditional processing loop, how many times is the variable 'even_count' incremented?",
 "code": "numbers = [2, 7, 4, 9, 6, 13, 8, 15, 10, 17]\n\neven_count = 0\nnodd_count = 0\n\nfor num in numbers:\n if num % 2 == 0:\n even_count += 1\n\n print(f\"Found even number: {num}\")\n\n else:\n odd_count += 1\n\n print(f\"Found odd number: {num}\")\n\n nprint(f\"Even count: {even_count}, Odd count: {odd_count}\")",
 "answer": 5
 }
},
{
 "id": "PL-LP-S004-V002",
 "metadata": {
 "name": "Nested Conditional Counting Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "条件计数变式",
 }
}

```



```

 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this string processing loop, how many words match the pattern criteria and increment the pattern_count?",

 "code": "words = ['apple', 'banana', 'cherry', 'date', 'elderberry', 'fig',

'grape']\npattern_count = 0\nvowel_start_count = 0\n\nfor word in words:\n if len(word)

>= 5: # Length condition\n if word[0].lower() in 'aeiou': # Starts with vowel\n pattern_count += 1\n print(f\"Pattern match: {word} (length={len(word)}),

starts with vowel\")\n else:\n print(f\"Length OK but no vowel start:

{word}\")\n elif word[0].lower() in 'aeiou':\n vowel_start_count += 1\n print(f\"Vowel start but short: {word}\")\n\n\nprint(f\"Pattern count:

{pattern_count}, Vowel start count: {vowel_start_count}\")",
 "answer": 2
 }
},
{
 "id": "PL-LP-S004-V005",
 "metadata": {
 "name": "Temperature Range Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "简单计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this temperature data processing loop and determine how many readings fall in the 'hot' category.",

 "code": "temperatures = [18, 25, 32, 28, 35, 22, 38, 29, 41, 26]\nhot_count =

0\nwarm_count = 0\ncool_count = 0\n\nfor temp in temperatures:\n if temp >= 35:\n hot_count += 1\n print(f\"Hot day: {temp}°C\")\n elif temp >= 25:\n warm_count += 1\n print(f\"Warm day: {temp}°C\")\n else:\n cool_count += 1\n print(f\"Cool day: {temp}°C\")\n\n\nprint(f\"Hot: {hot_count}, Warm:

{warm_count}, Cool: {cool_count}\")",
 "answer": 3
 }
},
{
 "id": "PL-LP-S004-V006",
 "metadata": {
 "name": "Grade Classification Counting",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "简单计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 }
}

```

```

 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this grade classification system, how many students receive an 'A' grade (score >= 90)?",
 "code": "scores = [95, 78, 82, 91, 76, 88, 93, 84, 97, 79, 85, 92]\na_grade_count = 0\nb_grade_count = 0\nc_grade_count = 0\nother_count = 0\n\nfor score in scores:\n if score >= 90:\n a_grade_count += 1\n print(f\"A grade: {score}\")\n elif score >= 80:\n b_grade_count += 1\n print(f\"B grade: {score}\")\n elif score >= 70:\n c_grade_count += 1\n print(f\"C grade: {score}\")\n else:\n other_count += 1\n print(f\"Below C: {score}\")\n\nprint(f\"A: {a_grade_count}, B: {b_grade_count}, C: {c_grade_count}, Other: {other_count}\")",
 "answer": 5
 }
},
{
 "id": "PL-LP-S004-V007",
 "metadata": {
 "name": "List Comprehension Alternative Count",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this complex filtering loop, how many items satisfy all the conditions and get added to the result_count?",
 "code": "data = [{value: 15, category: 'A'}, {value: 8, category: 'B'},\n {value: 22, category: 'A'}, {value: 12, category: 'C'},\n {value: 30, category: 'B'}, {value: 5, category: 'A'},\n {value: 18, category: 'C'}, {value: 25, category: 'B'}]\n\nresult_count = 0\n\nfor item in data:\n total_processed += 1\n if item['value'] > 10: # First condition\n if item['category'] in ['A', 'B']: # Second condition\n if item['value'] % 2 == 0: # Third condition: even numbers\n result_count += 1\n\nprint(f\"Match: {item}\")\n else:\n print(f\"Odd value: {item}\")\n else:\n print(f\"Wrong category: {item}\")\n\nprint(f\"Value too small: {item}\")\n \nprint(f\"Result count: {result_count}, Total processed: {total_processed}\")",
 "answer": 3
 }
}

```

```
 },
 },
 {
 "id": "PL-LP-S004-V008",
 "metadata": {
 "name": "sliding window Pattern Count",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "In this sliding window analysis, how many windows of size 3 have an average greater than 15?",
 "code": "numbers = [12, 18, 14, 20, 16, 22, 11, 19, 25, 13, 17]\nwindow_size = 3\nhigh_avg_count = 0\nntotal_windows = 0\nfor i in range(len(numbers) - window_size + 1):\n total_windows += 1\n window = numbers[i:i + window_size]\n avg = sum(window) / len(window)\n print(f\"window {i+1}: {window} -> avg = {avg:.2f}\")\n if avg > 15:\n high_avg_count += 1\n print(f\" High average window!\")\n\nprint(f\"High average windows: {high_avg_count} out of {total_windows}\")",
 "answer": 7
 }
 },
 {
 "id": "PL-LP-S004-V009",
 "metadata": {
 "name": "Palindrome Detection Count",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "条件计数变式",
 "property_focus": "iteration_count",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Count how many strings in this list are palindromes and increment the palindrome_count."
 }
 }
}
```

```
 "code": "def is_palindrome(s):\n cleaned = s.lower().replace(' ', '')\n\nreturn cleaned == cleaned[::-1]\n\nstrings = ['level', 'hello', 'radar', 'world', 'madam',\n'python', 'racecar', 'test']\n\npalindrome_count = 0\nnon_palindrome_count = 0\n\nfor string\nin strings:\n if is_palindrome(string):\n palindrome_count += 1\n print(f\"Palindrome found: '{string}'\")\n else:\n non_palindrome_count += 1\n\n print(f\"Not palindrome: '{string}'\")\n\nprint(f\"Palindromes: {palindrome_count}, Non-palindromes: {non_palindrome_count}\")",
 "answer": 4
 }
},
{
 "id": "PL-LP-S005-V001",
 "metadata": {
 "name": "Simple Hash Accumulation Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the accumulator value in this simplified hash function and determine the value of 'hash' after processing the 2nd character.",
 "code": "unsigned int simple_hash(const char *str) {\n unsigned int hash = 1;\n int pos = 0;\n char c;\n while ((c = str[pos]) != '\\0') {\n hash = hash * 31 + (unsigned int)c;\n pos++;\n }\n return hash;\n}\n// called with\nsimple_hash(\"abc\")\nunsigned int result = simple_hash(\"abc\");",
 "answer": 3039
 }
},
{
 "id": "PL-LP-S005-V002",
 "metadata": {
 "name": "Complex Accumulation Pattern Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
```

```
 "description": "In this complex accumulation calculation, what is the value of 'accumulator' after the 5th iteration?",
 "code": "long long complex_accumulator(int n) {\n long long accumulator =
1;\\n long long factor = 2;\\n \\n for (int i = 1; i <= n; i++) {\\n accumulator =
 accumulator * factor + i * i;\\n factor += i;\\n printf(\"Iteration %d:
 accumulator = %lld, factor = %lld\\n\", i, accumulator, factor);\\n }\\n \\n return
 accumulator;\\n}\\n\\n// Called with complex_accumulator(7)\\nlong long result =
complex_accumulator(7);",
 "answer": 7381
 }
},
{
 "id": "PL-LP-S005-V003",
 "metadata": {
 "name": "Polynomial Hash Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track this polynomial rolling hash calculation and determine the hash value after processing the 3rd character.",
 "code": "unsigned long polynomial_hash(const char *str, int prime, int mod) {\n unsigned long hash = 0;\\n unsigned long prime_power = 1;\\n int pos = 0;\\n char c;\\n \\n while ((c = str[pos]) != '\\0') {\\n hash = (hash + ((unsigned long)c *
 prime_power)) % mod;\\n prime_power = (prime_power * prime) % mod;\\n pos++;\\n printf(\"After char '%c': hash = %lu, prime_power = %lu\\n\", c, hash, prime_power);\\n }\\n \\n return hash;\\n}\\n\\n// Called with polynomial_hash(\"test\", 31,
1000000007)\\nunsigned long result = polynomial_hash(\"test\", 31, 1000000007);",
 "answer": 3027668
 }
},
{
 "id": "PL-LP-S005-V004",
 "metadata": {
 "name": "Checksum Calculation Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
```

```

 },
 "task": {
 "description": "In this checksum calculation loop, what is the value of 'checksum' after processing the 4th byte?",
 "code": "unsigned char calculate_checksum(unsigned char *data, int length) {\n unsigned char checksum = 0;\n unsigned int temp_sum = 0;\n for (int i = 0; i < length; i++) {\n temp_sum += data[i];\n checksum = temp_sum & 0xFF; // Keep only lower 8 bits\n temp_sum >>= 8; // Carry to next iteration\n }\n printf(\"Byte %d: data=%d, temp_sum=%d, checksum=%d\\n\", i+1, data[i], temp_sum, checksum);\n return checksum;\n}\n// Data: [0x42, 0x1F, 0x8A, 0x37, 0xC5, 0x63]\nunsigned char data[] = {0x42, 0x1F, 0x8A, 0x37, 0xC5, 0x63};\nunsigned char result = calculate_checksum(data, 6);",
 "answer": 34
 }
},
{
 "id": "PL-LP-S005-V005",
 "metadata": {
 "name": "CRC Computation Tracking",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the CRC calculation and determine the value of 'crc' after processing the 3rd byte.",
 "code": "unsigned int crc8_simple(unsigned char *data, int length) {\n unsigned int crc = 0;\n unsigned int polynomial = 0x1D; // x^8 + x^4 + x^3 + x^2 + 1\n for (int i = 0; i < length; i++) {\n crc ^= data[i];\n printf(\"After XOR with byte %d (0x%02X): crc = 0x%02X\\n\", i+1, data[i], crc);\n for (int bit = 0; bit < 8; bit++) {\n if (crc & 0x80) {\n crc = (crc << 1) ^ polynomial;\n } else {\n crc <= 1;\n }\n crc &= 0xFF;\n }\n printf(\"After bit processing: crc = 0x%02X\\n\", crc);\n }\n return crc;\n}\n// Data: [0xAB, 0xCD, 0xEF, 0x12]\nunsigned char data[] = {0xAB, 0xCD, 0xEF, 0x12};\nunsigned int result = crc8_simple(data, 4);",
 "answer": 246
 }
},
{
 "id": "PL-LP-S005-V006",
 "metadata": {
 "name": "XOR Chain Accumulation",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 }
}

```

```

"language": "c",
"difficulty": "medium",
"intervention": 0,
"variant_type": "累积器追踪变式",
"property_focus": "variable_tracking",
"analysis_type": "quantitative"
},
"task": {
 "description": "In this XOR chain calculation, what is the value of 'xor_result' after the 5th iteration?",
 "code": "unsigned int xor_chain(unsigned int *values, int count) {\n unsigned int xor_result = 0;\n unsigned int shift_amount = 1;\n for (int i = 0; i < count; i++) {\n xor_result ^= (values[i] << shift_amount);\n shift_amount = (shift_amount + i) % 8;\n printf(\"Iteration %d: value=0x%x, shifted=0x%x,\n xor_result=0x%x\\n\", i+1, values[i], values[i] << shift_amount,\n xor_result);\n }\n return xor_result;\n}\n// values: [0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xDE]\nunsigned int values[] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xDE};\nunsigned int result = xor_chain(values, 7);",
 "answer": 0x2AC
},
{
 "id": "PL-LP-S005-V007",
 "metadata": {
 "name": "Weighted Sum Accumulation",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Track the weighted sum calculation and determine the value of 'weighted_sum' after the 4th iteration.",
 "code": "double weighted_sum_calc(double *values, double *weights, int count)\n{\n double weighted_sum = 0.0;\n double weight_factor = 1.0;\n for (int i = 0; i < count; i++) {\n double contribution = values[i] * weights[i] *\n weight_factor;\n weighted_sum += contribution;\n weight_factor *= 0.9; // Decay factor\n printf(\"Iteration %d: value=%.2f, weight=%.2f, factor=%.3f,\n contribution=%.3f, sum=%.3f\\n\", i+1, values[i], weights[i],\n weight_factor, contribution, weighted_sum);\n }\n return weighted_sum;\n}\n// values: [10.0, 15.0, 20.0, 25.0, 30.0]\n// weights: [0.5, 0.7, 0.3, 0.8, 0.6]\ndouble values[] = {10.0, 15.0, 20.0, 25.0, 30.0};\ndouble weights[] = {0.5, 0.7, 0.3, 0.8, 0.6};\ndouble result = weighted_sum_calc(values, weights, 5);",
 "answer": 25.37
 }
}

```

```
"id": "PL-LP-S005-V008",
"metadata": {
 "name": "Base64 Encoding Accumulation",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "累积器追踪变式",
 "property_focus": "variable_tracking",
 "analysis_type": "quantitative"
},
"task": {
 "description": "In this simplified base64 encoding loop, what is the value of 'accumulator' after processing the 2nd input byte?",
 "code": "unsigned int base64_encode_step(unsigned char *input, int length) {\n unsigned int accumulator = 0;\n int bit_count = 0;\n for (int i = 0; i < length; i++) {\n accumulator = (accumulator << 8) | input[i];\n bit_count += 8;\n printf(\"After byte %d (0x%02X): accumulator = 0x%06X, bits = %d\\n\", \n i+1, input[i], accumulator, bit_count);\n while (bit_count >= 6) {\n unsigned int index = (accumulator >> (bit_count - 6)) & 0x3F;\n printf(\"Extracted 6-bit index: %d\\n\", index);\n bit_count -= 6;\n accumulator &= (1 << bit_count) - 1;\n }\n }\n return accumulator;\n}\n// Input: [0x4D, 0x61, 0x6E] // \"Man\" in ASCII\nunsigned char input[] = {0x4D, 0x61, 0x6E};\nunsigned int result = base64_encode_step(input, 3);",
 "answer": "0x0616E"
},
{
 "id": "PL-LP-S006-V001",
 "metadata": {
 "name": "Resource Threshold Exit Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "多出口条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
},
 "task": {
 "description": "This loop has multiple exit conditions based on resource consumption. Which specific condition will cause termination and on which iteration?",
```

```

 "code": "int resource_manager(int initial_memory) {\n int memory_used = 0;\n int cpu_cycles = 0;\n int iterations = 0;\n int memory_increment = 15;\n\n while (iterations < 10) {\n iterations++;\n cpu_cycles += iterations * 2;\n\n printf(\"Iteration %d: memory=%d,\n cpu=%d\\n\", iterations, memory_used, cpu_cycles);\n\n if (memory_used > initial_memory) {\n memory_increment += 5;\n printf(\"Memory limit exceeded at iteration %d\\n\", iterations);\n return 1;\n }\n\n // Condition 2: CPU cycles limit\n if (cpu_cycles > 50) {\n printf(\"CPU limit exceeded at iteration %d\\n\", iterations);\n return 2;\n }\n\n memory_increment += 5;\n }\n\n printf(\"Completed all iterations\\n\");\n return 0;\n}\n\n// Called with initial_memory = 80\nint result = resource_manager(80);",
 "answer": 4
}
},
{
 "id": "PL-LP-S006-V002",
 "metadata": {
 "name": "Data Structure Invariant Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "数据结构不变量变式",
 "property_focus": "invariant",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "Analyze this loop that maintains a data structure invariant. Which iteration will violate the invariant condition and cause early termination?",
 "code": "int maintain_balance(int arr[], int size) {\n int left_sum = 0,\n right_sum = 0;\n int pivot = size / 2;\n int iterations = 0;\n\n for (int i = 0; i < size; i++) {\n iterations++;\n\n if (i < pivot) {\n left_sum += arr[i];\n } else if (i > pivot) {\n right_sum += arr[i];\n }\n\n printf(\"Iteration %d: left_sum=%d, right_sum=%d, diff=%d\\n\", \n iterations, left_sum, right_sum, abs(left_sum - right_sum));\n\n // Invariant: difference should not exceed 20\n if (abs(left_sum - right_sum) > 20 && i > pivot) {\n printf(\"Invariant violated at iteration %d\\n\", iterations);\n return iterations;\n }\n\n printf(\"Invariant maintained through all iterations\\n\");\n }\n\n return iterations;\n}\n\n// Array: [5, 8, 12, 3, 15, 25, 7]\nint test_arr[] = {5, 8, 12, 3, 15, 25, 7};\nint result = maintain_balance(test_arr, 7);",
 "answer": 6
 }
},
{
 "id": "PL-LP-S006-V003",
 "metadata": {
 "name": "Priority Queue Exit Condition",
 "category": "Property-Level",

```

```

 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "异常终止变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This priority-based processing loop has multiple exit conditions. Determine which condition triggers termination and on which iteration.",
 "code": "int priority_processor() {\n int high_priority_count = 0;\n int total_processed = 0;\n int priority_queue[] = {1, 3, 2, 1, 3, 3, 2, 1, 3, 2};\n int queue_size = 10;\n int iterations = 0;\n for (int i = 0; i < queue_size; i++) {\n iterations++;\n total_processed++;\n if (priority_queue[i] == 3) {\n high_priority_count++;\n }\n printf(\"Iteration %d: priority=%d, high_count=%d, total=%d\\n\", iterations, priority_queue[i], high_priority_count, total_processed);\n }\n // Condition 1: Too many high priority items\n if (high_priority_count > 4) {\n printf(\"High priority overflow at iteration %d\\n\", iterations);\n return 1;\n }\n // Condition 2: Processing capacity exceeded\n if (total_processed > 7) {\n printf(\"Capacity exceeded at iteration %d\\n\", iterations);\n return 2;\n }\n printf(\"All items processed successfully\\n\");\n return 0;\n}\nint result = priority_processor();",
 "answer": 8
 }
},
{
 "id": "PL-LP-S006-V004",
 "metadata": {
 "name": "Stack Overflow Detection",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "异常终止变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This stack simulation will trigger an overflow condition. On which iteration will the overflow be detected?",
 }
}

```

```

 "code": "int stack_simulation() {\n int stack_size = 0;\n int\nmax_stack_size = 8;\n int operations[] = {1, 1, -1, 1, 1, 1, -1, 1, 1, 1, 1, -1}; //\n1=push, -1=pop\n int op_count = 12;\n int iterations = 0;\n for (int i = 0; i < op_count; i++) {\n iterations++; \n if (operations[i] == 1) { //\nPush operation\n stack_size++; \n printf(\"Iteration %d: PUSH,\nstack_size = %d\\n\", iterations, stack_size);\n if (stack_size > max_stack_size) {\n printf(\"Stack overflow detected at iteration %d\\n\\n\", iterations);\n return iterations;\n } else { // Pop\n if (stack_size > 0) {\n stack_size--;\n printf(\"Iteration %d: POP, stack_size = %d\\n\", iterations, stack_size);\n }\n }\n } else {\n printf(\"Stack underflow detected at iteration %d\\n\", iterations);\n return iterations;\n }\n }\n printf(\"All operations completed successfully\\n\");\n return 0;\n}\n\nint result = stack_simulation();\n\n \"answer\": 9\n }\n},\n{\n \"id\": \"PL-LP-S006-V005\",\n \"metadata\": {\n \"name\": \"Memory Pool Exhaustion\",\n \"category\": \"Property-Level\",\n \"subcategory\": \"Loop\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"hard\",\n \"intervention\": 0,\n \"variant_type\": \"多出口条件变式\",\n \"property_focus\": \"termination_condition\",\n \"analysis_type\": \"quantitative\"\n },\n \"task\": {\n \"description\": \"This memory allocation simulator has multiple failure\nconditions. Which iteration will trigger the first failure condition?\",\n \"code\": \"int memory_allocator() {\n int total_memory = 100;\n int\nallocated_blocks = 0;\n int max_blocks = 10;\n int allocation_sizes[] = {8, 12, 15,\n10, 20, 18, 7, 25, 14, 16, 22, 9};\n int requests = 12;\n int iterations = 0;\n for (int i = 0; i < requests; i++) {\n iterations++;\n int size =\nallocation_sizes[i];\n printf(\"Iteration %d: Requesting %d bytes,\navailable=%d, blocks=%d\\n\", iterations, size, total_memory,\nallocated_blocks);\n // Condition 1: Not enough memory\n if (size >\ntotal_memory) {\n printf(\"out of memory at iteration %d\\n\", iterations);\n return 1;\n }\n // Condition 2: Too many blocks\n if (allocated_blocks >= max_blocks) {\n printf(\"Too many blocks at iteration %d\\n\", iterations);\n return 2;\n }\n // Allocate\n total_memory -= size;\n allocated_blocks++;\n printf(\"Allocated successfully, remaining=%d\\n\", total_memory);\n }\n printf(\"All\nallocations successful\\n\");\n return 0;\n}\n\nint result = memory_allocator();\n\n \"answer\": 6\n }\n},\n{\n
```

```

"id": "PL-LP-S006-V006",
"metadata": {
 "name": "Network Packet Drop Analysis",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "多出口条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
},
"task": {
 "description": "This network packet processing simulation will drop packets under certain conditions. On which iteration will the first packet drop occur?",
 "code": "int packet_processor() {\n int buffer_size = 0;\n int max_buffer = 6;\n int consecutive_drops = 0;\n int max_consecutive = 3;\n int packet_sizes[] = {2, 1, 3, 2, 4, 1, 2, 3, 5, 1};\n int packet_count = 10;\n int iterations = 0;\n\n for (int i = 0; i < packet_count; i++) {\n iterations++;\n int size = packet_sizes[i];\n\n printf(\"Iteration %d: Packet size=%d, buffer=%d\\n\", iterations, size, buffer_size);\n\n if (buffer_size + size > max_buffer) {\n consecutive_drops++;\n printf(\" Packet dropped! Consecutive drops: %d\\n\", consecutive_drops);\n\n if (consecutive_drops > max_consecutive) {\n printf(\"Too many consecutive drops at iteration %d\\n\", iterations);\n return iterations;\n } else {\n buffer_size += size;\n consecutive_drops = 0;\n }\n }\n\n // Simulate buffer drain\n buffer_size = (buffer_size > 1) ? buffer_size - 1 : 0;\n }\n\n printf(\"All packets processed\\n\");\n return 0;\n}\n\nint result = packet_processor();",
 "answer": 9
},
{
 "id": "PL-LP-S006-V007",
 "metadata": {
 "name": "Database Transaction Rollback",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "异常终止变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
},
 "task": {
 "description": "This database transaction simulator will trigger a rollback under certain conditions. On which iteration will the rollback be triggered?",
```

```
 "code": "int transaction_processor() {\n int active_transactions = 0;\n int max_transactions = 5;\n int failed_operations = 0;\n int max_failures = 2;\n int operations[] = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0}; // 1=success, 0=failure\n int\n op_count = 10;\n int iterations = 0;\n for (int i = 0; i < op_count; i++) {\n iterations++;\n active_transactions++;\n printf(\"Iteration %d:\nStarting transaction %d\\n\", iterations, active_transactions);\n if\n (operations[i] == 0) { // operation failed\n failed_operations++;\n printf(\" Operation failed! Total failures: %d\\n\", failed_operations);\n if (failed_operations > max_failures) {\n printf(\"Too many failures - rollback at iteration %d\\n\", iterations);\n return iterations;\n }\n } else {\n printf(\" Operation succeeded\\n\");\n }\n if (active_transactions > max_transactions) {\n printf(\"Too many\nactive transactions at iteration %d\\n\", iterations);\n return iterations;\n }\n // Complete some transactions\n if (i % 3 == 2) {\n active_transactions = (active_transactions > 2) ? active_transactions - 2 : 0;\n }\n }\n printf(\"All transactions completed successfully\\n\");\n return\n0;\n}\n\nint result = transaction_processor();",
 "answer": 5
 },
 {
 "id": "PL-LP-S006-V008",
 "metadata": {
 "name": "Load Balancer Circuit Breaker",
 "category": "Property-Level",
 "subcategory": "Loop",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "复合条件变式",
 "property_focus": "termination_condition",
 "analysis_type": "quantitative"
 },
 "task": {
 "description": "This load balancer simulation implements a circuit breaker pattern. On which iteration will the circuit breaker be triggered?"
 }
 }
]
```

```

 "code": "int load_balancer() {\n int successful_requests = 0;\n int failed_requests = 0;\n double failure_rate_threshold = 0.4;\n int min_requests_for_check = 5;\n int request_results[] = {1, 1, 0, 1, 0, 0, 1, 0, 0, 1}; //\n 1=success, 0=failure\n int total_requests = 10;\n int iterations = 0;\n for (int i = 0; i < total_requests; i++) {\n iterations++;\n if (request_results[i] == 1) {\n successful_requests++;\n printf(\"Iteration %d: SUCCESS - Total: %d success, %d failures\\n\", \niterations, successful_requests, failed_requests);\n } else {\n failed_requests++;\n printf(\"Iteration %d: FAILURE - Total: %d success, %d failures\\n\", \niterations, successful_requests, failed_requests);\n }\n }\n int total_processed = successful_requests + failed_requests;\n if (total_processed >= min_requests_for_check) {\n double failure_rate = (double)failed_requests / total_processed;\n printf(\" Failure rate: %.2f\\n\", failure_rate);\n if (failure_rate > failure_rate_threshold) {\n printf(\"circuit breaker triggered at iteration %d\\n\", iterations);\n return iterations;\n }\n }\n printf(\"All requests processed successfully\\n\");\n return 0;\n}\nint result = load_balancer();",
 "answer": 6
}
]

```

## 3B - 分支属性 [ Branch ] (62)

### ## 任务目标

基于给定的分支属性推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对分支结构深层属性的理解能力，包括条件求值、路径选择和分支效果等核心分支属性推理技能。

### ## 分支属性推理特征分析

分支属性推理关注条件分支结构的内在机制和决策过程，重点测试模型对条件表达式计算、执行路径选择和分支结果影响的深层理解，强调对分支逻辑的精确分析能力。

### ## 关键要求

- \*\*描述语言\*\*：所有task描述必须使用英文
- \*\*变式数量\*\*：每个种子任务严格生成8-12个变式，确保数量充足
- \*\*属性聚焦\*\*：重点关注分支的属性分析，而非简单的执行结果

### ## 变式生成维度

#### ### 1. 条件求值分析变式

- \*\*布尔表达式求值变式\*\*：复杂布尔表达式的逐步求值分析
- \*\*比较运算求值变式\*\*：多重比较运算的求值顺序和结果
- \*\*算术条件求值变式\*\*：包含算术运算的条件表达式求值
- \*\*函数调用条件变式\*\*：条件中包含函数调用的求值分析
- \*\*类型转换条件变式\*\*：涉及隐式类型转换的条件求值
- \*\*位运算条件变式\*\*：位运算结果的条件判断分析

#### ### 2. 短路求值分析变式

- \*\*逻辑AND短路变式\*\*：`&&`运算符的短路行为分析
- \*\*逻辑OR短路变式\*\*：`||`运算符的短路行为分析
- \*\*函数副作用短路变式\*\*：短路对函数副作用的影响分析
- \*\*嵌套短路变式\*\*：多层嵌套的短路求值分析

- \*\*条件链短路变式\*\*：条件链中的短路行为
- \*\*复合短路变式\*\*：复合表达式的短路求值模式

### ### 3. 路径选择分析变式

- \*\*单一路径选择变式\*\*：简单**if-else**的路径选择分析
- \*\*多路径选择变式\*\*：**if-elif-else**链的路径选择
- \*\*嵌套路选择变式\*\*：嵌套分支的路径选择分析
- \*\*switch路径选择变式\*\*：**switch-case**的路径选择机制
- \*\*三元运算路径变式\*\*：三元运算符的路径选择
- \*\*条件跳转路径变式\*\*：**goto**等跳转的路径分析

### ### 4. 分支效果分析变式

- \*\*变量修改效果变式\*\*：分支对变量状态的修改效果
- \*\*累积效果变式\*\*：多个分支的累积效果分析
- \*\*副作用分析变式\*\*：分支执行的副作用分析
- \*\*资源影响变式\*\*：分支对资源状态的影响
- \*\*状态转换效果变式\*\*：分支导致的状态转换分析
- \*\*控制流效果变式\*\*：分支对后续控制流的影响

### ### 5. 条件复杂度变式

- \*\*简单条件变式\*\*：单一条件表达式的分析
- \*\*复合条件变式\*\*：多个条件组合的复杂表达式
- \*\*嵌套条件变式\*\*：条件表达式的嵌套结构
- \*\*动态条件变式\*\*：运行时变化的条件表达式
- \*\*递归条件变式\*\*：递归函数中的条件分析
- \*\*条件依赖变式\*\*：条件之间相互依赖的分析

### ### 6. 分支覆盖分析变式

- \*\*路径覆盖变式\*\*：所有可能路径的覆盖分析
- \*\*条件覆盖变式\*\*：条件真假值的覆盖分析
- \*\*分支覆盖变式\*\*：所有分支的执行覆盖
- \*\*MC/DC覆盖变式\*\*：修改条件/决策覆盖分析
- \*\*边界覆盖变式\*\*：边界条件的覆盖分析
- \*\*异常路径覆盖变式\*\*：异常情况的路径覆盖

### ### 7. 优先级和结合性变式

- \*\*运算符优先级变式\*\*：复杂表达式的运算优先级分析
- \*\*结合性分析变式\*\*：同优先级运算符的结合性
- \*\*括号影响变式\*\*：括号对运算顺序的影响
- \*\*类型提升变式\*\*：类型提升对条件求值的影响
- \*\*隐式转换变式\*\*：隐式类型转换的优先级
- \*\*表达式求值顺序变式\*\*：子表达式的求值顺序

### ### 8. 特殊分支结构变式

- \*\*fallthrough分析变式\*\*：**switch**语句的**fallthrough**行为
- \*\*break/continue效果变式\*\*：跳转语句的分支效果
- \*\*return分支变式\*\*：函数返回的分支分析
- \*\*异常分支变式\*\*：异常处理的分支逻辑
- \*\*goto分支变式\*\*：**goto**跳转的分支分析
- \*\*宏条件分支变式\*\*：预处理条件的分支分析

### ### 9. 数据类型特化变式

- \*\*数值比较分支变式\*\*：数值类型的比较分支分析

- \*\*字符串分支变式\*\*: 字符串比较的分支逻辑
- \*\*指针分支变式\*\*: 指针比较和NULL检查分支
- \*\*枚举分支变式\*\*: 枚举类型的分支处理
- \*\*浮点分支变式\*\*: 浮点数比较的特殊性分析
- \*\*位字段分支变式\*\*: 位字段操作的分支逻辑

### ### 10. 高级分支属性变式

- \*\*分支预测变式\*\*: 分支预测对性能的影响分析
- \*\*分支消除变式\*\*: 编译器分支优化的分析
- \*\*条件移动变式\*\*: 条件移动指令的分析
- \*\*分支密度变式\*\*: 代码中分支密度的分析
- \*\*分支复杂度变式\*\*: 分支结构复杂度的量化
- \*\*分支等价性变式\*\*: 不同分支实现的等价性分析

## ## 复杂度层次设计

### ### 简单分支属性 (Easy)

- 单一条件的简单分支分析
- 基础的条件求值和路径选择
- 明确的分支效果分析
- 直观的逻辑推理

### ### 中等分支属性 (Medium)

- 2-3个条件的复合分支分析
- 涉及短路求值的条件分析
- 多层条件嵌套的路径分析
- 中等复杂度的分支效果

### ### 复杂分支属性 (Hard)

- 复杂的条件表达式和嵌套结构
- 多重短路和副作用分析
- 复杂的路径选择和状态管理
- 涉及优化和性能的分支分析

### ### 专家级分支属性 (Expert)

- 极度复杂的条件逻辑分析
- 涉及编译器优化和架构特性
- 高级的分支覆盖和测试分析
- 需要深度计算机科学知识

## ## 生成策略

### ### 种子分析策略

1. \*\*识别核心属性\*\*: 分析种子任务关注的主要分支属性
2. \*\*提取分析模式\*\*: 识别条件求值和路径选择的模式
3. \*\*确定复杂度基准\*\*: 评估种子任务的复杂度水平
4. \*\*分析关键变量\*\*: 找出影响分支行为的关键因素

### ### 变式设计原则

1. \*\*属性导向\*\*: 每个变式都应明确关注特定的分支属性
2. \*\*英文描述\*\*: 所有task描述必须使用标准英文
3. \*\*数量保证\*\*: 严格确保每个种子生成8-12个变式
4. \*\*递进复杂\*\*: 从简单到复杂的渐进式设计

### 质量保证

1. \*\*逻辑验证\*\*：验证分支逻辑的正确性和一致性
2. \*\*路径检查\*\*：确保所有执行路径的分析正确
3. \*\*求值验证\*\*：手工验证条件求值的准确性
4. \*\*英文质量\*\*：确保描述的英文表达准确清晰

## 输出格式要求

```
```json
[
  {
    "id": "PL-BR-S00X-V001",
    "metadata": {
      "name": "PropertyLevel-Branch-VariantName",
      "category": "Property-Level",
      "subcategory": "Branch",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "property_focus": "condition_evaluation/path_selection/branch_effect/short_circuit/coverage",
      "analysis_depth": "surface/detailed/comprehensive"
    },
    "task": {
      "description": "English description of the branch property analysis task",
      "code": "Code containing branch structures to analyze",
      "answer": "Expected analysis result or property value"
    }
  },
  {下一个变式...}
]

```

特殊字段说明

property_focus: 标识主要关注的分支属性类型

analysis_depth: 标识分析的深度层次

variant_type: 描述具体的变式类型

生成目标

为每个提供的种子任务严格生成8-12个分支属性变式，确保：

每种难度等级至少包含2-3个变式

涵盖至少6-8种不同的分支属性分析类型

包含不同的分析深度和复杂度

所有**task**描述使用标准英文表达

所有答案必须准确且唯一

平衡条件求值、路径选择和分支效果三个核心方面

重点测试场景

条件求值精确性：准确分析复杂条件表达式的求值过程

短路逻辑理解：正确理解短路求值的机制和影响

路径选择分析：准确确定给定输入下的执行路径

分支效果追踪：正确分析分支对程序状态的影响

覆盖分析能力：分析分支覆盖和测试充分性

优化理解：理解编译器对分支的优化处理

英文描述模板示例

```
"Analyze the condition evaluation process and determine..."  
"Trace the path selection mechanism when..."  
"Evaluate the branch effects on variable states..."  
"Determine which conditions are evaluated due to short-circuit..."  
"Calculate the final result after all branch executions..."  
"Identify the execution path taken with the given inputs..."
```

请基于此提示词，为给定的分支属性推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，输出格式为包含所有变式的JSON数组。

```
[  
  {  
    "id": "PL-BR-S001-V001",  
    "metadata": {  
      "name": "PropertyLevel-Branch-SimpleConditionEvaluation",  
      "category": "Property-Level",  
      "subcategory": "Branch",  
      "type": "variant",  
      "source": "Generated",  
      "language": "c",  
      "difficulty": "easy",  
      "intervention": 0,  
      "variant_type": "simple_condition_evaluation",  
      "property_focus": "condition_evaluation",  
      "analysis_depth": "surface"  
    },  
    "task": {  
      "description": "Analyze the condition evaluation process and determine which branch is executed for the given input values.",  
      "code": "int check_range(int x, int y) {\n        if (x >= 0 && y <= 100) {\n          return 10;\n        } else {\n          return -10;\n        }\n      } // What value is returned when called with check_range(5, 50)?",  
      "answer": 10  
    },  
    {  
      "id": "PL-BR-S001-V002",  
      "metadata": {  
        "name": "PropertyLevel-Branch-ArithmeticConditionEvaluation",  
        "category": "Property-Level",  
        "subcategory": "Branch",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "medium",  
        "intervention": 1,  
        "variant_type": "arithmetic_condition_evaluation",  
        "property_focus": "condition_evaluation",  
        "analysis_depth": "detailed"  
      },  
    }  
]
```

```

"task": {
    "description": "Evaluate arithmetic expressions within conditions and trace the execution path taken.",
    "code": "int calculate_branch(int a, int b, int c) {\n    if ((a * 2) > (b + c) && (c - a) < b) {\n        return a + b;\n    } else if ((a + b) == c || (b * 2) > a) {\n        return a - c;\n    } else {\n        return b * c;\n    }\n}\n// What value is returned when called with calculate_branch(3, 7, 9)?",
    "answer": -6
},
{
    "id": "PL-BR-S001-V003",
    "metadata": {
        "name": "PropertyLevel-Branch-TypeConversionCondition",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "type_conversion_condition",
        "property_focus": "condition_evaluation",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Analyze condition evaluation involving implicit type conversion and determine the execution path.",
        "code": "int type_branch(float f, int i, double d) {\n    if (f > i && d < (i + 0.5)) {\n        return 1;\n    } else if (f <= i && (int)d == i) {\n        return 2;\n    } else {\n        return 3;\n    }\n}\n// What value is returned when called with type_branch(4.2, 4, 4.0)?",
        "answer": 1
    },
    {
        "id": "PL-BR-S001-V004",
        "metadata": {
            "name": "PropertyLevel-Branch-BitwiseConditionEvaluation",
            "category": "Property-Level",
            "subcategory": "Branch",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "bitwise_condition_evaluation",
            "property_focus": "condition_evaluation",
            "analysis_depth": "comprehensive"
        },
        "task": {
            "description": "Evaluate bitwise operations within conditional expressions and trace the branch selection."
        }
    }
}

```

```

        "code": "int bitwise_branch(int a, int b) {\n            if ((a & 0x0F) > 5 && (b | 0x10) == b) {\n                return a ^ b;\n            } else if ((a >> 2) == (b << 1) || (~a & 0xFF) > 200) {\n                return a & b;\n            } else {\n                return a | b;\n            }\n        }\\n// What value is returned when called with bitwise_branch(22, 16)?",
        "answer": 22
    },
},
{
    "id": "PL-BR-S001-V005",
    "metadata": {
        "name": "PropertyLevel-Branch-FunctionCallCondition",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "function_call_condition",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze condition evaluation when functions are called within the conditional expression.",
        "code": "int helper(int x) { return x % 3; }\\nint another(int y) { return y / 2; }\\n\\nint func_condition(int a, int b) {\n            if (helper(a) == 1 && another(b) > 3) {\n                return a + b;\n            } else if (helper(b) == 0 || another(a) <= 2) {\n                return a * b;\n            } else {\n                return a - b;\n            }\n        }\\n// What value is returned when called with func_condition(7, 8)?",
        "answer": 56
    },
},
{
    "id": "PL-BR-S001-V006",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedConditionalPath",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_conditional_path",
        "property_focus": "path_selection",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Trace the execution path through nested conditional structures and determine the final return value."
    }
}

```

```
        "code": "int nested_path(int x, int y, int z) {\n            if (x > 0) {\n                if (y > x) {\n                    if (z > y) return 1;\n                    else return 2;\n                } else {\n                    if (z < x) return 3;\n                    else return 4;\n                }\n            } else {\n                if (y < 0) return 5;\n                else return 6;\n            }\n        }\n    } // what value is returned when called with nested_path(5, 3, 7)?",\n    "answer": 4\n},\n{\n    "id": "PL-BR-S001-V007",\n    "metadata": {\n        "name": "PropertyLevel-Branch-CompoundBooleanEvaluation",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "compound_boolean_evaluation",\n        "property_focus": "condition_evaluation",\n        "analysis_depth": "comprehensive"\n    },\n    "task": {\n        "description": "Evaluate complex compound boolean expressions with multiple operators and determine the branch taken.",\n        "code": "int complex_boolean(int a, int b, int c, int d) {\n            if (((a > b) && (c < d)) || ((a + b) > (c + d)) && !((a % 2) == (b % 2))) {\n                return 100;\n            } else if ((a * b) > (c * d)) || ((a ^ b) == (c ^ d)) && ((a | b) < (c | d))) {\n                return 200;\n            } else {\n                return 300;\n            }\n        } // what value is returned when called with complex_boolean(4, 6, 3, 8)?",\n        "answer": 100\n    },\n},\n{\n    "id": "PL-BR-S001-V008",\n    "metadata": {\n        "name": "PropertyLevel-Branch-StringConditionEvaluation",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "medium",\n        "intervention": 1,\n        "variant_type": "string_condition_evaluation",\n        "property_focus": "condition_evaluation",\n        "analysis_depth": "detailed"\n    },\n    "task": {\n        "description": "Analyze string comparison conditions and determine which execution path is followed.",\n    }\n}
```

```

    "code": "#include <string.h>\nint string_branch(char* str1, char* str2, int len)\n{\n    if (strlen(str1) > len && strcmp(str1, str2) > 0) {\n        return 1;\n    } else if\n    (strlen(str2) <= len || strncmp(str1, str2, 3) == 0) {\n        return 2;\n    } else {\n        return 3;\n    }\n}\n// What value is returned when called with\nstring_branch(\"hello\", \"world\", 4)?",
    "answer": 2
}
},
{
    "id": "PL-BR-S001-V009",
    "metadata": {
        "name": "PropertyLevel-Branch-PointerConditionEvaluation",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "pointer_condition_evaluation",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate pointer-based conditions and trace the execution path\nthrough the branches.",
        "code": "int pointer_branch(int* p1, int* p2, int* p3) {\n    if (p1 != NULL &&\n*p1 > 10) {\n        if (p2 != NULL && *p2 < *p1) return *p1 + *p2;\n        else return\n*p1;\n    } else if (p3 != NULL && *p3 == 0) {\n        return -1;\n    } else {\n        return 0;\n    }\n}\nint a = 15, b = 8, c = 5;\n// What value is returned when called with\npointer_branch(&a, &b, &c)?",
        "answer": 23
    }
},
{
    "id": "PL-BR-S001-V010",
    "metadata": {
        "name": "PropertyLevel-Branch-FloatingPointPrecision",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "floating_point_precision",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze floating-point comparison precision issues and determine\nthe correct branch execution."
    }
}

```

```
        "code": "int float_precision_branch(double a, double b, double epsilon) {\nif ((a - b) < epsilon && (a - b) > -epsilon) {\n            return 1; // approximately equal\n} else if (a > b) {\n            return 2; // a is greater\n        } else {\n            return 3; // b\n        }\n    }\n\n    {\n        "id": "PL-BR-S002-V001",\n        "metadata": {\n            "name": "PropertyLevel-Branch-BasicShortCircuitAND",\n            "category": "Property-Level",\n            "subcategory": "Branch",\n            "type": "variant",\n            "source": "Generated",\n            "language": "c",\n            "difficulty": "easy",\n            "intervention": 0,\n            "variant_type": "basic_short_circuit_and",\n            "property_focus": "short_circuit",\n            "analysis_depth": "surface"\n        },\n        "task": {\n            "description": "Analyze short-circuit evaluation in AND operations and count\nfunction executions.",\n            "code": "int count = 0;\nint func1() { count++; return 0; }\nint func2() {\n    count++; return 1; }\n\nint result = func1() && func2();\n// What is the value of count\n// after this expression?",\n            "answer": 1\n        }\n    },\n    {\n        "id": "PL-BR-S002-V002",\n        "metadata": {\n            "name": "PropertyLevel-Branch-BasicShortCircuitOR",\n            "category": "Property-Level",\n            "subcategory": "Branch",\n            "type": "variant",\n            "source": "Generated",\n            "language": "c",\n            "difficulty": "easy",\n            "intervention": 0,\n            "variant_type": "basic_short_circuit_or",\n            "property_focus": "short_circuit",\n            "analysis_depth": "surface"\n        },\n        "task": {\n            "description": "Determine how many functions are called due to short-circuit\nevaluation in OR operations.",\n            "code": "int call_tracker = 0;\nint first() { call_tracker++; return 1; }\nint second() { call_tracker++; return 0; }\n\nint outcome = first() || second();\n// What is the\n// value of call_tracker after this expression?",\n            "answer": 1\n        }\n    }\n}
```

```
        "answer": 1
    }
},
{
    "id": "PL-BR-S002-V003",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedShortCircuit",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "nested_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Analyze nested short-circuit evaluation and count total function invocations.",
        "code": "int exec_count = 0;\nint f1() { exec_count++; return 1; }\nint f2() {\nexec_count++; return 0; }\nint f3() { exec_count++; return 1; }\nint f4() { exec_count++; return 1; }\n\nint result = (f1() && f2()) || (f3() && f4());\n// What is the value of exec_count after this expression?",
        "answer": 4
    }
},
{
    "id": "PL-BR-S002-V004",
    "metadata": {
        "name": "PropertyLevel-Branch-SideEffectShortCircuit",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "side_effect_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate short-circuit behavior with side effects and determine final variable values.",
        "code": "int global_var = 10;\nint modify_and_return(int val) { global_var += val; return val > 0; }\n\nint result = modify_and_return(-5) && modify_and_return(3) && modify_and_return(7);\n// What is the value of global_var after this expression?",
        "answer": 5
    }
},
{
```

```
"id": "PL-BR-S002-V005",
"metadata": {
    "name": "PropertyLevel-Branch-ConditionalChainShortCircuit",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "conditional_chain_short_circuit",
    "property_focus": "short_circuit",
    "analysis_depth": "comprehensive"
},
"task": {
    "description": "Analyze short-circuit evaluation in chained conditional expressions with multiple operators.",
    "code": "int ops = 0;\nint check1() { ops++; return 0; }\nint check2() { ops++; return 1; }\nint check3() { ops++; return 1; }\nint check4() { ops++; return 0; }\n\nfinal = check1() && check2() || check3() && check4();\n// What is the value of ops after this expression?",
    "answer": 3
},
{
    "id": "PL-BR-S002-V006",
    "metadata": {
        "name": "PropertyLevel-Branch-ArrayAccessShortCircuit",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "array_access_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Determine array access behavior under short-circuit evaluation conditions.",
        "code": "int access_count = 0;\nint safe_access(int arr[], int index, int size)\n{\n    access_count++;\n    return (index >= 0 && index < size) ? arr[index] : -1;\n}\n\nint numbers[] = {1, 2, 3};\nint result = (0 > 1) && (safe_access(numbers, 1, 3) > 0);\n// What is the value of access_count after this expression?",
        "answer": 0
    }
},
{
    "id": "PL-BR-S002-V007",
    "metadata": {
        "name": "PropertyLevel-Branch-ComplexShortCircuitPattern",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "complex_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "detailed"
    }
}
```

```
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "complex_short_circuit_pattern",
        "property_focus": "short_circuit",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze complex short-circuit patterns with mixed logical operators and function calls.",
        "code": "int counter = 0;\nint inc_and_check(int threshold) { return ++counter > threshold; }\n\nint complex_result = (inc_and_check(0) || inc_and_check(1)) && \n                           (inc_and_check(2) || inc_and_check(3)) && \n                           (inc_and_check(4) && inc_and_check(5));\n// what is the value of counter after this expression?",
        "answer": 4
    }
},
{
    "id": "PL-BR-S002-V008",
    "metadata": {
        "name": "PropertyLevel-Branch-TernaryShortCircuit",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "ternary_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Evaluate short-circuit behavior within ternary operator expressions.",
        "code": "int eval_count = 0;\nint evaluate(int val) { eval_count++; return val; }\n\nint x = 5;\nint result = (x > 0) ? evaluate(10) : (evaluate(20) && evaluate(30));\n// what is the value of eval_count after this expression?",
        "answer": 1
    }
},
{
    "id": "PL-BR-S002-V009",
    "metadata": {
        "name": "PropertyLevel-Branch-RecursiveshortCircuit",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "recursive_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "detailed"
    }
}
```

```
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "recursive_short_circuit",
        "property_focus": "short_circuit",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze short-circuit evaluation in recursive function calls and count total executions.",
        "code": "int recursion_count = 0;\nint recursive_check(int n) {\nrecursion_count++;\n    if (n <= 0) return 0;\n    return (n % 2 == 0) && recursive_check(n - 1);\n}\nint result = recursive_check(4); // What is the value of recursion_count after this call?",
        "answer": 2
    }
},
{
    "id": "PL-BR-S003-V001",
    "metadata": {
        "name": "PropertyLevel-Branch-SimpleTernaryEvaluation",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simple_ternary_evaluation",
        "property_focus": "path_selection",
        "analysis_depth": "surface"
    },
    "task": {
        "description": "Evaluate a simple ternary expression and determine the selected path result.",
        "code": "int a = 8, b = 12;\nint result = (a < b) ? a * 2 : b / 2; // What is the value of result?",
        "answer": 16
    }
},
{
    "id": "PL-BR-S003-V002",
    "metadata": {
        "name": "PropertyLevel-Branch-ChainedTernaryEvaluation",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "chained_ternary_evaluation",
        "property_focus": "path_selection",
        "analysis_depth": "surface"
    }
}
```

```
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Trace the execution path through chained ternary operators and calculate the final result.",
        "code": "int x = 6, y = 4, z = 2;\nint result = (x > y) ? (y > z ? x + y : x - z) : (y < z ? y + z : y - z);\n// what is the value of result?",
        "answer": 10
    }
},
{
    "id": "PL-BR-S003-V003",
    "metadata": {
        "name": "PropertyLevel-Branch-ArithmeticTernaryNesting",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "arithmetic_ternary_nesting",
        "property_focus": "path_selection",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate nested ternary expressions with complex arithmetic operations in each branch.",
        "code": "int a = 9, b = 6, c = 3;\nint result = (a % 3 == 0) ? ((b * 2) > (c * 4) ? a / c + b : a * c - b) : \n                           ((a + b) < (c * 5) ? a - b + c : a + b * c);\n// what is the value of result?",
        "answer": 21
    }
},
{
    "id": "PL-BR-S003-V004",
    "metadata": {
        "name": "PropertyLevel-Branch-FunctionCallTernary",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "function_call_ternary",
        "property_focus": "path_selection",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze ternary expressions containing function calls and determine which functions are executed."
    }
}
```

```

        "code": "int double_val(int x) { return x * 2; }\nint square_val(int x) { return x * x; }\nint add_five(int x) { return x + 5; }\n\nint n = 4;\nint result = (n > 3) ?\n    double_val(n) : (n == 2) ? square_val(n) : add_five(n);\n// What is the value of result?",\n    "answer": 8\n}\n},\n{\n    "id": "PL-BR-S003-V005",\n    "metadata": {\n        "name": "PropertyLevel-Branch-BitwiseTernaryOperations",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "hard",\n        "intervention": 2,\n        "variant_type": "bitwise_ternary_operations",\n        "property_focus": "path_selection",\n        "analysis_depth": "comprehensive"\n    },\n    "task": {\n        "description": "Evaluate ternary expressions involving bitwise operations and\ntrace the selected computation path.",\n        "code": "int x = 14, y = 7;\nint result = ((x & 0x0F) > (y << 1)) ? (x ^ y) :\n((x | y) > 15) ? (x & y) : (~x & 0xFF);\n// What is the value of result?",\n        "answer": 9\n    }\n},\n{\n    "id": "PL-BR-S003-V006",\n    "metadata": {\n        "name": "PropertyLevel-Branch-MultiLevelTernaryNesting",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "multi_level_ternary_nesting",\n        "property_focus": "path_selection",\n        "analysis_depth": "comprehensive"\n    },\n    "task": {\n        "description": "Navigate through deeply nested ternary expressions and calculate\nthe final selected value.",\n        "code": "int p = 10, q = 5, r = 3, s = 1;\nint result = (p > q) ? \n    ((q > r) ? \n        ((r > s) ? p + q + r + s : p - q + r - s) : \n        ((r < s) ? p * q - r : p / q + r)) : \n    ((q < r) ? q + r - s : q - r + s);\n// What is\nthe value of result?",\n        "answer": 19\n    }\n}

```

```

},
{
  "id": "PL-BR-S003-V007",
  "metadata": {
    "name": "PropertyLevel-Branch-ConditionalAssignmentTernary",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "conditional_assignment_ternary",
    "property_focus": "branch_effect",
    "analysis_depth": "detailed"
  },
  "task": {
    "description": "Analyze the effect of ternary operators on variable assignments and final state.",
    "code": "int base = 10;\nint modifier = 3;\nbase = (base > 5) ? (base + modifier) : (base - modifier);\nmodifier = (base % 2 == 1) ? modifier * 2 : modifier + 1;\n// what are the final values of base and modifier?",
    "answer": "base=13, modifier=6"
  }
},
{
  "id": "PL-BR-S003-V008",
  "metadata": {
    "name": "PropertyLevel-Branch-TernaryWithSideEffects",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "ternary_with_side_effects",
    "property_focus": "branch_effect",
    "analysis_depth": "comprehensive"
  },
  "task": {
    "description": "Evaluate ternary expressions with side effects and determine both result and side effect values.",
    "code": "int global_counter = 0;\nint increment() { return ++global_counter; }\nint decrement() { return --global_counter; }\n\nint x = 5;\nint result = (x > 0) ? increment() : (x < 0) ? decrement() : 0;\n// what are the values of result and global_counter?",
    "answer": "result=1, global_counter=1"
  }
},
{
  "id": "PL-BR-S003-V009",
  "metadata": {

```

```
        "name": "PropertyLevel-Branch-PointerTernarySelection",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "pointer_ternary_selection",
        "property_focus": "path_selection",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze ternary expressions involving pointer operations and determine the selected memory access.",
        "code": "int arr1[] = {10, 20, 30};\nint arr2[] = {40, 50, 60};\nint index = 1;\nint* selected = (index > 0) ? arr1 : arr2;\nint result = (selected[index] > 25) ? selected[index] + 5 : selected[index] - 5;\n// what is the value of result?",
        "answer": 15
    }
},
{
    "id": "PL-BR-S004-V001",
    "metadata": {
        "name": "PropertyLevel-Branch-BasicSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "basic_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "surface"
    },
    "task": {
        "description": "Calculate the accumulated value through basic switch statement fallthrough behavior.",
        "code": "int total = 0;\nint choice = 1;\nswitch (choice) {\n    case 1: total += 5;\n    case 2: total += 10;\n    case 3: total += 15;\n    break;\n    default: total = -1;\n}\n// what is the final value of total?",
        "answer": 30
    }
},
{
    "id": "PL-BR-S004-V002",
    "metadata": {
        "name": "PropertyLevel-Branch-PartialSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "partial_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "surface"
    }
}
```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "partial_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Analyze switch statement with mixed break and fallthrough patterns and calculate final state.",
        "code": "int accumulator = 5;\nint selector = 3;\nswitch (selector) {\n    case 1: accumulator *= 2; break;\n    case 2: accumulator += 10;\n    case 3: accumulator -= 3;\n    case 4: accumulator *= 3; break;\n    case 5: accumulator /= 2;\n    default: accumulator += 1;\n}\n// what is the final value of accumulator?",
        "answer": 6
    }
},
{
    "id": "PL-BR-S004-V003",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Trace execution through nested switch statements with complex fallthrough patterns.",
        "code": "int result = 0;\nint outer = 2, inner = 1;\nswitch (outer) {\n    case 1: result += 5; break;\n    case 2:\n        switch (inner) {\n            case 1: result += 10;\n            case 2: result += 20; break;\n            case 3: result += 30;\n        }\n        result += 40;\n    case 3: result += 50; break;\n}\n// what is the final value of result?",
        "answer": 120
    }
},
{
    "id": "PL-BR-S004-V004",
    "metadata": {
        "name": "PropertyLevel-Branch-CharacterSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "character_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "detailed"
    }
}
```

```

        "intervention": 1,
        "variant_type": "character_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Analyze character-based switch statement with fallthrough and calculate string processing result.",
        "code": "int vowel_count = 0, consonant_count = 0;\nchar ch = 'e';\nswitch (ch)\n    case 'a':\n    case 'e':\n    case 'i': vowel_count += 2;\n    case 'o':\n    case 'u': vowel_count += 1; break;\n    default: consonant_count += 1;\n}// what are the final values of vowel_count and consonant_count?",
        "answer": "vowel_count=3, consonant_count=0"
    }
},
{
    "id": "PL-BR-S004-V005",
    "metadata": {
        "name": "PropertyLevel-Branch-EnumSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "enum_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate enum-based switch statement with complex fallthrough logic and state changes.",
        "code": "enum State { INIT = 0, READY = 1, RUNNING = 2, PAUSED = 3, STOPPED = 4 };\nint status_value = 0;\nenum State current = RUNNING;\nswitch (current) {\n    case INIT: status_value = 10;\n    case READY: status_value += 20;\n    case RUNNING: status_value += 30; break;\n    case PAUSED: status_value += 40;\n    case STOPPED: status_value += 50;\n}\n// what is the final value of status_value?",
        "answer": 30
    }
},
{
    "id": "PL-BR-S004-V006",
    "metadata": {
        "name": "PropertyLevel-Branch-FunctionCallSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "function_call_switch_fallthrough",
    }
}

```

```
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze switch statement with function calls in fallthrough cases and track execution effects.",
        "code": "int global_sum = 0;\nvoid add_value(int val) { global_sum += val; }\nvoid multiply_value(int val) { global_sum *= val; }\n\nint mode = 2;\nswitch (mode) {\n    case 1: add_value(5);\n    case 2: add_value(10);\n    case 3: multiply_value(2); break;\n    case 4: add_value(20);\n    default: multiply_value(3);}\n// what is the final value of global_sum?",
        "answer": 20
    }
},
{
    "id": "PL-BR-S004-V007",
    "metadata": {
        "name": "PropertyLevel-Branch-ConditionalBreakSwitch",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "conditional_break_switch",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate switch statement with conditional break statements and analyze execution flow.",
        "code": "int counter = 0;\nint flag = 1;\nint input = 2;\nswitch (input) {\n    case 1: counter += 10; if (flag) break;\n    case 2: counter += 20; if (counter > 15) break;\n    case 3: counter += 30; if (flag == 0) break;\n    case 4: counter += 40;\n    default: counter += 5;}\n// what is the final value of counter?",
        "answer": 20
    }
},
{
    "id": "PL-BR-S004-V008",
    "metadata": {
        "name": "PropertyLevel-Branch-LoopControlSwitchFallthrough",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "loop_control_switch_fallthrough",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    }
}
```

```
        },
        "task": {
            "description": "Analyze switch statement with loop control statements and determine final iteration effects.",
            "code": "int total = 0;\nfor (int i = 1; i <= 3; i++) {\n    switch (i) {\n        case 1: total += i;\n        case 2: total += i * 2; if (i == 2) continue;\n        case 3: total += i * 3; break;\n        default: total += 100;\n    }\n    total += 1;\n}\n//\nWhat is the final value of total?",
            "answer": 25
        }
    },
    {
        "id": "PL-BR-S005-V001",
        "metadata": {
            "name": "PropertyLevel-Branch-SimplevariableModification",
            "category": "Property-Level",
            "subcategory": "Branch",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "easy",
            "intervention": 0,
            "variant_type": "simple_variable_modification",
            "property_focus": "branch_effect",
            "analysis_depth": "surface"
        },
        "task": {
            "description": "Trace simple variable modifications through sequential conditional branches.",
            "code": "value = 8\nif value > 5:\n    value = value + 3\nif value < 15:\n    value = value * 2\n# What is the final value of value?",
            "answer": 22
        }
    },
    {
        "id": "PL-BR-S005-V002",
        "metadata": {
            "name": "PropertyLevel-Branch-MultiplevariableTracking",
            "category": "Property-Level",
            "subcategory": "Branch",
            "type": "variant",
            "source": "Generated",
            "language": "python",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "multiple_variable_tracking",
            "property_focus": "branch_effect",
            "analysis_depth": "detailed"
        },
        "task": {
            "description": "Track modifications to multiple variables through interdependent conditional branches."
        }
    }
]
```

```

        "code": "a = 6\nb = 4\nif a > b:\n    a = a + 2\n    b = b - 1\nif (a + b) >\n10:\n    a = a // 2\n    b = b * 2\nif a == b:\n    a = a + 5\n# What are the final values\nof a and b?",\n        "answer": "a=4, b=6"\n    }\n},\n{\n    "id": "PL-BR-S005-V003",\n    "metadata": {\n        "name": "PropertyLevel-Branch-ListModificationTracking",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "medium",\n        "intervention": 1,\n        "variant_type": "list_modification_tracking",\n        "property_focus": "branch_effect",\n        "analysis_depth": "detailed"\n    },\n    "task": {\n        "description": "Analyze list modifications through conditional branches and\ndetermine final list state.",\n        "code": "numbers = [1, 2, 3]\nif len(numbers) > 2:\n    numbers.append(4)\nif\nnumbers[0] < numbers[-1]:\n    numbers[0] = numbers[0] * 2\nif sum(numbers) > 10:\n    numbers.pop()\n# What is the final state of numbers?",\n        "answer": "[2, 2, 3]"\n    }\n},\n{\n    "id": "PL-BR-S005-V004",\n    "metadata": {\n        "name": "PropertyLevel-Branch-DictionaryStateTracking",\n        "category": "Property-Level",\n        "subcategory": "Branch",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "hard",\n        "intervention": 2,\n        "variant_type": "dictionary_state_tracking",\n        "property_focus": "branch_effect",\n        "analysis_depth": "comprehensive"\n    },\n    "task": {\n        "description": "Track dictionary state changes through complex conditional logic\nand key operations.",\n        "code": "data = {'count': 5, 'total': 20}\nif 'count' in data and data['count']\n> 3:\n    data['total'] += data['count'] * 2\nif data.get('total', 0) > 25:\n    data['average'] = data['total'] // data['count']\nif 'average' in data:\n    data['count']\n+= 1\n# What is the final state of data?",\n        "answer": "{\n    'count': 6,\n    'total': 30,\n    'average': 6\n}"\n}
```

```
        }
    },
{
    "id": "PL-BR-S005-V005",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedObjectModification",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_object_modification",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze nested object modifications through conditional branches and track deep state changes.",
        "code": "config = {'settings': {'level': 2, 'active': True}, 'user': {'score': 100}}\nif config['settings']['active']:\n    config['user']['score'] += config['settings']['level'] * 10\nif config['user']['score'] > 115:\n    config['settings']['level'] += 1\nconfig['user']['bonus'] = True\nif 'bonus' in config['user']:\n    config['user']['score'] *= 1.1\n# what is the final value of config['user']['score']?",
        "answer": 132.0
    }
},
{
    "id": "PL-BR-S005-V006",
    "metadata": {
        "name": "PropertyLevel-Branch-StringModificationPattern",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "string_modification_pattern",
        "property_focus": "branch_effect",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Track string modifications through conditional branches based on string properties.",
        "code": "text = 'hello'\nif len(text) < 10:\n    text = text + ' world'\nif 'o' in text:\n    text = text.replace('o', '0')\nif text.count('l') > 2:\n    text = text.upper()\n# what is the final value of text?",
        "answer": "HELLO WORLD"
    }
},
{
```

```

    "id": "PL-BR-S005-V007",
    "metadata": {
        "name": "PropertyLevel-Branch-ConditionalFunctionModification",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "conditional_function_modification",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze variable modifications through conditional function calls and side effects.",
        "code": "state = 10\ndef modify_state(value):\n    global state\n    state += value\n    return state > 20\nif state < 15:\n    if modify_state(8):\n        modify_state(-5)\nif state % 2 == 1:\n    modify_state(3)\n# What is the final value of state?",
        "answer": 16
    }
},
{
    "id": "PL-BR-S005-V008",
    "metadata": {
        "name": "PropertyLevel-Branch-ClassAttributeModification",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "class_attribute_modification",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Track class attribute modifications through conditional branches and method calls.",
        "code": "class Counter:\n    def __init__(self):\n        self.value = 5\n    self.multiplier = 2\n    def update(self):\n        self.value *= self.multiplier\n    return self.value\n    c = Counter()\n    if c.value < 10:\n        if c.update() > 8:\n            c.multiplier += 1\n    if c.value > 5:\n        c.value /= c.multiplier\n# What is the final value of c.value?",
        "answer": 3
    }
},
{
    "id": "PL-BR-S005-V009",
    "metadata": {

```

```
        "name": "PropertyLevel-Branch-LoopVariableModification",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "loop_variable_modification",
        "property_focus": "branch_effect",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze variable modifications within loops with conditional control flow changes.",
        "code": "result = 0\nfor i in range(5):\n    if i % 2 == 0:\n        result += i\n    if result > 4:\n        result -= 1\n    if i > 2:\n        break\n    if i == 3:\n        result *= 2\n# what is the final value of result?",
        "answer": 5
    }
},
{
    "id": "PL-BR-S006-V001",
    "metadata": {
        "name": "PropertyLevel-Branch-SimpleMultipleConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "simple_multiple_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "surface"
    },
    "task": {
        "description": "Evaluate multiple simple conditions and calculate the accumulated result.",
        "code": "int calculate(int n) {\n    int result = n;\n    if (n > 5) result += 10;\n    if (n < 15) result -= 3;\n    return result;\n}\n// What value is returned when called with calculate(8)?",
        "answer": 15
    }
},
{
    "id": "PL-BR-S006-V002",
    "metadata": {
        "name": "PropertyLevel-Branch-ModuloBasedConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "modulo_based_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "surface"
    }
}
```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "modulo_based_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Analyze modulo-based conditions and trace the cumulative effects on the result value.",
        "code": "int modulo_process(int num) {\n    int output = num;\n    if (num % 4 == 0) output *= 2;\n    if (num % 6 == 0) output += 15;\n    if (num % 8 == 0) output /= 3;\n    return output;\n}\n// What value is returned when called with modulo_process(24)?",
        "answer": 26
    }
},
{
    "id": "PL-BR-S006-v003",
    "metadata": {
        "name": "PropertyLevel-Branch-RangeBasedConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "range_based_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Evaluate range-based conditions and determine the final processed value.",
        "code": "int range_checker(int val) {\n    int processed = val;\n    if (val >= 10 && val <= 20) processed += val / 2;\n    if (val >= 15 && val <= 25) processed -= 5;\n    if (val >= 5 && val <= 30) processed *= 1.2;\n    return processed;\n}\n// What value is returned when called with range_checker(18)?",
        "answer": 31
    }
},
{
    "id": "PL-BR-S006-v004",
    "metadata": {
        "name": "PropertyLevel-Branch-BitwiseMultipleConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "bitwise_multiple_conditions",
        "analysis_depth": "detailed"
    }
}
```

```

        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze multiple bitwise conditions and calculate the final transformed value.",
        "code": "int bitwise_transform(int x) {\n    int result = x;\n    if ((x & 0x01) == 1) result <= 1;\n    if ((x & 0x02) == 2) result |= 0x10;\n    if ((x & 0x04) == 4) result ^= 0xF0;\n    if ((x & 0x08) == 8) result &= 0xF0;\n    return result;\n}\n// what value is returned when called with bitwise_transform(14)?",
        "answer": 32
    }
},
{
    "id": "PL-BR-S006-v005",
    "metadata": {
        "name": "PropertyLevel-Branch-PowerBasedConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "power_based_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate power-of-two based conditions and trace the mathematical transformations.",
        "code": "int power_processor(int n) {\n    int value = n;\n    if ((n & (n - 1)) == 0 && n > 0) value += n; // power of 2\n    if (n % 3 == 1) value *= 2;\n    if (n > 10) value -= n / 4;\n    if (value % 7 == 0) value += 21;\n    return value;\n}\n// what value is returned when called with power_processor(16)?",
        "answer": 60
    }
},
{
    "id": "PL-BR-S006-v006",
    "metadata": {
        "name": "PropertyLevel-Branch-DigitSumConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "digit_sum_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze digit sum conditions and calculate the final transformed value.",
        "code": "int digit_sum_transform(int x) {\n    int result = x;\n    if ((x & 0x01) == 1) result <= 1;\n    if ((x & 0x02) == 2) result |= 0x10;\n    if ((x & 0x04) == 4) result ^= 0xF0;\n    if ((x & 0x08) == 8) result &= 0xF0;\n    return result;\n}\n// what value is returned when called with digit_sum_transform(14)?",
        "answer": 32
    }
}

```

```

"task": {
    "description": "Analyze conditions based on digit properties and calculate the processed result.",
    "code": "int digit_processor(int num) {\n    int result = num;\n    int digit_sum = (num / 100) + ((num / 10) % 10) + (num % 10);\n    if (digit_sum % 2 == 0) result += digit_sum;\n    if (digit_sum > 15) result *= 1.5;\n    if (num % digit_sum == 0) result -= 20;\n    return result;\n}\n// What value is returned when called with digit_processor(123)?",
    "answer": 112
},
{
    "id": "PL-BR-S006-V007",
    "metadata": {
        "name": "PropertyLevel-Branch-PrimeFactorConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "prime_factor_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate conditions based on prime factorization properties and determine the final value.",
        "code": "int prime_factor_process(int n) {\n    int result = n;\n    if (n % 2 == 0) { result /= 2; if (result % 2 == 0) result += 10; }\n    if (n % 3 == 0) { result += 6; if (n % 9 == 0) result *= 2; }\n    if (n % 5 == 0) { result -= 5; if (n % 25 == 0) result += 15; }\n    if (n % 7 == 0) result ^= 7;\n    return result;\n}\n// What value is returned when called with prime_factor_process(30)?",
        "answer": 22
    }
},
{
    "id": "PL-BR-S006-V008",
    "metadata": {
        "name": "PropertyLevel-Branch-FibonacciSequenceConditions",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "fibonacci_sequence_conditions",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {

```

```
        "description": "Analyze conditions based on Fibonacci sequence properties and mathematical relationships.",
        "code": "int is_fibonacci(int n) {\n    int a = 0, b = 1;\n    while (b < n) {\n        int temp = a + b; a = b; b = temp; }\n        return (b == n || n == 0);}\nint fibonacci_processor(int num) {\n    int result = num;\n    if (is_fibonacci(num)) result += num / 2;\n    if (num > 8 && is_fibonacci(num - 3)) result *= 2;\n    if (result % 13 == 0) result += 26;\n    return result;}// What value is returned when called with fibonacci_processor(13)?",
        "answer": 45
    },
    {
        "id": "PL-BR-S007-v001",
        "metadata": {
            "name": "PropertyLevel-Branch-SimpleLogicalEvaluation",
            "category": "Property-Level",
            "subcategory": "Branch",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "easy",
            "intervention": 0,
            "variant_type": "simple_logical_evaluation",
            "property_focus": "condition_evaluation",
            "analysis_depth": "surface"
        },
        "task": {
            "description": "Evaluate simple compound logical expression and determine the boolean result.",
            "code": "int x = 7, y = 4;\nint result = (x > y) && (x + y > 10); // What is the value of result (1 for true, 0 for false)?",
            "answer": 1
        }
    },
    {
        "id": "PL-BR-S007-v002",
        "metadata": {
            "name": "PropertyLevel-Branch-MixedOperatorLogic",
            "category": "Property-Level",
            "subcategory": "Branch",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "mixed_operator_logic",
            "property_focus": "condition_evaluation",
            "analysis_depth": "detailed"
        },
        "task": {
            "description": "Analyze compound logical expression with mixed arithmetic and comparison operators."
        }
    }
]
```

```
        "code": "int a = 6, b = 9, c = 2;\nint result = ((a * 2) < (b + c)) || ((b / 3)\n== c) && (a > c);\n// what is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
},
{
    "id": "PL-BR-S007-v003",
    "metadata": {
        "name": "PropertyLevel-Branch-NestedParenthesesLogic",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "nested_parentheses_logic",
        "property_focus": "condition_evaluation",
        "analysis_depth": "detailed"
    },
    "task": {
        "description": "Evaluate nested parentheses in compound logical expressions with proper operator precedence.",
        "code": "int p = 4, q = 7, r = 3;\nint result = ((p < q) && (q > r)) || (!(p > r) && (q - p == r));\n// what is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
},
{
    "id": "PL-BR-S007-v004",
    "metadata": {
        "name": "PropertyLevel-Branch-BitwiseLogicalCombination",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "bitwise_logical_combination",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze compound expressions combining bitwise and logical operations with complex precedence.",
        "code": "int x = 12, y = 8, z = 5;\nint result = ((x & 0x0F) > z) && ((y | 0x01) == 9) || ((x ^ y) < (z * 3)) && !(z & 0x01);\n// what is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
},
{

```

```
"id": "PL-BR-S007-V005",
"metadata": {
    "name": "PropertyLevel-Branch-ModuloComparisonLogic",
    "category": "Property-Level",
    "subcategory": "Branch",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "modulo_comparison_logic",
    "property_focus": "condition_evaluation",
    "analysis_depth": "comprehensive"
},
"task": {
    "description": "Evaluate compound logical expressions involving modulo operations and multiple comparisons.",
    "code": "int m = 15, n = 6, o = 4;\nint result = ((m % 5) == 0) && ((n % 3) == 0) || ((m % n) > o) && !((o % 2) == 0);\n// What is the value of result (1 for true, 0 for false)?",
    "answer": 1
},
{
    "id": "PL-BR-S007-V006",
    "metadata": {
        "name": "PropertyLevel-Branch-FloatingPointLogic",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "floating_point_logic",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
},
    "task": {
        "description": "Analyze compound logical expressions with floating-point comparisons and type conversions.",
        "code": "float f1 = 3.7, f2 = 2.3;\nint i1 = 4, i2 = 2;\nint result = ((f1 > i1) || (f2 < i2)) && ((int)f1 == i1) && !((f1 + f2) > (i1 + i2));\n// What is the value of result (1 for true, 0 for false)?",
        "answer": 0
},
{
    "id": "PL-BR-S007-V007",
    "metadata": {
        "name": "PropertyLevel-Branch-StringComparisonLogic",
        "category": "Property-Level",
        "subcategory": "Branch",
```

```
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "string_comparison_logic",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate compound logical expressions involving string operations and character comparisons.",
        "code": "#include <string.h>\nchar str1[] = \"test\";\nchar str2[] =\n\"demo\";\nint len1 = strlen(str1), len2 = strlen(str2);\nint result = ((len1 == len2) &&\n(str1[0] > str2[0])) || ((strcmp(str1, str2) > 0) && (len1 > 3));\n// What is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
},
{
    "id": "PL-BR-S007-v008",
    "metadata": {
        "name": "PropertyLevel-Branch-PointerArithmeticLogic",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "pointer_arithmetic_logic",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze compound logical expressions involving pointer arithmetic and memory comparisons.",
        "code": "int arr[] = {10, 20, 30, 40};\nint *p1 = &arr[1], *p2 = &arr[3];\nint diff = p2 - p1;\nint result = ((p1 != NULL) && (*p1 < *p2)) || ((diff == 2) && (*p1 + *p2 > 50)) && !(p1 > p2);\n// What is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
},
{
    "id": "PL-BR-S007-v009",
    "metadata": {
        "name": "PropertyLevel-Branch-FunctionReturnLogic",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
```

```

        "intervention": 3,
        "variant_type": "function_return_logic",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Evaluate compound logical expressions with function return values and complex operator precedence.",
        "code": "int func1(int x) { return x * 2; }\nint func2(int x) { return x + 3; }\nint func3(int x) { return x - 1; }\n\nint a = 5;\nint result = ((func1(a) > func2(a)) && (func3(a) < a)) ||\n                ((func1(a) + func3(a)) == (func2(a) * 3)) && !(func2(a) % 2 == 0);\n\n// what is the value of result (1 for true, 0 for false)?",
        "answer": 0
    }
},
{
    "id": "PL-BR-S007-V010",
    "metadata": {
        "name": "PropertyLevel-Branch-RecursiveLogicalEvaluation",
        "category": "Property-Level",
        "subcategory": "Branch",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "recursive_logical_evaluation",
        "property_focus": "condition_evaluation",
        "analysis_depth": "comprehensive"
    },
    "task": {
        "description": "Analyze compound logical expressions involving recursive function calls and state dependencies.",
        "code": "int factorial(int n) { return (n <= 1) ? 1 : n * factorial(n - 1); }\nint fibonacci(int n) { return (n <= 1) ? n : fibonacci(n - 1) + fibonacci(n - 2); }\n\nint n = 4;\nint result = ((factorial(n) > 20) && (fibonacci(n) < 5)) ||\n                ((factorial(n - 1) == 6) && (fibonacci(n + 1) > 4)) && !((factorial(n) % fibonacci(n)) == 0);\n\n// what is the value of result (1 for true, 0 for false)?",
        "answer": 1
    }
}
]

```

3C - 内存属性 [Memory] (59)

```
# 内存属性推理变式生成提示词
```

```
## 任务目标
```

基于给定的内存属性推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对内存结构深层属性的理解能力，包括引用关系、生命周期和访问模式等核心内存属性推理技能。

```
## 内存属性推理特征分析
```

内存属性推理关注程序内存使用的内在机制和数据关系，重点测试模型对指针操作、内存布局、数据引用和内存访问模式的深层理解，强调对内存语义的精确分析能力。

关键要求

- **描述语言**：所有**task**描述必须使用英文
- **变式数量**：每个种子任务严格生成**8-12**个变式，确保数量充足
- **答案唯一性**：保证答案准确且唯一，避免歧义和多解情况
- **属性聚焦**：重点关注内存的属性分析，而非简单的执行结果

变式生成维度

1. 引用关系分析变式

- **指针指向分析变式**：分析指针变量的指向关系和引用链
- **多级指针变式**：多重间接引用的指针关系分析
- **指针别名分析变式**：不同指针指向同一内存位置的分析
- **结构体指针变式**：结构体成员的指针引用关系
- **数组指针关系变式**：数组名与指针的等价性分析
- **函数指针引用变式**：函数指针的引用和调用分析

2. 生命周期分析变式

- **栈变量生命周期变式**：局部变量的生命周期分析
- **堆内存生命周期变式**：动态分配内存的生命周期
- **静态变量生命周期变式**：静态存储期变量的生命周期
- **临时对象生命周期变式**：临时对象的创建和销毁
- **作用域生命周期变式**：不同作用域中变量的生命周期
- **内存泄漏分析变式**：内存泄漏的识别和分析

3. 访问模式分析变式

- **顺序访问模式变式**：连续内存的顺序访问分析
- **随机访问模式变式**：非连续内存的随机访问分析
- **跨步访问模式变式**：固定间隔的内存访问模式
- **重叠访问模式变式**：内存区域重叠的访问分析
- **缓存友好访问变式**：缓存友好的内存访问模式
- **内存对齐访问变式**：内存对齐对访问的影响分析

4. 指针算术分析变式

- **指针递增递减变式**：指针自增自减操作的分析
- **指针加减运算变式**：指针与整数的加减运算
- **指针差值计算变式**：两个指针之间的距离计算
- **数组索引等价变式**：指针算术与数组索引的等价性
- **越界检测变式**：指针算术导致的越界访问检测
- **指针比较变式**：指针之间的大小比较分析

5. 内存布局分析变式

- **结构体布局变式**：结构体成员的内存布局分析
- **数组布局变式**：多维数组的内存布局和访问
- **内存对齐变式**：数据对齐对内存布局的影响
- **联合体布局变式**：联合体的内存共享分析
- **位字段布局变式**：位字段的内存布局和访问
- **填充字节变式**：结构体填充字节的分析

6. 内存拷贝和移动变式

- **内存拷贝操作变式**：`memcpy`等内存拷贝的分析

- **内存移动操作变式**: `memmove`的重叠内存处理
- **字符串拷贝变式**: `strcpy`等字符串拷贝操作
- **结构体拷贝变式**: 结构体赋值的内存拷贝分析
- **浅拷贝深拷贝变式**: 浅拷贝与深拷贝的区别分析
- **批量数据移动变式**: 大量数据的内存移动操作

7. 内存安全分析变式

- **缓冲区溢出变式**: 缓冲区溢出的检测和分析
- **空指针解引用变式**: 空指针访问的安全性分析
- **野指针访问变式**: 野指针的识别和风险分析
- **双重释放变式**: 内存双重释放的问题分析
- **使用已释放内存变式**: `use-after-free`问题的分析
- **内存权限变式**: 内存访问权限的安全性分析

8. 动态内存管理变式

- **内存分配变式**: `malloc/calloc`等动态分配的分析
- **内存释放变式**: `free`等内存释放操作的分析
- **内存重分配变式**: `realloc`内存重新分配的分析
- **内存池管理变式**: 内存池的分配和管理分析
- **垃圾回收变式**: 自动内存管理的分析
- **引用计数变式**: 引用计数内存管理的分析

9. 并发内存访问变式

- **竞态条件变式**: 并发访问的竞态条件分析
- **内存同步变式**: 内存访问的同步机制分析
- **原子操作变式**: 原子内存操作的分析
- **内存屏障变式**: 内存屏障对访问顺序的影响
- **锁保护内存变式**: 锁保护的内存访问分析
- **无锁数据结构变式**: 无锁数据结构的内存访问

10. 高级内存属性变式

- **内存映射变式**: 内存映射文件的访问分析
- **虚拟内存变式**: 虚拟内存机制的分析
- **内存压缩变式**: 内存压缩和优化的分析
- **内存碎片变式**: 内存碎片化的分析
- **内存局部性变式**: 内存访问局部性的分析
- **内存带宽变式**: 内存带宽使用的分析

复杂度层次设计

简单内存属性 (Easy)

- 基础的指针操作和数组访问
- 简单的引用关系分析
- 直接的内存访问模式
- 明确的生命周期边界

中等内存属性 (Medium)

- 2-3级的指针间接引用
- 结构体和数组的复合访问
- 中等复杂的内存布局分析
- 基础的内存安全问题

复杂内存属性 (Hard)

- 复杂的多级指针操作
- 重叠内存访问和复杂布局
- 动态内存管理的复杂场景
- 高级的内存安全分析

专家级内存属性 (Expert)

- 极度复杂的内存引用关系
- 并发环境的内存访问分析
- 系统级内存管理机制
- 需要深度系统知识的内存分析

生成策略

种子分析策略

1. **识别核心属性**: 分析种子任务关注的主要内存属性
2. **提取访问模式**: 识别内存访问的核心模式和特征
3. **确定引用关系**: 分析指针和引用的关系结构
4. **评估复杂度基准**: 评估种子任务的复杂度水平

变式设计原则

1. **属性导向**: 每个变式都应密切关注特定的内存属性
2. **英文描述**: 所有task描述必须使用标准英文
3. **答案唯一**: 严格确保答案的准确性和唯一性
4. **数量保证**: 严格确保每个种子生成8-12个变式

质量保证

1. **内存语义验证**: 验证内存操作的语义正确性
2. **指针关系检查**: 确保指针引用关系的准确性
3. **答案唯一性验证**: 严格检查答案的唯一性和确定性
4. **英文质量保证**: 确保描述的英文表达准确清晰

输出格式要求

```
```json
[
 {
 "id": "PL-MEM-S00X-V001",
 "metadata": {
 "name": "PropertyLevel-Memory-VariantName",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "target_language",
 "difficulty": "easy/medium/hard/expert",
 "intervention": 0,
 "variant_type": "variant_type_label",
 "property_focus": "reference_relation/lifetime/access_pattern/pointer_arithmetic/layout",
 "memory_operation": "read/write/allocate/deallocate/copy/move"
 },
 "task": {
 "description": "English description of the memory property analysis task",
 }
 }
]
```

```
 "code": "Code containing memory operations to analyze",
 "answer": "Unique and accurate analysis result or property value"
 }
},
{下一个变式...}
]
```

特殊字段说明

**property\_focus**: 标识主要关注的内存属性类型

**memory\_operation**: 标识主要的内存操作类型

**variant\_type**: 描述具体的变式类型

生成目标

为每个提供的种子任务严格生成8-12个内存属性变式，确保：

每种难度等级至少包含2-3个变式

涵盖至少6-8种不同的内存属性分析类型

包含不同的内存操作和访问模式

所有**task**描述使用标准英文表达

严格保证答案的准确性和唯一性

重点测试场景

指针引用追踪：准确追踪复杂指针引用关系

内存布局理解：正确理解数据结构的内存布局

生命周期分析：准确分析变量和内存的生命周期

访问模式识别：识别和分析内存访问模式

内存安全评估：评估内存操作的安全性

指针算术计算：准确计算指针算术操作结果

英文描述模板示例

"Calculate the memory address accessed after pointer arithmetic operations..."

"Determine the final value stored at the specified memory location..."

"Analyze the reference relationships between pointers and variables..."

"Trace the memory access pattern in the given code segment..."

"Evaluate the lifetime of variables in different scopes..."

"Calculate the result of overlapping memory operations..."

答案唯一性保证策略

精确计算：使用精确的数值计算，避免近似值

明确状态：确保程序状态在任何时刻都是确定的

避免未定义行为：避免产生未定义行为的代码

具体输入：提供具体的输入值，避免参数化答案

清晰边界：明确定义内存访问的边界和范围

请基于此提示词，为给定的内存属性推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

[

```
{
 "id": "PL-MEM-S001-V001",
 "metadata": {
 "name": "PropertyLevel-Memory-PointerDecrementAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
```

```
 "intervention": 1,
 "variant_type": "pointer_arithmetic_reversal",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
 },
 "task": {
 "description": "Calculate the final value accessed through pointer decrement operations.",
 "code": "int numbers[6] = {15, 25, 35, 45, 55, 65};\nint *ptr = numbers + 4;\nptr--;\nptr--;\nint result = *ptr;\n// What is the value of 'result'?",
 "answer": 25
 }
},
{
 "id": "PL-MEM-S001-V002",
 "metadata": {
 "name": "PropertyLevel-Memory-MultiplePointerJumps",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "multiple_pointer_operations",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
 },
 "task": {
 "description": "Determine the accessed value after multiple pointer arithmetic operations.",
 "code": "int sequence[7] = {100, 200, 300, 400, 500, 600, 700};\nint *ptr = sequence;\nptr += 3;\nptr -= 1;\nptr += 2;\nint final_value = *ptr;\n// What is the value of 'final_value'?",
 "answer": 500
 }
},
{
 "id": "PL-MEM-S001-V003",
 "metadata": {
 "name": "PropertyLevel-Memory-PointerArrayBoundary",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "boundary_access",
 "property_focus": "access_pattern",
 "memory_operation": "read"
 },
 "task": {
```

```
 "description": "Calculate the value accessed at array boundary through pointer arithmetic.",
 "code": "int data[8] = {11, 22, 33, 44, 55, 66, 77, 88};\nint *start_ptr = data + 1;\nint *end_ptr = data + 6;\nint *middle = start_ptr + ((end_ptr - start_ptr) / 2);\nint boundary_value = *middle;\n// What is the value of 'boundary_value'?",
 "answer": 44
 },
},
{
 "id": "PL-MEM-S001-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-PointerOffsetCalculation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "offset_calculation",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
 },
 "task": {
 "description": "Find the value accessed using calculated pointer offset.",
 "code": "int matrix[5] = {8, 16, 24, 32, 40};\nint *base = matrix;\nint offset = 2;\nint *target = base + offset + 1;\nint accessed_value = *target;\n// What is the value of 'accessed_value'?",
 "answer": 32
 },
},
{
 "id": "PL-MEM-S001-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-ReversePointerTraversal",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "reverse_traversal",
 "property_focus": "access_pattern",
 "memory_operation": "read"
 },
 "task": {
 "description": "Calculate the value accessed through reverse pointer traversal.",
 "code": "int values[6] = {12, 24, 36, 48, 60, 72};\nint *ptr = values + 5;\nfor (int i = 0; i < 3; i++) {\n ptr--;\n}\nint reverse_value = *ptr;\n// What is the value of 'reverse_value'?",
 "answer": 36
 }
}
```

```

 }
 },
 {
 "id": "PL-MEM-S001-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-ConditionalPointerAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_access",
 "property_focus": "access_pattern",
 "memory_operation": "read"
 },
 "task": {
 "description": "Determine the value accessed through conditional pointer movement.",
 "code": "int array[7] = {7, 14, 21, 28, 35, 42, 49};\nint *ptr = array + 2;\nif (*ptr + 1) > 25 {ptr += 2;} else {ptr -= 1;}\nint conditional_value = *ptr;\n// What is the value of 'conditional_value'?",
 "answer": 42
 }
 },
 {
 "id": "PL-MEM-S001-V007",
 "metadata": {
 "name": "PropertyLevel-Memory-PointerDifferenceAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "pointer_difference",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
 },
 "task": {
 "description": "Calculate the value accessed using pointer difference for indexing.",
 "code": "int buffer[9] = {9, 18, 27, 36, 45, 54, 63, 72, 81};\nint *ptr1 = buffer + 7;\nint *ptr2 = buffer + 3;\nint distance = ptr1 - ptr2;\nint *access_ptr = buffer + distance;\nint difference_value = *access_ptr;\n// What is the value of 'difference_value'?",
 "answer": 45
 }
 },
 {
 "id": "PL-MEM-S001-V008",

```

```
"metadata": {
 "name": "PropertyLevel-Memory-ChainedPointerAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "chained_operations",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
},
"task": {
 "description": "Find the final value through chained pointer arithmetic operations.",
 "code": "int chain[10] = {5, 10, 15, 20, 25, 30, 35, 40, 45, 50};\nint *p1 =\nchain + 3;\nint *p2 = p1 + 2;\nint *p3 = p2 - 1;\nint *final_ptr = p3 + (p2 - p1);\nint\nchained_value = *final_ptr;\n// What is the value of 'chained_value'?",
 "answer": 40
},
{
 "id": "PL-MEM-S001-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-ModuloPointerAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "modulo_indexing",
 "property_focus": "access_pattern",
 "memory_operation": "read"
},
"task": {
 "description": "Calculate the value accessed using modulo-based pointer indexing.",
 "code": "int circular[4] = {13, 26, 39, 52};\nint *base = circular;\nint index =\n7;\nint *modulo_ptr = base + (index % 4);\nint modulo_value = *modulo_ptr;\n// What is the\nvalue of 'modulo_value'?",
 "answer": 52
},
{
 "id": "PL-MEM-S001-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-BiDirectionalPointerscan",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "bidirectional_scans",
 "property_focus": "pointer_arithmetic",
 "memory_operation": "read"
}
}
```

```
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "bidirectional_scan",
 "property_focus": "access_pattern",
 "memory_operation": "read"
 },
 "task": {
 "description": "Determine the final value after bidirectional pointer scanning.",
 "code": "int scan_array[8] = {3, 6, 9, 12, 15, 18, 21, 24};\nint *left =\nscan_array + 1;\nint *right = scan_array + 6;\nwhile (right - left > 2) {\n left++;\n right--;\n}\nint scan_value = *left;\n// What is the value of 'scan_value'?",
 "answer": 12
 }
},
{
 "id": "PL-MEM-S002-V001",
 "metadata": {
 "name": "PropertyLevel-Memory-CrossReferenceModification",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "cross_reference",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the final array value after cross-reference modifications.",
 "code": "int values[5] = {2, 4, 6, 8, 10};\nint *p1 = &values[0];\nint *p2 =\n&values[3];\n*p1 = *p2 - 3;\n*p2 = *p1 + 7;\n// What is the value of values[3] after these operations?",
 "answer": 12
 }
},
{
 "id": "PL-MEM-S002-V002",
 "metadata": {
 "name": "PropertyLevel-Memory-CircularReferenceUpdate",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
```

```
 "variant_type": "circular_reference",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the array element value after circular reference updates.",
 "code": "int circle[6] = {10, 20, 30, 40, 50, 60};\nint *pa = &circle[1];\nint *pb = &circle[4];\nint *pc = &circle[2];\n*pa = *pb / 2;\n*pc = *pa * 3;\n*pb = *pc + 10;\n// what is the value of circle[4] after these operations?",
 "answer": 85
 }
},
{
 "id": "PL-MEM-S002-V003",
 "metadata": {
 "name": "PropertyLevel-Memory-ChainedModifications",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "chained_modifications",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of chained pointer-based modifications.",
 "code": "int chain[7] = {1, 3, 5, 7, 9, 11, 13};\nint *step1 = &chain[2];\nint *step2 = &chain[5];\nint *step3 = &chain[1];\n*step1 = *step2 + 4;\n*step3 = *step1 - 8;\n*step2 = *step3 * 2;\n// what is the value of chain[5] after these operations?",
 "answer": 14
 }
},
{
 "id": "PL-MEM-S002-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-ConditionalArrayUpdate",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_update",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
```

```
 "description": "Determine the array value after conditional pointer modifications.",
 "code": "int conditional[4] = {12, 24, 36, 48};\nint *ptr_a = &conditional[0];\nint *ptr_b = &conditional[2];\nif (*ptr_a < *ptr_b) {\n *ptr_a = *ptr_b / 3;\n *ptr_b = *ptr_a + 20;\n}\n// What is the value of conditional[2] after these operations?",
 "answer": 32
 }
},
{
 "id": "PL-MEM-S002-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-OverwriteSequence",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "overwrite_sequence",
 "property_focus": "lifetime",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the final value after sequential overwrites through pointers.",
 "code": "int overwrite[3] = {100, 200, 300};\nint *target = &overwrite[1];\nint temp = *target;\n*target = temp / 4;\n*target = *target + 25;\n// What is the value of overwrite[1] after these operations?",
 "answer": 75
 }
},
{
 "id": "PL-MEM-S002-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-SwapOperation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "swap_operation",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the array values after pointer-based swap operations."
 }
}
```

```
 "code": "int swap_data[4] = {15, 30, 45, 60};\nint *left = &swap_data[0];\nint\n*right = &swap_data[3];\nint temp = *left;\n*left = *right;\n*right = temp + 10;\n// What is\nthe value of swap_data[0] after these operations?",\n "answer": 60\n }\n},\n{\n "id": "PL-MEM-S002-V007",\n "metadata": {\n "name": "PropertyLevel-Memory-AccumulativeModification",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "accumulative_modification",\n "property_focus": "access_pattern",\n "memory_operation": "write"\n },\n "task": {\n "description": "Calculate the result of accumulative modifications through\nmultiple pointers.",\n "code": "int accumulate[5] = {5, 10, 15, 20, 25};\nint *p1 =\n&accumulate[1];\nint *p2 = &accumulate[3];\nint *p3 = &accumulate[2];\n*p3 += *p1;\n*p3 +=\n*p2;\n*p1 = *p3 / 3;\n// What is the value of accumulate[1] after these operations?",\n "answer": 15\n }\n},\n{\n "id": "PL-MEM-S002-V008",\n "metadata": {\n "name": "PropertyLevel-Memory-BitwisePointerModification",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "bitwise_modification",\n "property_focus": "reference_relation",\n "memory_operation": "write"\n },\n "task": {\n "description": "Determine the final value after bitwise operations through\npointers.",\n "code": "int bitwise[3] = {16, 32, 64};\nint *bp1 = &bitwise[0];\nint *bp2 =\n&bitwise[2];\n*bp1 = (*bp1 << 1) | 1;\n*bp2 = *bp2 >> 2;\n*bp1 = *bp1 & *bp2;\n// What is\nthe value of bitwise[0] after these operations?",\n "answer": 16\n }\n}
```

```

},
{
 "id": "PL-MEM-S002-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-NestedReferenceChain",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "nested_reference",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the value after nested reference chain modifications.",
 "code": "int nested[6] = {8, 16, 24, 32, 40, 48};\nint *n1 = &nested[1];\nint *n2 = &nested[*n1 / 8];\nint *n3 = &nested[*n2 / 8];\n*n3 = *n1 + *n2;\n*n1 = *n3 - 10;\n// What is the value of nested[1] after these operations?",
 "answer": 30
 }
},
{
 "id": "PL-MEM-S002-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-RecursivePointerUpdate",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_update",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the final value after recursive pointer-based updates.",
 "code": "int recursive[4] = {2, 4, 8, 16};\nint *r1 = &recursive[0];\nint *r2 = &recursive[2];\nfor (int i = 0; i < 2; i++) {\n *r1 = *r1 * 2;\n *r2 = *r2 +\n *r1;\n}\n// What is the value of recursive[2] after the loop?",
 "answer": 20
 }
},
{
 "id": "PL-MEM-S003-V001",
 "metadata": {

```

```
 "name": "PropertyLevel-Memory-NestedStructModification",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_struct",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the nested struct member value after pointer operations.",
 "code": "typedef struct { int a; int b; } Inner;\ntypedef struct { Inner in; int c; } Outer;\nOuter data[2] = {{10, 20}, 30}, {{40, 50}, 60};\nOuter *ptr = &data[0];\nptr->in.a = ptr->in.a + data[1].in.b;\nptr->c = ptr->in.a - 15;\n// What is the value of data[0].c after these operations?",
 "answer": 45
 }
},
{
 "id": "PL-MEM-S003-V002",
 "metadata": {
 "name": "PropertyLevel-Memory-StructArrayTraversal",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "array_traversal",
 "property_focus": "access_pattern",
 "memory_operation": "read"
 },
 "task": {
 "description": "Determine the struct member value after array traversal with pointers.",
 "code": "typedef struct { int id; int value; } Node;\nNode nodes[4] = {{1, 100}, {2, 200}, {3, 300}, {4, 400}};\nNode *current = nodes + 1;\nncurrent += 2;\nncurrent->value = (current - 1)->value + current->id;\n// What is the value of nodes[3].value after this operation?",
 "answer": 304
 }
},
{
 "id": "PL-MEM-S003-V003",
 "metadata": {
 "name": "PropertyLevel-Memory-StructMemberSwap",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "member_swap",
 "property_focus": "layout",
 "memory_operation": "read"
 }
}
```

```
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "member_swap",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the struct member value after swapping operations.",
 "code": "typedef struct { int first; int second; } Pair;\nPair items[3] = {{12, 24}, {36, 48}, {60, 72}};\nPair *p1 = &items[0];\nPair *p2 = &items[2];\nint temp = p1->first;\n*p1->first = p2->second;\n*p2->second = temp + 10;\n// What is the value of items[0].first after these operations?",
 "answer": 72
 }
},
{
 "id": "PL-MEM-S003-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-StructFieldCalculation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "field_calculation",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the struct field value after mathematical operations through pointers.",
 "code": "typedef struct { int width; int height; int area; } Rectangle;\nRectangle rect[2] = {{5, 10, 0}, {8, 12, 0}};\nRectangle *r = rect + 1;\nrect->area = r->width * r->height;\n(r - 1)->area = (r - 1)->width * (r - 1)->height + r->area;\n// What is the value of rect[0].area after these operations?",
 "answer": 146
 }
},
{
 "id": "PL-MEM-S003-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-StructPointerChain",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3
 }
}
```

```

 "intervention": 3,
 "variant_type": "pointer_chain",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the final struct value through pointer chain operations.",
 "code": "typedef struct { int key; int data; } Record;\nRecord records[5] = {{1, 11}, {2, 22}, {3, 33}, {4, 44}, {5, 55}};\nRecord *p1 = records + 1;\nRecord *p2 = records + p1->key + 1;\nRecord *p3 = records + p2->key;\n*p3->data = p1->data + p2->data;\n// what is the value of records[4].data after this operation?",
 "answer": 66
 }
},
{
 "id": "PL-MEM-S003-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-StructArrayRotation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "array_rotation",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the struct value after array rotation through pointers.",
 "code": "typedef struct { int x; int y; } Vector;\nVector vectors[3] = {{10, 20}, {30, 40}, {50, 60}};\nVector *v = vectors;\nint temp_x = v->x;\n*v->x = (v + 1)->x;\n(v + 1)->x = (v + 2)->x;\n*(v + 2)->x = temp_x + 5;\n// what is the value of vectors[2].x after these operations?",
 "answer": 15
 }
},
{
 "id": "PL-MEM-S003-V007",
 "metadata": {
 "name": "PropertyLevel-Memory-ConditionalStructUpdate",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_update",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 }
}

```

```
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the struct member value after conditional updates.",
 "code": "typedef struct { int min; int max; } Range;\nRange ranges[3] = {{10, 50}, {20, 60}, {30, 70}};\nRange *r = ranges + 1;\nif (r->min < (r - 1)->max) {\n r->min = (r - 1)->max + 10;\n r->max = r->min + 25;\n}\n// What is the value of ranges[1].max after these operations?",
 "answer": 85
 }
},
{
 "id": "PL-MEM-S003-V008",
 "metadata": {
 "name": "PropertyLevel-Memory-StructMemberAggregation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "member_aggregation",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the aggregated struct member value through pointer operations.",
 "code": "typedef struct { int count; int sum; int avg; } Statistics;\nStatistics stats[2] = {{3, 30, 0}, {5, 100, 0}};\nStatistics *s1 = &stats[0];\nStatistics *s2 = &stats[1];\ns1->avg = s1->sum / s1->count;\ns2->avg = s2->sum / s2->count;\ns1->sum = s1->avg + s2->avg;\n// What is the value of stats[0].sum after these operations?",
 "answer": 30
 }
},
{
 "id": "PL-MEM-S003-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-StructBitwiseOperation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "bitwise_operation",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
```

```

 "description": "Calculate the struct member value after bitwise operations
through pointers.",
 "code": "typedef struct { int flags; int mask; } BitField;\nBitField fields[2] =
{{0x0F, 0xFF}, {0xF0, 0x0F}};\nBitField *bf1 = &fields[0];\nBitField *bf2 =
&fields[1];\nbf1->flags = bf1->flags | bf2->flags;\nbf1->mask = bf1->mask & bf2->mask;\n//\nwhat is the value of fields[0].mask after these operations?",
 "answer": 15
 },
},
{
 "id": "PL-MEM-S003-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-StructOffsetAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "offset_access",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the struct value using calculated offset-based
access.",
 "code": "typedef struct { int alpha; int beta; int gamma; } Triple;\nTriple
triples[4] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};\nTriple *base = triples;\nint
offset = 2;\n(base + offset)->beta = (base + 1)->alpha + (base + offset - 1)->gamma;\n//\nwhat is the value of triples[2].beta after this operation?",
 "answer": 10
 },
},
{
 "id": "PL-MEM-S004-V001",
 "metadata": {
 "name": "PropertyLevel-Memory-QuadrupleIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "quadruple_indirection",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the value after quadruple pointer indirection
operations."
 }
}

```

```

 "code": "int base_value = 80;\nint *level1 = &base_value;\nint **level2 =\n&level1;\nint ***level3 = &level2;\nint ****level4 = &level3;\n\n***level4 = ***level4 +\n40;\n// what is the value of 'base_value' after this operation?",\n "answer": 120\n }\n},\n{\n "id": "PL-MEM-S004-V002",\n "metadata": {\n "name": "PropertyLevel-Memory-IndirectionChainModification",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "chain_modification",\n "property_focus": "reference_relation",\n "memory_operation": "write"\n },\n "task": {\n "description": "Determine the final value through indirection chain\nmodifications.",\n "code": "int target = 60;\nint *ptr_a = ⌖\nint **ptr_b = &ptr_a;\nint\n***ptr_c = &ptr_b;\n\n*ptr_a = *ptr_a * 2;\n**ptr_b = **ptr_b - 30;\n\n***ptr_c = ***ptr_c +\n15;\n// what is the value of 'target' after these operations?",\n "answer": 105\n }\n},\n{\n "id": "PL-MEM-S004-V003",\n "metadata": {\n "name": "PropertyLevel-Memory-MultiLevelPointerswap",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "multilevel_swap",\n "property_focus": "reference_relation",\n "memory_operation": "write"\n },\n "task": {\n "description": "Calculate the value after multi-level pointer swap operations.",\n "code": "int val1 = 25, val2 = 75;\nint *p1 = &val1, *p2 = &val2;\nint **pp1 =\n&p1, **pp2 = &p2;\nint ***ppp = &pp1;\nint temp = **pp1;\n**pp1 = **pp2;\n**pp2 = temp +\n10;\n// what is the value of 'val1' after these operations?",\n "answer": 75\n }\n},\n{

```

```
{
 "id": "PL-MEM-S004-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-ConditionalIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "conditional_indirection",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after conditional multi-level indirection.",
 "code": "int data = 45;\nint *p1 = &data;\nint **p2 = &p1;\nint ***p3 = &p2;\nif\n(**p3 > 40) {\n ***p3 = ***p3 * 2;\n} else {\n ***p3 = ***p3 + 20;\n}\n// What is the value of 'data' after this operation?",
 "answer": 90
 }
},
{
 "id": "PL-MEM-S004-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-PointerArrayIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "array_indirection",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the array value through pointer array indirection.",
 "code": "int values[3] = {12, 24, 36};\nint *ptrs[3] = {&values[0], &values[1],\n&values[2]};\nint **ptr_to_ptrs = ptrs;\nint ***triple_ptr = &ptr_to_ptrs;\n**((*triple_ptr)\n+ 1) = **((*triple_ptr) + 2) + 8;\n\n// What is the value of values[1] after this operation?",
 "answer": 44
 }
},
{
 "id": "PL-MEM-S004-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-RecursiveIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_indirection",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 }
}
```

```
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_indirection",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the final value through recursive indirection operations.",
 "code": "int result = 10;\nint *level1 = &result;\nint **level2 = &level1;\nint ***level3 = &level2;\nfor (int i = 0; i < 3; i++) {\n ***level3 = ***level3 + 5;\n}\n// What is the value of 'result' after the loop?",
 "answer": 25
 }
},
{
 "id": "PL-MEM-S004-V007",
 "metadata": {
 "name": "PropertyLevel-Memory-IndirectionCalculation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "calculation_indirection",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the value using mathematical operations through indirection.",
 "code": "int operand = 16;\nint *op_ptr = &operand;\nint **op_ptr_ptr = &op_ptr;\nint ***op_ptr_ptr_ptr = &op_ptr_ptr;\n***op_ptr_ptr_ptr = (**op_ptr_ptr_ptr << 1) + 8;\n// What is the value of 'operand' after this operation?",
 "answer": 40
 }
},
{
 "id": "PL-MEM-S004-V008",
 "metadata": {
 "name": "PropertyLevel-Memory-BitwiseIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "bitwise_indirection",
 "memory_operation": "write"
 }
}
```

```

 "property_focus": "reference_relation",
 "memory_operation": "write"
},
"task": {
 "description": "Determine the value after bitwise operations through multiple
indirection levels.",
 "code": "int bits = 0x0F;\nint *bit_ptr = &bits;\nint **bit_ptr_ptr =
&bit_ptr;\nint ***bit_ptr_ptr_ptr = &bit_ptr_ptr;\n***bit_ptr_ptr_ptr = (***bit_ptr_ptr_ptr
<< 4) | 0x05;\n// what is the value of 'bits' after this operation?",
 "answer": 245
}
},
{
 "id": "PL-MEM-S004-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-ChainedIndirectionUpdate",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "chained_update",
 "property_focus": "reference_relation",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the final value through chained indirection updates.",
 "code": "int chain_value = 20;\nint *c1 = &chain_value;\nint **c2 = &c1;\nint
c3 = &c2;\nint temp = ***c3;\nc3 = temp + 15;\nint temp = ***c3;\n***c3 = temp * 2;\n//
what is the value of 'chain_value' after these operations?",
 "answer": 70
 }
},
{
 "id": "PL-MEM-S004-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-NestedStructIndirection",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "struct_indirection",
 "property_focus": "layout",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the struct member value through nested indirection."
 }
}

```

```
 "code": "typedef struct { int data; } Container;\nContainer container =\n{30};\nContainer *cont_ptr = &container;\nContainer **cont_ptr_ptr = &cont_ptr;\nContainer ***cont_ptr_ptr_ptr = &cont_ptr_ptr;\n(**cont_ptr_ptr_ptr).data =\n(**cont_ptr_ptr_ptr).data + 25;\n// What is the value of container.data after this\noperation?",\n "answer": 55\n }\n},\n{\n "id": "PL-MEM-S005-V001",\n "metadata": {\n "name": "PropertyLevel-Memory-StridedArrayCopy",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "strided_copy",\n "property_focus": "access_pattern",\n "memory_operation": "copy"\n },\n "task": {\n "description": "Calculate the array value after strided copy operations.",\n "code": "int original[8] = {2, 4, 6, 8, 10, 12, 14, 16};\nint copied[8] = {0, 0,\n0, 0, 0, 0, 0, 0};\nfor (int i = 0; i < 4; i++) {\n copied[i * 2] = original[i] *\n3;\n}\n// What is the value of copied[6] after the loop?",\n "answer": 24\n }\n},\n{\n "id": "PL-MEM-S005-V002",\n "metadata": {\n "name": "PropertyLevel-Memory-ReverseArrayCopy",\n "category": "Property-Level",\n "subcategory": "Memory",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "medium",\n "intervention": 1,\n "variant_type": "reverse_copy",\n "property_focus": "access_pattern",\n "memory_operation": "copy"\n },\n "task": {\n "description": "Determine the array element after reverse copy operations.",\n "code": "int src[6] = {10, 20, 30, 40, 50, 60};\nint dst[6] = {0, 0, 0, 0, 0,\n0};\nfor (int i = 0; i < 6; i++) {\n dst[i] = src[5 - i] + 5;\n}\n// What is the value of dst[2] after the loop?",\n "answer": 45\n }\n}
```

```

 }
 },
 {
 "id": "PL-MEM-S005-V003",
 "metadata": {
 "name": "PropertyLevel-Memory-OverlappingCopyPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "overlapping_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Calculate the result of overlapping copy pattern operations.",
 "code": "int pattern[7] = {1, 3, 5, 7, 9, 11, 13};\nfor (int i = 1; i < 4; i++)\n pattern[i + 2] = pattern[i] + pattern[i - 1];\n// what is the value of pattern[5] after the loop?",
 "answer": 8
 }
 },
 {
 "id": "PL-MEM-S005-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-ConditionalCopyOperation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Determine the array value after conditional copy operations.",
 "code": "int input[5] = {15, 25, 35, 45, 55};\nint output[5] = {0, 0, 0, 0, 0};\nfor (int i = 0; i < 5; i++) {\n if (input[i] % 10 == 5) {\n output[i] = input[i] / 5;\n } else {\n output[i] = input[i] * 2;\n }\n}\n// what is the value of output[3] after the loop?",
 "answer": 9
 }
 },
 {
 "id": "PL-MEM-S005-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-BlockCopyWithOffset",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "block_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Determine the array value after block copy with offset operations.",
 "code": "int input[5] = {15, 25, 35, 45, 55};\nint output[5] = {0, 0, 0, 0, 0};\nint offset = 2;\nfor (int i = 0; i < 5; i++) {\n output[i] = input[i + offset];\n}\n// what is the value of output[3] after the loop?",
 "answer": 10
 }
 }
]
```

```
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "block_copy_offset",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Calculate the array element after block copy with offset.",
 "code": "int block[9] = {100, 200, 300, 400, 500, 600, 700, 800, 900};\nfor (int i = 0; i < 3; i++) {\n block[i + 3] = block[i] + 50;\n}\n// What is the value of block[4] after the loop?",
 "answer": 250
 }
},
{
 "id": "PL-MEM-S005-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-InterleavedCopyPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "interleaved_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Determine the value after interleaved copy pattern operations.",
 "code": "int even[4] = {2, 6, 10, 14};\nint odd[4] = {1, 5, 9, 13};\nint interleaved[8] = {0, 0, 0, 0, 0, 0, 0, 0};\nfor (int i = 0; i < 4; i++) {\n interleaved[i * 2] = even[i];\n interleaved[i * 2 + 1] = odd[i] + 2;\n}\n// What is the value of interleaved[5] after the loop?",
 "answer": 7
 }
},
{
 "id": "PL-MEM-S005-V007",
 "metadata": {
 "name": "PropertyLevel-Memory-CumulativeCopySum",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "cumulative_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 }
}
```

```

 "intervention": 2,
 "variant_type": "cumulative_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Calculate the cumulative sum result after copy operations.",
 "code": "int values[6] = {5, 10, 15, 20, 25, 30};\nint cumulative[6] = {0, 0, 0, 0, 0};\n\nint cumulative[0] = values[0];\nfor (int i = 1; i < 6; i++) {\n cumulative[i] = cumulative[i - 1] + values[i];\n}\n// What is the value of cumulative[4] after the loop?",
 "answer": 75
 }
},
{
 "id": "PL-MEM-S005-V008",
 "metadata": {
 "name": "PropertyLevel-Memory-MatrixRowCopy",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "matrix_copy",
 "property_focus": "layout",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Determine the matrix element after row copy operations.",
 "code": "int matrix[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};\nint copied_row[4] = {0, 0, 0, 0};\nfor (int j = 0; j < 4; j++) {\n copied_row[j] = matrix[1][j] * 2;\n}\nmatrix[2][1] = copied_row[1] + copied_row[3];\n// What is the value of matrix[2][1] after these operations?",
 "answer": 28
 }
},
{
 "id": "PL-MEM-S005-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-CircularBufferCopy",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "circular_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {

```

```

 "description": "Calculate the circular buffer value after copy operations.",
 "code": "int circular[5] = {10, 20, 30, 40, 50};\nint temp[5] = {0, 0, 0, 0, 0};\nfor (int i = 0; i < 5; i++) {\n temp[i] = circular[(i + 2) % 5];\n}\nfor (int i = 0; i < 5; i++) {\n circular[i] = temp[i] + 5;\n}// What is the value of circular[1] after these operations?",
 "answer": 45
 },
 {
 "id": "PL-MEM-S005-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-FilteredCopyOperation",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "filtered_copy",
 "property_focus": "access_pattern",
 "memory_operation": "copy"
 },
 "task": {
 "description": "Determine the filtered copy result after selective operations.",
 "code": "int source[8] = {12, 18, 24, 30, 36, 42, 48, 54};\nint filtered[8] = {0, 0, 0, 0, 0, 0, 0, 0};\nint write_pos = 0;\nfor (int i = 0; i < 8; i++) {\n if (source[i] % 6 == 0) {\n filtered[write_pos] = source[i] / 6;\n write_pos++;\n }\n}\n// What is the value of filtered[3] after the loop?",
 "answer": 5
 }
 },
 {
 "id": "PL-MEM-S006-V001",
 "metadata": {
 "name": "PropertyLevel-Memory-slidingwindowAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "sliding_window",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of sliding window overlapping access pattern."
 }
 }
]
```

```

 "code": "int window[8] = {5, 10, 15, 20, 25, 30, 35, 40};\nint *left = window + 1;\nint *right = window + 5;\nfor (int i = 0; i < 3; i++) {\n left[i] = right[i] + left[i + 1];\n right[i - 1] = left[i] - 5;\n}\n// What is the value of window[4] after the loop?",

 "answer": 40
 }
},
{
 "id": "PL-MEM-S006-V002",
 "metadata": {
 "name": "PropertyLevel-Memory-CrossPatternAccess",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "cross_pattern",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after cross-pattern overlapping access operations.",

 "code": "int cross[9] = {8, 16, 24, 32, 40, 48, 56, 64, 72};\nint *center = cross + 4;\nfor (int offset = 1; offset <= 2; offset++) {\n center[-offset] = center[offset] + center[-offset + 1];\n center[offset] = center[-offset] * 2;\n}\n// What is the value of cross[6] after the loop?",

 "answer": 256
 }
},
{
 "id": "PL-MEM-S006-V003",
 "metadata": {
 "name": "PropertyLevel-Memory-DiagonalAccessPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "diagonal_access",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of diagonal overlapping access pattern.",

 "code": "int diagonal[7] = {7, 14, 21, 28, 35, 42, 49};\nint *p1 = diagonal + 1;\nint *p2 = diagonal + 4;\nfor (int step = 0; step < 2; step++) {\n p1[step * 2] = p2[step] + p1[step];\n p2[step - 1] = p1[step * 2] - 7;\n}\n// What is the value of diagonal[3] after the loop?",

 "answer": 42
 }
}

```

```

 "answer": 42
 }
},
{
 "id": "PL-MEM-S006-V004",
 "metadata": {
 "name": "PropertyLevel-Memory-SpiralAccessPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "spiral_access",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after spiral overlapping access operations.",
 "code": "int spiral[10] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};\nint *outer =\nspiral + 1;\nint *inner = spiral + 6;\nfor (int rotation = 0; rotation < 3; rotation++) {\n outer[rotation] = inner[rotation] + outer[rotation + 1];\n inner[rotation + 1] =\n outer[rotation] / 2;\n}\n// What is the value of spiral[8] after the loop?",
 "answer": 10
 }
},
{
 "id": "PL-MEM-S006-V005",
 "metadata": {
 "name": "PropertyLevel-Memory-WaveformAccessPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "waveform_access",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of waveform overlapping access pattern.",
 "code": "int waveform[12] = {3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36};\nint\n*wave1 = waveform + 2;\nint *wave2 = waveform + 7;\nfor (int phase = 0; phase < 4; phase++) {\n if (phase % 2 == 0) {\n wave1[phase] = wave2[phase] - wave1[phase - 1];\n }\n else {\n wave2[phase - 2] = wave1[phase] + 6;\n }\n}\n// What is the value of waveform[5] after the loop?",
 "answer": 15
 }
},

```

```
{
 "id": "PL-MEM-S006-V006",
 "metadata": {
 "name": "PropertyLevel-Memory-FibonacciOverlapPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "fibonacci_overlap",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after Fibonacci-style overlapping operations.",
 "code": "int fib_array[8] = {1, 1, 2, 3, 5, 8, 13, 21};\nint *fib_ptr =\n fib_array + 2;\nfor (int i = 0; i < 4; i++) {\n fib_ptr[i + 2] = fib_ptr[i] + fib_ptr[i + 1] + 1;\n fib_ptr[i] = fib_ptr[i + 2] - fib_ptr[i + 1];\n}\n// What is the value of fib_array[4] after the loop?",
 "answer": 2
 }
},
{
 "id": "PL-MEM-S006-V007",
 "metadata": {
 "name": "PropertyLevel-Memory-PyramidAccessPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "pyramid_access",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of pyramid overlapping access pattern.",
 "code": "int pyramid[11] = {11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121};\nint\n*apex = pyramid + 5;\nfor (int level = 1; level <= 3; level++) {\n apex[-level] = apex[0] + apex[level];\n apex[level] = apex[-level] - level * 11;\n}\n// What is the value of pyramid[8] after the loop?",
 "answer": 99
 }
},
{
 "id": "PL-MEM-S006-V008",
 "metadata": {
 "name": "PropertyLevel-Memory-ZigzagOverlapPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "zigzag_overlap",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 }
}
```

```
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "zigzag_overlap",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after zigzag overlapping access operations.",
 "code": "int zigzag[9] = {9, 18, 27, 36, 45, 54, 63, 72, 81};\nint *z_left = zigzag + 1;\nint *z_right = zigzag + 6;\nfor (int zag = 0; zag < 3; zag++) {\n if (zag % 2 == 0) {\n z_left[zag] = z_right[zag] - z_left[zag + 1];\n } else {\n z_right[zag - 1] = z_left[zag] + 9;\n }\n}\n// what is the value of zigzag[6] after the loop?",
 "answer": 45
 }
},
{
 "id": "PL-MEM-S006-V009",
 "metadata": {
 "name": "PropertyLevel-Memory-ConcentricOverlapPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "concentric_overlap",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Calculate the result of concentric overlapping access pattern.",
 "code": "int concentric[13] = {13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 143, 156, 169};\nint *core = concentric + 6;\nfor (int ring = 1; ring <= 4; ring++) {\n core[-ring] = core[ring] + core[0];\n core[ring] = core[-ring] - ring * 13;\n core[0] = (core[-ring] + core[ring]) / 2;\n}\n// what is the value of concentric[6] after the loop?",
 "answer": 91
 }
},
{
 "id": "PL-MEM-S006-V010",
 "metadata": {
 "name": "PropertyLevel-Memory-RecursiveOverlapPattern",
 "category": "Property-Level",
 "subcategory": "Memory",
 "type": "variant",
```

```

 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "recursive_overlap",
 "property_focus": "access_pattern",
 "memory_operation": "write"
 },
 "task": {
 "description": "Determine the value after recursive overlapping access operations.",
 "code": "int recursive[10] = {4, 8, 12, 16, 20, 24, 28, 32, 36, 40};\nint\n*rec_base = recursive + 3;\nfor (int depth = 0; depth < 3; depth++) {\n for (int span =\n1; span <= 2; span++) {\n rec_base[span] = rec_base[-span] + rec_base[span + 1];\n rec_base[-span] = rec_base[span] - span * 4;\n }\n}\n// What is the value of\nrecursive[2] after the loops?",
 "answer": 20
 }
}
]

```

## 3D - 作用域属性 [ Scope ] (59)

# 作用域属性推理变式生成提示词

## 任务目标

基于给定的作用域属性推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对作用域机制深层属性的理解能力，包括可见性规则、生存期管理和变量遮蔽等核心作用域属性推理技能。

## 作用域属性推理特征分析

作用域属性推理关注程序中变量和标识符的可见性、生存期和访问规则，重点测试模型对词法作用域、动态作用域、变量遮蔽和生存期管理的深层理解，强调对作用域语义的精确分析能力。

## 关键要求

- \*\*描述语言\*\*：所有task描述必须使用英文
- \*\*变式数量\*\*：每个种子任务严格生成8-12个变式，确保数量充足
- \*\*答案唯一性\*\*：保证答案准确且唯一，避免歧义和多解情况
- \*\*属性聚焦\*\*：重点关注作用域的属性分析，而非简单的执行结果

## 变式生成维度

### 1. 可见性规则分析变式

- \*\*词法作用域变式\*\*：静态作用域的变量可见性分析
- \*\*动态作用域变式\*\*：动态作用域的变量可见性分析
- \*\*函数作用域变式\*\*：函数内变量的可见性范围
- \*\*块作用域变式\*\*：块级作用域的变量可见性
- \*\*全局作用域变式\*\*：全局变量的可见性和访问
- \*\*模块作用域变式\*\*：模块级别的变量可见性

### 2. 生存期管理分析变式

- \*\*自动存储期变式\*\*：自动变量的生存期分析
- \*\*静态存储期变式\*\*：静态变量的生存期管理
- \*\*动态存储期变式\*\*：动态分配变量的生存期

- **临时对象生存期变式**: 临时对象的生存期分析
- **栈变量生存期变式**: 栈上变量的生存期管理
- **堆变量生存期变式**: 堆上变量的生存期控制

### ### 3. 变量遮蔽分析变式

- **同名变量遮蔽变式**: 同名变量在不同作用域的遮蔽
- **参数遮蔽变式**: 函数参数对外层变量的遮蔽
- **局部遮蔽全局变式**: 局部变量遮蔽全局变量
- **嵌套块遮蔽变式**: 嵌套块中的变量遮蔽
- **类成员遮蔽变式**: 类成员变量的遮蔽关系
- **多层遮蔽变式**: 多层嵌套的复杂遮蔽关系

### ### 4. 作用域链分析变式

- **作用域链查找变式**: 变量查找的作用域链分析
- **闭包作用域变式**: 闭包中的作用域链管理
- **嵌套函数作用域变式**: 嵌套函数的作用域链
- **this绑定作用域变式**: `this`关键字的作用域绑定
- **作用域链断裂变式**: 作用域链中断的情况
- **动态作用域链变式**: 动态改变的作用域链

### ### 5. 声明和定义分析变式

- **前向声明变式**: 前向声明的作用域影响
- **重复声明变式**: 同一作用域内的重复声明
- **声明顺序变式**: 声明顺序对作用域的影响
- **条件声明变式**: 条件语句中的变量声明
- **循环声明变式**: 循环中的变量声明和作用域
- **初始化时机变式**: 变量初始化的时机和作用域

### ### 6. 闭包和捕获分析变式

- **值捕获变式**: 闭包的值捕获机制分析
- **引用捕获变式**: 闭包的引用捕获机制分析
- **捕获列表变式**: 显式捕获列表的分析
- **自动捕获变式**: 自动捕获机制的分析
- **捕获时机变式**: 变量捕获的时机分析
- **捕获生存期变式**: 捕获变量的生存期管理

### ### 7. 名称空间和模块分析变式

- **名称空间作用域变式**: 名称空间的作用域规则
- **using声明变式**: `using`声明对作用域的影响
- **模块导入作用域变式**: 模块导入的作用域规则
- **别名作用域变式**: 别名对作用域的影响
- **限定名称变式**: 限定名称的作用域解析
- **名称冲突解决变式**: 名称冲突的解决机制

### ### 8. 特殊作用域规则变式

- **for循环作用域变式**: `for`循环变量的特殊作用域
- **switch作用域变式**: `switch`语句的作用域规则
- **异常处理作用域变式**: `try-catch`的作用域规则
- **lambda作用域变式**: `lambda`表达式的作用域规则
- **宏作用域变式**: 宏替换的作用域影响
- **模板作用域变式**: 模板实例化的作用域规则

### ### 9. 作用域优化分析变式

- **死代码消除变式**: 作用域分析中的死代码消除
- **变量提升变式**: 编译器的变量提升优化
- **寄存器分配变式**: 作用域对寄存器分配的影响
- **内联优化变式**: 函数内联对作用域的影响
- **常量折叠变式**: 常量折叠在作用域中的应用
- **循环优化变式**: 循环优化对作用域的影响

#### ### 10. 高级作用域属性变式

- **动态作用域链变式**: 运行时动态构建的作用域链
- **元编程作用域变式**: 元编程中的作用域管理
- **反射作用域变式**: 反射机制中的作用域访问
- **动态绑定变式**: 动态绑定的作用域规则
- **作用域污染变式**: 作用域污染的检测和分析
- **作用域安全变式**: 作用域相关的安全性分析

### ## 复杂度层次设计

#### ### 简单作用域属性 (Easy)

- 基础的块作用域和函数作用域
- 简单的变量遮蔽和可见性
- 直接的生存期管理
- 明确的作用域边界

#### ### 中等作用域属性 (Medium)

- 2-3层的嵌套作用域分析
- 静态变量和参数遮蔽
- 中等复杂的可见性规则
- 基础的闭包和捕获

#### ### 复杂作用域属性 (Hard)

- 复杂的多层嵌套作用域
- 复杂的变量遮蔽关系
- 高级的闭包和作用域链
- 特殊的作用域规则

#### ### 专家级作用域属性 (Expert)

- 极度复杂的作用域嵌套
- 动态作用域和元编程
- 编译器优化对作用域的影响
- 需要深度语言知识的作用域分析

### ## 生成策略

#### ### 种子分析策略

1. **识别核心属性**: 分析种子任务关注的主要作用域属性
2. **提取作用域模式**: 识别作用域规则和访问模式
3. **确定嵌套层次**: 分析作用域的嵌套结构和复杂度
4. **评估遮蔽关系**: 分析变量遮蔽的复杂程度

#### ### 变式设计原则

1. **属性导向**: 每个变式都应明确关注特定的作用域属性
2. **英文描述**: 所有task描述必须使用标准英文
3. **答案唯一**: 严格确保答案的准确性和唯一性

#### 4. \*\*数量保证\*\*: 严格确保每个种子生成8-12个变式

##### ### 质量保证

1. \*\*作用域语义验证\*\*: 验证作用域规则的语义正确性
2. \*\*可见性规则检查\*\*: 确保变量可见性分析的准确性
3. \*\*答案唯一性验证\*\*: 严格检查答案的唯一性和确定性
4. \*\*英文质量保证\*\*: 确保描述的英文表达准确清晰

##### ## 输出格式要求

```
```json
[
  {
    "id": "PL-SC-S00X-V001",
    "metadata": {
      "name": "PropertyLevel-Scope-VariantName",
      "category": "Property-Level",
      "subcategory": "Scope",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "property_focus": "visibility/lifetime/shadowing/closure/namespace",
      "scope_depth": "single/nested/deep_nested",
      "shadowing_complexity": "none/simple/complex"
    },
    "task": {
      "description": "English description of the scope property analysis task",
      "code": "Code containing scope-related constructs to analyze",
      "answer": "Unique and accurate analysis result or property value"
    }
  },
  {下一个变式...}
]
```
```

##### 特殊字段说明

**property\_focus**: 标识主要关注的作用域属性类型

**scope\_depth**: 标识作用域的嵌套深度

**shadowing\_complexity**: 标识变量遮蔽的复杂程度

##### 生成目标

为每个提供的种子任务严格生成8-12个作用域属性变式, 确保:

每种难度等级至少包含2-3个变式

涵盖至少6-8种不同的作用域属性分析类型

包含不同的作用域嵌套深度和复杂度

所有**task**描述使用标准英文表达

严格保证答案的准确性和唯一性

##### 重点测试场景

变量可见性分析: 准确分析变量在不同作用域的可见性

遮蔽关系理解: 正确理解复杂的变量遮蔽关系

生存期管理: 准确分析变量的生存期和销毁时机

作用域链追踪: 正确追踪变量查找的作用域链

闭包捕获分析：分析闭包中的变量捕获机制

特殊作用域规则：理解语言特定的作用域规则

英文描述模板示例

```
"Determine which variable is accessed in the given scope context..."
"Calculate the final value considering variable shadowing rules..."
"Analyze the variable lifetime in the nested scope structure..."
"Trace the scope chain to identify the accessed variable..."
"Evaluate the visibility of variables across different scopes..."
"Determine the final state after scope-based variable modifications..."
```

答案唯一性保证策略

明确作用域边界：清晰定义每个作用域的边界和规则

具体变量值：使用具体的变量值，避免符号化答案

确定性执行：确保代码执行路径是确定的

避免未定义行为：避免产生未定义行为的作用域使用

清晰语义：确保作用域语义在目标语言中是明确的

常见作用域模式

函数参数遮蔽：函数参数遮蔽同名全局变量

块级作用域：块内声明的变量遮蔽外层变量

静态变量持久：静态变量在函数调用间保持状态

闭包捕获：内部函数捕获外部作用域变量

循环变量作用域：循环控制变量的特殊作用域规则

条件声明：条件语句中变量的作用域范围

请基于此提示词，为给定的作用域属性推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

[

{

```
 "id": "PL-SC-S001-V001",
 "metadata": {
 "name": "PropertyLevel-Scope-NestedBlockShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "nested_block_visibility",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine which variable value is accessed in the deeply nested scope with multiple shadowing levels.",
 "code": "int x = 5;\nvoid nested_test() {\n int x = 15;\n {\n int x = 25;\n {\n int x = 35;\n printf(\"%d\\\", x);\n }\n }\n}\n// what value is printed by the printf statement?",
 "answer": 35
 },
 {
```

```

 "id": "PL-SC-S001-V002",
 "metadata": {
 "name": "PropertyLevel-Scope-ConditionalShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_scope",
 "property_focus": "visibility",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the variable value accessed in a conditional block with variable shadowing.",
 "code": "int y = 8;\nvoid conditional_scope(int condition) {\n int y = 16;\n if (condition > 0) {\n int y = 24;\n y = y + 6;\n printf(\"%d\\n\", y);\n }\n}\n// what value is printed when conditional_scope(1) is called?",
 "answer": 30
 }
},
{
 "id": "PL-SC-S001-V003",
 "metadata": {
 "name": "PropertyLevel-Scope-Loopvariableshadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_scope",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the variable value inside a for loop with local variable shadowing.",
 "code": "int i = 100;\nvoid loop_shadow() {\n int i = 200;\n for (int i = 0; i < 3; i++) {\n if (i == 2) {\n printf(\"%d\\n\", i);\n }
 }
}\n// what value is printed by the printf statement?",
 "answer": 2
 }
},
{
 "id": "PL-SC-S001-V004",
 "metadata": {
 "name": "PropertyLevel-Scope-FunctionParameterShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "function_parameter",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the variable value inside a function with parameter shadowing.",
 "code": "int i = 100;\nvoid function_shadow() {\n int i = 200;\n if (i == 2) {\n printf(\"%d\\n\", i);\n }
}\n// what value is printed by the printf statement?",
 "answer": 2
 }
}

```

```
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "parameter_shadowing",
 "property_focus": "shadowing",
 "scope_depth": "single",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the value accessed when function parameters shadow global variables.",
 "code": "int value = 50;\nint calculate_shadow(int value) {\n int\nvalue = 75;\n value *= 2;\n return value;\n}\n// What value is returned by calculate_shadow(10)?",
 "answer": 150
 }
},
{
 "id": "PL-SC-S001-V005",
 "metadata": {
 "name": "PropertyLevel-Scope-Multiplevariableshadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multiple_shadowing",
 "property_focus": "shadowing",
 "scope_depth": "deep_nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Determine which variables are accessed in a complex multi-variable shadowing scenario.",
 "code": "int a = 1, b = 2;\nvoid complex_shadow() {\n int a = 10;\n int b = 20;\n {\n int a = 100;\n printf(\"%d %d\", a,\n b);\n }\n}\n// What values are printed by the printf statement (format: a b)?",
 "answer": "100 20"
 }
},
{
 "id": "PL-SC-S001-V006",
 "metadata": {
 "name": "PropertyLevel-Scope-SwitchCaseShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
```

```
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "switch_scope",
 "property_focus": "visibility",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Calculate the variable value accessed within a switch case block with shadowing.",
 "code": "int num = 99;\nvoid switch_shadow(int choice) {\n int num = 88;\n switch (choice) {\n case 1: {\n int num = 77;\n num += 3;\n printf(\"%d\\n\", num);\n break;\n }\n }\n} // what value is printed when switch_shadow(1) is called?",
 "answer": 80
 }
},
{
 "id": "PL-SC-S001-V007",
 "metadata": {
 "name": "PropertyLevel-Scope-ArrayIndexShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "array_scope",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the index variable value in nested loops with variable shadowing.",
 "code": "int index = 0;\nvoid array_shadow() {\n int index = 5;\n for (int i = 0; i < 2; i++) {\n int index = 10;\n index += i;\n if (i == 1) {\n printf(\"%d\\n\", index);\n }\n }\n} // what value is printed by the printf statement?",
 "answer": 11
 }
},
{
 "id": "PL-SC-S001-V008",
 "metadata": {
 "name": "PropertyLevel-Scope-StructMembersShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
```

```
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "struct_scope",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the variable value when local variables shadow struct member names.",
 "code": "struct Data { int value; };\nstruct Data data = {42};\nvoid\nstruct_shadow()\n{\n int value = 84;\n {\n int value = 126;\n value =\n value / 2;\n printf(\"%d\\\", value);\n }\n}\n// What value is printed by the printf statement?",
 "answer": 63
 }
},
{
 "id": "PL-SC-S001-V009",
 "metadata": {
 "name": "PropertyLevel-Scope-Macroshadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "macro_scope",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine which value is accessed when local variables shadow macro-defined constants.",
 "code": "#define MAX_VAL 999\nvoid macro_shadow()\n{\n int MAX_VAL = 111;\n int MAX_VAL = 222;\n MAX_VAL += 8;\n printf(\"%d\\\", MAX_VAL);\n}\n\n// What value is printed by the printf statement?",
 "answer": 230
 }
},
{
 "id": "PL-SC-S001-V010",
 "metadata": {
 "name": "PropertyLevel-Scope-TypedefShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
```

```
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "typedef_scope",
 "property_focus": "shadowing",
 "scope_depth": "deep_nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the final value when variables shadow typedef names in nested scopes.",
 "code": "typedef int MyInt;\nMyInt num = 300;\nvoid typedef_shadow() {\n int MyInt = 400;\n {\n int MyInt = 500;\n {\n int MyInt = 600;\n MyInt -= 50;\n printf(\"%d\\n\", MyInt);\n }\n }\n}\n// What value is printed by the printf statement?",
 "answer": 550
 }
},
{
 "id": "PL-SC-S002-V001",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticCounterIncrement",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "static_persistence",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the static variable value after multiple function invocations with different increments.",
 "code": "int increment_counter(int step) {\n static int counter = 10;\n counter += step;\n return counter;\n}\n// What value is returned by the fourth call: increment_counter(3), increment_counter(2), increment_counter(5), increment_counter(1)?",
 "answer": 21
 }
},
{
 "id": "PL-SC-S002-V002",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticArrayAccumulation",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 }
}
```

```
 "variant_type": "static_array",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Determine the static array element value after multiple function calls with array modifications.",
 "code": "int modify_array(int index, int value) {\n static int arr[3] = {1,\n2, 3};\n arr[index] += value;\n return arr[index];\n}\n// what is the value of arr[1] after: modify_array(1, 4), modify_array(1, 3), modify_array(1, 2)?",
 "answer": 11
 }
},
{
 "id": "PL-SC-S002-V003",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticStructMember",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "static_struct",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the static struct member value after multiple function calls with struct modifications.",
 "code": "typedef struct { int x, y; } Point;\nPoint* get_static_point(int dx,\nint dy) {\n static Point p = {0, 0};\n p.x += dx;\n p.y += dy;\n return\n &p;\n}\n// what is the x value after: get_static_point(3, 2), get_static_point(1, 4),\nget_static_point(2, 1)?",
 "answer": 6
 }
},
{
 "id": "PL-SC-S002-V004",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticConditionalUpdate",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "static_conditional",
 "property_focus": "lifetime",
```

```
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Determine the static variable value after conditional updates across multiple calls.",
 "code": "int conditional_update(int flag) {\n static int value = 20;\n if (flag > 0) {\n value *= 2;\n } else {\n value += 5;\n }\n return value;\n}\n// what value is returned by the third call: conditional_update(1), conditional_update(0), conditional_update(1)?",
 "answer": 90
 }
},
{
 "id": "PL-SC-S002-V005",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticRecursiveAccumulation",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "static_recursive",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the static variable value in a recursive function after complete execution.",
 "code": "int recursive_static(int n) {\n static int sum = 0;\n if (n > 0)\n sum += n;\n return recursive_static(n - 1);\n}\nreturn sum;\n// what value is returned by recursive_static(4)?",
 "answer": 10
 }
},
{
 "id": "PL-SC-S002-V006",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticMultipleVariables",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "static_multiple",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 }
}
```

```
 },
 "task": {
 "description": "Determine the sum of multiple static variables after several function calls.",
 "code": "int updateStatics(int a, int b) {\n static int x = 1, y = 2;\n x += a;\n y += b;\n return x + y;\n}\n// What value is returned by the second call: updateStatics(3, 4), updateStatics(2, 1)?",
 "answer": 13
 }
 },
 {
 "id": "PL-SC-S002-V007",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticPointerManagement",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "static_pointer",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the value pointed to by a static pointer after multiple function calls.",
 "code": "int* get_static_ptr(int increment) {\n static int value = 100;\n static int* ptr = &value;\n *ptr += increment;\n return ptr;\n}\n// What value is pointed to after: get_static_ptr(5), get_static_ptr(10), get_static_ptr(7)?",
 "answer": 122
 }
 },
 {
 "id": "PL-SC-S002-V008",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticBitwiseOperations",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "static_bitwise",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
```

```
 "description": "Determine the static variable value after bitwise operations across multiple calls.",
 "code": "int bitwise_static(int operation) {\n static int bits = 15; // 1111\n if (operation == 1) {\n bits <= 1;\n } else {\n bits >= 1;\n }\n return bits;\n} // What value is returned by the third call: bitwise_static(1), bitwise_static(0), bitwise_static(1)?",
 "answer": 30
 }
},
{
 "id": "PL-SC-S002-V009",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticFloatingPoint",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "static_float",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the static floating-point variable after multiple arithmetic operations.",
 "code": "int get_rounded_static(double multiplier) {\n static double value = 2.5;\n value *= multiplier;\n return (int)value;\n} // What integer value is returned by the second call: get_rounded_static(1.5), get_rounded_static(2.0)?",
 "answer": 7
 }
},
{
 "id": "PL-SC-S002-V010",
 "metadata": {
 "name": "PropertyLevel-Scope-StaticModuloCounter",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "static_modulo",
 "property_focus": "lifetime",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Determine the static counter value with modulo operations after multiple calls."
 }
}
```

```
 "code": "int modulo_counter(int increment) {\n static int counter = 8;\n counter = (counter + increment) % 10;\n return counter;\n}\n// What value is returned by the fourth call: modulo_counter(3), modulo_counter(4), modulo_counter(6), modulo_counter(2)?",
 "answer": 3
 },
},
{
 "id": "PL-SC-S003-V001",
 "metadata": {
 "name": "PropertyLevel-Scope-BlockExitRestoration",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "block_exit",
 "property_focus": "visibility",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Calculate the outer variable value after nested block modifications and exit.",
 "code": "int outer = 12;\n{\n int outer = 24;\n outer = outer + 6;\n\n int outer = 48;\n outer *= 2;\n}\n\nouter = outer - 2;\n// What is the final value of the outermost 'outer' variable?",
 "answer": 10
 }
},
{
 "id": "PL-SC-S003-V002",
 "metadata": {
 "name": "PropertyLevel-Scope-ConditionalBlockLifetime",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "conditional_block",
 "property_focus": "lifetime",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the variable value after conditional block execution and exit."
 }
}
```

```
 "code": "int count = 7;\nint condition = 1;\nif (condition) {\n int count =\n14;\n count += 3;\n if (count > 15) {\n int count = 28;\n count /= 2;\n }\n}\n\n// what is the final value of the outer 'count' variable?",\n "answer": 21\n }\n},\n{\n "id": "PL-SC-S003-V003",\n "metadata": {\n "name": "PropertyLevel-Scope-LoopBlockLifetime",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "loop_block",\n "property_focus": "lifetime",\n "scope_depth": "nested",\n "shadowing_complexity": "simple"\n },\n "task": {\n "description": "Calculate the outer variable value after loop block execution\nwith local variable destruction.",\n "code": "int sum = 4;\nfor (int i = 0; i < 3; i++) {\n int sum = 8;\n sum\n+= i;\n if (i == 2) {\n int sum = 16;\n sum *= 2;\n }\n}\n\nsum += 6;\n\n//\n\n// what is the final value of the outer 'sum' variable?",\n "answer": 10\n }\n},\n{\n "id": "PL-SC-S003-V004",\n "metadata": {\n "name": "PropertyLevel-Scope-SwitchBlockLifetime",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "switch_block",\n "property_focus": "lifetime",\n "scope_depth": "nested",\n "shadowing_complexity": "simple"\n },\n "task": {\n "description": "Determine the variable value after switch case block execution\nand exit.",\n "code": "int sum = 4;\nswitch (sum) {\n case 1:\n sum = 5;\n case 2:\n sum = 6;\n case 3:\n sum = 7;\n default:\n sum = 8;\n}\n\n//\n\n// what is the final value of the outer 'sum' variable?",\n "answer": 8\n }\n}
```



```

 "code": "int base = 15;\nint flag = 0;\ndo {\n int base = 30;\n base +=\n5;\n flag = 1;\n} while (flag == 0);\nbase -= 3;\n// What is the final value of the outer\n'base' variable?",\n "answer": 12\n },\n},\n{\n "id": "PL-SC-S003-V007",\n "metadata": {\n "name": "PropertyLevel-Scope-NestedBlockChain",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "nested_chain",\n "property_focus": "visibility",\n "scope_depth": "deep_nested",\n "shadowing_complexity": "complex"\n },\n "task": {\n "description": "Calculate the variable value after multiple nested block\nexecutions with sequential modifications.",\n "code": "int chain = 3;\n{\n int chain = 6;\n chain += 2;\n {\n int chain = 12;\n chain *= 2;\n {\n int chain = 24;\n chain /= 3;\n }\n }\n}\nchain += 7;\n// What is the final value of the outermost\n'chain' variable?",\n "answer": 10\n },\n},\n{\n "id": "PL-SC-S003-V008",\n "metadata": {\n "name": "PropertyLevel-Scope-ArrayBlockLifetime",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "array_block",\n "property_focus": "lifetime",\n "scope_depth": "nested",\n "shadowing_complexity": "simple"\n },\n "task": {\n "description": "Determine the array element value after block-scoped array\nmodifications and destruction.",\n }\n}
```

```
 "code": "int arr[3] = {1, 2, 3};\n{ int arr[3] = {4, 5, 6};\n arr[1] +=\n10;\n {\n int arr[3] = {7, 8, 9};\n arr[1] *= 2;\n }\n}narr[1] +=\n5;\n// what is the final value of the outer arr[1]?",\n "answer": 7\n }\n},\n{\n "id": "PL-SC-S003-V009",\n "metadata": {\n "name": "PropertyLevel-Scope-PointerBlockLifetime",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "pointer_block",\n "property_focus": "lifetime",\n "scope_depth": "nested",\n "shadowing_complexity": "complex"\n },\n "task": {\n "description": "Calculate the pointer value after block-scoped pointer manipulations and restoration.",\n "code": "int original = 20;\nint* ptr = &original;\n\n int local = 40;\nint* ptr = &local;\n *ptr += 8;\n {\n int inner = 80;\n int* ptr = &inner;\n *ptr /= 2;\n }\n*ptr += 10;\n// what is the final value of the variable pointed to by the outer ptr?",\n "answer": 30\n }\n},\n{\n "id": "PL-SC-S003-V010",\n "metadata": {\n "name": "PropertyLevel-Scope-StructBlockLifetime",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "struct_block",\n "property_focus": "lifetime",\n "scope_depth": "nested",\n "shadowing_complexity": "simple"\n },\n "task": {\n "description": "Determine the struct member value after block-scoped struct modifications and destruction.",\n "code": "struct S {\n int a;\n int b;\n};\n\nS s;\ns.a = 10;\ns.b = 20;\n\n{\n S s;\n s.a = 30;\n s.b = 40;\n}\ns.a = 50;\ns.b = 60;\n\n// what is the final value of the outer s.a and s.b?"\n "answer": 50\n }\n}
```

```
 "code": "struct Point { int x, y; };\\nstruct Point p = {5, 10};\\n{\n struct Point p = {15, 20};\\n p.x += 5;\\n {\n struct Point p = {25, 30};\\n p.x\n *= 2;\\n }\\n}np.x += 3;\\n// what is the final value of the outer p.x?",\n "answer": 8\n }\n},\n{\n "id": "PL-SC-S004-V001",\n "metadata": {\n "name": "PropertyLevel-Scope-GlobalArrayModification",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "medium",\n "intervention": 1,\n "variant_type": "global_array",\n "property_focus": "visibility",\n "scope_depth": "single",\n "shadowing_complexity": "none"\n },\n "task": {\n "description": "Calculate the global array element value after multiple function calls with array modifications.",\n "code": "int global_array[3] = {2, 4, 6};\\nvoid modify_element(int index, int value) {\n global_array[index] += value;\\n}\\nvoid process_array() {\n modify_element(1, 3);\\n modify_element(1, 2);\\n modify_element(1, -1);\\n}\\n// What is the value of global_array[1] after calling process_array()?",\n "answer": 8\n }\n},\n{\n "id": "PL-SC-S004-V002",\n "metadata": {\n "name": "PropertyLevel-Scope-GlobalStructModification",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "global_struct",\n "property_focus": "visibility",\n "scope_depth": "single",\n "shadowing_complexity": "none"\n },\n "task": {\n "description": "Determine the global struct member value after function calls with struct modifications.",\n "code": "struct Point {\n int x, y;\n};\\nstruct Point p = {\n x: 5,\n y: 10\n};\\n{\n struct Point p = {\n x: 15,\n y: 20\n };\\n p.x += 5;\\n {\n struct Point p = {\n x: 25,\n y: 30\n };\\n p.x\n *= 2;\\n }\\n}np.x += 3;\\n// What is the final value of the outer p.x?",\n "answer": 8\n }\n}
```

```
 "code": "struct Counter { int value; int step; };\\nstruct Counter global_counter\n= {10, 2};\\nvoid update_counter(int multiplier) {\\n global_counter.value +=\nglobal_counter.step * multiplier;\\n}\\nvoid execute_updates() {\\n update_counter(3);\\n update_counter(2);\\n update_counter(1);\\n}\\n// what is the value of global_counter.value\n// after calling execute_updates()?",\n "answer": 22\n }\n},\n{\n "id": "PL-SC-S004-V003",\n "metadata": {\n "name": "PropertyLevel-Scope-GlobalPointerChain",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "global_pointer",\n "property_focus": "visibility",\n "scope_depth": "single",\n "shadowing_complexity": "none"\n },\n "task": {\n "description": "Calculate the value pointed to by global pointer after function\nchain modifications.",\n "code": "int target = 50;\\nint* global_ptr = ⌖\\nvoid\nmodify_through_ptr(int delta) {\\n *global_ptr += delta;\\n}\\nvoid chain_modifications()\n{\\n modify_through_ptr(8);\\n modify_through_ptr(-3);\\n modify_through_ptr(5);\\n modify_through_ptr(-2);\\n}\\n// what is the value pointed to by global_ptr after calling\nchain_modifications()?",\n "answer": 58\n }\n},\n{\n "id": "PL-SC-S004-V004",\n "metadata": {\n "name": "PropertyLevel-Scope-GlobalVariablewithRecursion",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "global_recursive",\n "property_focus": "visibility",\n "scope_depth": "single",\n "shadowing_complexity": "none"\n },\n "task": {
```

```

 "description": "Determine the global variable value after recursive function calls with accumulation.",
 "code": "int global_sum = 1;\nvoid recursive_accumulate(int n) {\n if (n > 0)\n global_sum *= 2;\n recursive_accumulate(n - 1);\n }\n}\nvoid start_recursion() {\n recursive_accumulate(3);\n}\n// What is the value of global_sum after calling start_recursion()?",
 "answer": 8
 }
},
{
 "id": "PL-SC-S004-V005",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalBitwiseOperations",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "global_bitwise",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the global variable value after function calls with bitwise operations.",
 "code": "int global_bits = 12; // 1100 in binary\nvoid shift_bits(int direction)\n{\n if (direction > 0) {\n global_bits <= 1;\n } else {\n global_bits >>= 1;\n }\n}\nvoid bit_sequence()\n{\n shift_bits(1);\n shift_bits(1);\n shift_bits(-1);\n}\n// What is the value of global_bits after calling bit_sequence()?",
 "answer": 24
 }
},
{
 "id": "PL-SC-S004-V006",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalFloatingPoint",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "global_float",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {

```

```
 "description": "Determine the integer value of global floating-point variable after function modifications.",
 "code": "double global_float = 3.2;\nvoid multiply_float(double factor) {\n global_float *= factor;\n}\nvoid calculate_sequence() {\n multiply_float(2.5);\n multiply_float(1.25);\n} // What is the integer value of global_float after calling calculate_sequence()?",
 "answer": 10
 },
},
{
 "id": "PL-SC-S004-V007",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalStringLength",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "global_string",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the global string length after function calls with string modifications.",
 "code": "#include <string.h>\nchar global_str[20] = \"hello\";\nvoid append_char(char c) {\n int len = strlen(global_str);\n global_str[len] = c;\n global_str[len + 1] = '\\0';\n}\nvoid build_string() {\n append_char('w');\n append_char('o');\n append_char('r');\n} // What is the length of global_str after calling build_string()?",
 "answer": 8
 }
},
{
 "id": "PL-SC-S004-V008",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalModuloCounter",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "global_modulo",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {

```

```

 "description": "Determine the global counter value with modulo operations after function calls.",
 "code": "int global_counter = 7;\nvoid increment_modulo(int step, int modulus)\n{\n global_counter = (global_counter + step) % modulus;\n}\nvoid counter_sequence()\n{\n increment_modulo(4, 10);\n increment_modulo(6, 10);\n increment_modulo(8, 10);\n}\n// what is the value of global_counter after calling counter_sequence()?",
 "answer": 5
 },
 {
 "id": "PL-SC-S004-V009",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalConditionalUpdate",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "global_conditional",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 },
 "task": {
 "description": "Calculate the global variable value after conditional function calls with different operations.",
 "code": "int global_value = 16;\nvoid conditional_modify(int condition, int operand)\n{\n if (condition == 1) {\n global_value += operand;\n } else if (condition == 2) {\n global_value -= operand;\n } else {\n global_value *= operand;\n }\n}\nvoid execute_conditions()\n{\n conditional_modify(1, 4);\n conditional_modify(2, 6);\n conditional_modify(3, 2);\n}\n// what is the value of global_value after calling execute_conditions()?",
 "answer": 28
 }
 },
 {
 "id": "PL-SC-S004-V010",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalArraySum",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "global_array_sum",
 "property_focus": "visibility",
 "scope_depth": "single",
 "shadowing_complexity": "none"
 }
 }
]
```

```

"task": {
 "description": "Determine the sum of global array elements after function calls with array manipulations.",
 "code": "int global_nums[4] = {1, 2, 3, 4};\nvoid rotate_and_add(int add_value)\n{\n int temp = global_nums[0];\n for (int i = 0; i < 3; i++) {\n global_nums[i] = global_nums[i + 1];\n }\n global_nums[3] = temp + add_value;\n}\nvoid process_array()\n{\n rotate_and_add(2);\n rotate_and_add(1);\n}\n// what is the sum of all elements in global_nums after calling process_array()?",
 "answer": 15
},
{
 "id": "PL-SC-S005-V001",
 "metadata": {
 "name": "PropertyLevel-Scope-MultiParameterShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "multi_parameter",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the return value when multiple parameters shadow global variables in nested scopes.",
 "code": "int a = 1, b = 2, c = 3;\nint complex_shadow(int a, int b) {\n int c = a + b;\n int a = 10;\n c += a;\n return c;\n}\n// what value is returned by complex_shadow(4, 5)?",
 "answer": 19
 }
},
{
 "id": "PL-SC-S005-V002",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterArrayShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_array",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {

```

```
 "description": "Determine the return value when parameter arrays shadow global arrays.",
 "code": "int arr[3] = {1, 2, 3};\nint array_shadow(int arr[]) {\n arr[1] = 20;\n int arr[3] = {10, 11, 12};\n arr[1] += 5;\n return arr[1];\n}\nint input[3] = {7, 8, 9};\n// What value is returned by array_shadow(input)?",
 "answer": 16
 },
},
{
 "id": "PL-SC-S005-V003",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterPointersShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_pointer",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the final value when pointer parameters shadow global pointer variables.",
 "code": "int value = 100;\nint* ptr = &value;\nint pointer_shadow(int* ptr) {\n *ptr = 200;\n int local = 300;\n int* ptr = &local;\n *ptr += 50;\n return *ptr;\n}\nint target = 150;\n// What value is returned by pointer_shadow(&target)?",
 "answer": 350
 },
},
{
 "id": "PL-SC-S005-V004",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterStructShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "parameter_struct",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {

```

```
 "description": "Determine the return value when struct parameters shadow global struct variables.",
 "code": "struct Point { int x, y; };\nstruct Point point = {5, 10};\nint struct_shadow(struct Point point) {\n point.x = 15;\n {\n struct Point point =\n{25, 30};\n point.x += point.y;\n return point.x;\n }\n}\nstruct Point\ninput = {8, 12};\n// what value is returned by struct_shadow(input)?",
 "answer": 55
 }
},
{
 "id": "PL-SC-S005-V005",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterFunctionPointer",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_function_ptr",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the return value when function pointer parameters are used in shadowing contexts.",
 "code": "int global_func() { return 10; }\nint local_func() { return 20; }\nint func_ptr_shadow(int (*func)()) {\n int result = func();\n {\n int (*func)() =\nlocal_func;\n result += func();\n }\n return result;\n}\n// what value is returned by func_ptr_shadow(global_func)?",
 "answer": 30
 }
},
{
 "id": "PL-SC-S005-V006",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterConstShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "parameter_const",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
```

```

 "description": "Determine the return value when const parameters shadow global
const variables.",
 "code": "const int MAX = 100;\nint const_shadow(const int MAX) {\n int result
= MAX * 2;\n {\n const int MAX = 50;\n result += MAX * 3;\n }\n return result;\n}\n// What value is returned by const_shadow(25)?",
 "answer": 200
 },
 {
 "id": "PL-SC-S005-V007",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterVolatileShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_volatile",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the return value when volatile parameters shadow
global variables.",
 "code": "volatile int flag = 1;\nint volatile_shadow(volatile int flag) {\n int total = flag * 10;\n {\n volatile int flag = 5;\n total += flag * 6;\n int flag = 8;\n total += flag * 2;\n }\n return total;\n}\n// What value is returned by volatile_shadow(3)?",
 "answer": 76
 }
 },
 {
 "id": "PL-SC-S005-V008",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterRecursiveShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_recursive",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Determine the return value when parameters shadow variables in
recursive function calls."
 }
 }
]
```

```
 "code": "int count = 0;\nint recursive_shadow(int count) {\n if (count <= 0)\n return 1;\n int count = 2;\n return count + recursive_shadow(count - 1);\n}\n// what value is returned by recursive_shadow(1)?",
 "answer": 3
 }
},
{
 "id": "PL-SC-S005-V009",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterUnionShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "parameter_union",
 "property_focus": "shadowing",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the return value when union parameters shadow global union variables.",
 "code": "union Data { int i; float f; };\nunion Data global_data = {42};\nint union_shadow(union Data data) {\n data.i = 84;\n {\n union Data data = {168};\n data.i /= 2;\n return data.i;\n }\n}\nunion Data input = {21};\n// what value is returned by union_shadow(input)?",
 "answer": 84
 }
},
{
 "id": "PL-SC-S005-V010",
 "metadata": {
 "name": "PropertyLevel-Scope-ParameterTypedefShadowing",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "parameter_typedef",
 "property_focus": "shadowing",
 "scope_depth": "deep_nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Determine the return value when typedef parameters create complex shadowing scenarios."
 }
}
```

```

 "code": "typedef int MyType;\nMyType global_value = 10;\nint\ntypedef_shadow(MyType value) {\n value *= 3;\n {\n typedef float MyType;\n MyType value = 2.5;\n int result = (int)(value * 4);\n {\n int\n value = 15;\n result += value;\n }\n return result;\n }\n}\n\nwhat value is returned by typedef_shadow(7)?",
 "answer": 25
}
},
{
 "id": "PL-SC-S006-V001",
 "metadata": {
 "name": "PropertyLevel-Scope-NestedFunctionCapture",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "nested_capture",
 "property_focus": "closure",
 "scope_depth": "deep_nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the final value considering multiple nested function scopes with variable capture.",
 "code": "def triple_nested(x):\n def level_one():\n def level_two():\n nonlocal x\n x *= 3\n return x\n x += 5\n return level_two()\n x -= 2\n return result\n return level_one()\n\nwhat value is returned by triple_nested(4)?",
 "answer": 27
 }
},
{
 "id": "PL-SC-S006-V002",
 "metadata": {
 "name": "PropertyLevel-Scope-ClosurewithLoop",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "closure_loop",
 "property_focus": "closure",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the value returned by closure created within a loop with variable capture."
 }
}

```

```
"code": "def closure_loop(n):\n result = 0\n for i in range(3):\n nonlocal result\n result += n * i\n return result\n\n if i == 2:\n return inner()\n return result\n\n# What value is returned by closure_loop(5)?",
 "answer": 10
}
},
{
 "id": "PL-SC-S006-V003",
 "metadata": {
 "name": "PropertyLevel-Scope-GlobalNonlocalInteraction",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "global_nonlocal",
 "property_focus": "visibility",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the final value with both global and nonlocal variable declarations.",
 "code": "value = 100\ndef global_nonlocal_test(x):\n value = 200\n def inner():\n global value\n value = 300\n def deep_inner():\n nonlocal value\n value += x\n return value\n return deep_inner()\n return inner()\n\n# What value is returned by global_nonlocal_test(50)?",
 "answer": 250
 }
},
{
 "id": "PL-SC-S006-V004",
 "metadata": {
 "name": "PropertyLevel-Scope-ClosureListComprehension",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "closure_comprehension",
 "property_focus": "closure",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
 "description": "Determine the value when closures interact with list comprehension scoping rules."
 }
}
```

```
 "code": "def comprehension_closure(multiplier):\n base = 10\n def\nmake_func(x):\n def inner():\n nonlocal base\n base += x *\nmultiplier\n return base\n return inner\n funcs = [make_func(i) for i\nin range(3)]\n return funcs[2]()\n\n# What value is returned by\ncomprehension_closure(3)?",
 "answer": 16
 },
{
 "id": "PL-SC-S006-V005",
 "metadata": {
 "name": "PropertyLevel-Scope-ExceptionHandlingScope",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "exception_scope",
 "property_focus": "visibility",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the final value when nested functions interact with\nexception handling scopes.",
 "code": "def exception_scope(x):\n result = x\n def risky_operation():\n nonlocal result\n try:\n result *= 2\n if result > 15:\n raise ValueError()\n result += 3\n except ValueError:\n result -= 5\n return result\n\n# What value is\nreturned by exception_scope(8)?",
 "answer": 11
 },
{
 "id": "PL-SC-S006-V006",
 "metadata": {
 "name": "PropertyLevel-Scope-ClosureFactoryPattern",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "closure_factory",
 "property_focus": "closure",
 "scope_depth": "nested",
 "shadowing_complexity": "simple"
 },
 "task": {
```

```
 "description": "Determine the value from closure factory with multiple function\ncreations.",\n "code": "def closure_factory(initial):\n def create_counter(step):\n count = initial\n def increment():\n nonlocal count\n count += step\n return count\n return increment\n counter1 = create_counter(5)\n counter2 = create_counter(3)\n return counter1() + counter2()\n\n # What value is returned by closure_factory(10)?",\n "answer": 28\n }\n },\n {\n "id": "PL-SC-S006-V007",\n "metadata": {\n "name": "PropertyLevel-Scope-LambdaClosure",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "python",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "lambda_closure",\n "property_focus": "closure",\n "scope_depth": "nested",\n "shadowing_complexity": "complex"\n },\n "task": {\n "description": "Calculate the value when lambda functions create closures with\nvariable capture.",\n "code": "def lambda_closure(x):\n multiplier = 2\n def outer():\n nonlocal multiplier\n multiplier += 1\n inner = lambda: x * multiplier\n return inner()\n result = outer()\n multiplier += 2\n second = lambda: x *\n multiplier\n return result + second()\n\n # What value is returned by lambda_closure(4)?",\n "answer": 32\n }\n },\n {\n "id": "PL-SC-S006-V008",\n "metadata": {\n "name": "PropertyLevel-Scope-DecoratorScope",\n "category": "Property-Level",\n "subcategory": "Scope",\n "type": "variant",\n "source": "Generated",\n "language": "python",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "decorator_scope",\n "property_focus": "closure",\n "scope_depth": "deep_nested",\n "shadowing_complexity": "complex"\n },\n "task": {\n
```

```
 "description": "Determine the value when decorator functions create complex closure relationships.",
 "code": "def decorator_scope(base):\n def decorator(func):\n def wrapper(x):\n nonlocal base\n base *= 2\n return result\n result = func(base)\n def calculate(value):\n return value + 10\n calculate(5)\n # What value is returned by decorator_scope(3)?",
 "answer": 18
 },
 {
 "id": "PL-SC-S006-V009",
 "metadata": {
 "name": "PropertyLevel-Scope-GeneratorScope",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "generator_scope",
 "property_focus": "closure",
 "scope_depth": "nested",
 "shadowing_complexity": "complex"
 },
 "task": {
 "description": "Calculate the value when generator functions interact with closure scoping.",
 "code": "def generator_scope(start):\n accumulator = start\n def create_generator():\n def gen():\n nonlocal accumulator\n for i in range(3):\n accumulator += i * 2\n yield accumulator\n return gen()\n g = create_generator()\n next(g)\n next(g)\n return next(g)\n # What value is returned by generator_scope(5)?",
 "answer": 15
 },
 },
 {
 "id": "PL-SC-S006-V010",
 "metadata": {
 "name": "PropertyLevel-Scope-ClassMethodClosure",
 "category": "Property-Level",
 "subcategory": "Scope",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "class_method_closure",
 "property_focus": "closure",
 "scope_depth": "deep_nested",
 "shadowing_complexity": "complex"
 },
 },
```

```

"task": {
 "description": "Determine the value when class methods create closures with
instance variable interaction.",
 "code": "class ScopeTest:\n def __init__(self, value):\n self.value = value\n def create_closure(self, multiplier):\n def inner():\n self.value *= multiplier\n return self.value + 7\n self.value += 3\n return inner()\n obj = ScopeTest(6)\n # What value is returned by obj.create_closure(2)?",
 "answer": 25
}
]

```

## 3E - 大混合 (51)

# 代码属性推理大混合变式生成提示词

## 任务目标

基于提供的循环属性、分支属性、内存属性、作用域属性种子任务和大混合示例，生成50个左右融合多种代码属性推理类型的综合变式任务，全面测试大模型对循环、分支、内存、作用域等多重属性的综合理解和交叉应用能力。

## 代码属性推理大混合特征分析

代码属性推理大混合关注程序中多种属性机制的交互和综合效应，重点测试模型对循环计数与分支选择、内存访问与作用域管理、变量追踪与控制流等复杂组合的深层理解，强调对多维度属性的综合分析能力。

## 关键要求

- \*\*描述语言\*\*：所有task描述必须使用英文
- \*\*变式数量\*\*：总计生成约50个混合变式任务
- \*\*答案唯一性\*\*：保证答案准确且唯一，避免歧义和多解情况
- \*\*混合复杂度\*\*：每个变式都应融合至少2-3种属性类型

## 混合变式生成维度

### 1. 循环+分支混合变式

- \*\*条件循环计数变式\*\*：带有条件分支的循环迭代计数分析
- \*\*分支内循环变式\*\*：分支内部循环的变量追踪和计数
- \*\*多出口循环变式\*\*：具有多个分支出口的循环终止分析
- \*\*嵌套条件循环变式\*\*：嵌套条件判断的复杂循环结构
- \*\*循环分支累积变式\*\*：循环中分支选择的累积效果分析
- \*\*短路循环变式\*\*：短路求值影响循环执行的分析

### 2. 循环+内存混合变式

- \*\*指针循环遍历变式\*\*：指针在循环中的算术操作和内存访问
- \*\*循环内存修改变式\*\*：循环中对内存的修改和累积效果
- \*\*数组循环处理变式\*\*：数组处理循环的内存访问模式分析
- \*\*指针追踪循环变式\*\*：循环中指针状态的追踪和变化
- \*\*内存拷贝循环变式\*\*：循环中的内存拷贝和数据移动
- \*\*重叠访问循环变式\*\*：循环中重叠内存访问的分析

### 3. 循环+作用域混合变式

- \*\*循环作用域变量变式\*\*：循环中不同作用域变量的生存期分析
- \*\*静态变量循环变式\*\*：静态变量在循环中的状态管理
- \*\*循环局部遮蔽变式\*\*：循环中局部变量遮蔽的分析

- \*\*嵌套作用域循环变式\*\*：嵌套作用域中的循环变量管理
- \*\*循环闭包捕获变式\*\*：循环中闭包变量捕获的分析
- \*\*循环变量生存期变式\*\*：循环变量的生存期和可见性分析

#### #### 4. 分支+内存混合变式

- \*\*条件内存访问变式\*\*：条件分支中的指针操作和内存访问
- \*\*分支指针修改变式\*\*：不同分支路径的指针修改效果
- \*\*条件内存分配变式\*\*：条件分支中的内存分配和释放
- \*\*分支数组操作变式\*\*：分支中数组操作的内存效果分析
- \*\*多路径内存变式\*\*：多个分支路径的内存状态分析
- \*\*条件指针算术变式\*\*：条件控制的指针算术操作

#### #### 5. 分支+作用域混合变式

- \*\*条件作用域变式\*\*：条件分支中的作用域变量管理
- \*\*分支变量遮蔽变式\*\*：分支中变量遮蔽的复杂情况
- \*\*多路径作用域变式\*\*：多个分支路径的作用域分析
- \*\*条件静态变量变式\*\*：条件控制的静态变量修改
- \*\*分支闭包变式\*\*：分支中闭包的作用域分析
- \*\*嵌套分支作用域变式\*\*：嵌套分支的作用域管理

#### #### 6. 内存+作用域混合变式

- \*\*指针作用域变式\*\*：指针在不同作用域中的生存期分析
- \*\*内存遮蔽变式\*\*：内存访问中的变量遮蔽效果
- \*\*作用域内存管理变式\*\*：不同作用域的内存分配和释放
- \*\*指针生存期变式\*\*：指针变量的生存期和有效性分析
- \*\*局部指针变式\*\*：局部作用域中的指针操作分析
- \*\*全局内存访问变式\*\*：全局变量的内存访问模式

#### #### 7. 三重混合变式 (循环+分支+内存)

- \*\*条件循环内存变式\*\*：条件循环中的内存操作综合分析
- \*\*循环分支指针变式\*\*：循环中分支控制的指针操作
- \*\*内存条件循环变式\*\*：基于内存状态的条件循环
- \*\*指针循环分支变式\*\*：指针操作驱动的循环分支逻辑
- \*\*数组处理综合变式\*\*：数组处理中的循环、条件和内存
- \*\*缓冲区处理变式\*\*：缓冲区处理的综合属性分析

#### #### 8. 三重混合变式 (循环+分支+作用域)

- \*\*循环条件作用域变式\*\*：循环中条件控制的作用域管理
- \*\*分支循环遮蔽变式\*\*：分支和循环中的变量遮蔽
- \*\*作用域循环条件变式\*\*：作用域变量影响的循环条件
- \*\*嵌套结构综合变式\*\*：嵌套循环、分支和作用域的综合
- \*\*静态变量复合变式\*\*：静态变量在复合结构中的管理
- \*\*闭包循环分支变式\*\*：闭包中的循环和分支综合

#### #### 9. 三重混合变式 (循环+内存+作用域)

- \*\*循环内存作用域变式\*\*：循环中内存操作的作用域分析
- \*\*指针循环生存期变式\*\*：指针在循环中的生存期管理
- \*\*作用域内存循环变式\*\*：作用域控制的内存循环操作
- \*\*局部指针循环变式\*\*：局部指针在循环中的操作
- \*\*内存生存期循环变式\*\*：内存生存期与循环的交互
- \*\*循环局部内存变式\*\*：循环中局部内存的管理

#### #### 10. 四重混合变式 (循环+分支+内存+作用域)

- **全属性综合变式**: 四种属性的完全融合分析
- **复杂算法模拟变式**: 复杂算法中的多属性交互
- **系统级操作变式**: 系统级操作的多属性分析
- **数据结构操作变式**: 复杂数据结构操作的综合分析
- **状态机实现变式**: 状态机实现的多属性分析
- **内存管理系统变式**: 内存管理系统的综合属性

## ## 复杂度层次设计

### ### 简单混合 (Easy) - 10个变式

- 2种属性类型的基础组合
- 清晰的执行流程和属性交互
- 直观的分析路径
- 基础的综合理解

### ### 中等混合 (Medium) - 15个变式

- 2-3种属性类型的中度组合
- 包含一定的嵌套结构
- 中等复杂度的属性交互
- 需要仔细的分析过程

### ### 复杂混合 (Hard) - 15个变式

- 3种属性类型的深度融合
- 复杂的嵌套和交互结构
- 多层次的属性分析
- 涉及高级概念的理解

### ### 专家级混合 (Expert) - 8个变式

- 3-4种属性类型的极度复杂组合
- 深度嵌套和复杂交互
- 需要专业级别的综合分析
- 涉及系统级和算法级的理解

### ### 大师级混合 (Master) - 2个变式

- 四种属性的完全融合
- 极度复杂的综合分析
- 需要最高水平的理解能力
- 挑战性的综合推理

## ## 生成策略

### ### 种子分析策略

1. **识别属性特征**: 分析每类种子的核心属性特征
2. **提取交互模式**: 识别不同属性间的潜在交互模式
3. **确定融合点**: 找出属性融合的自然切入点
4. **保持分析深度**: 在混合中保持每种属性的分析深度

### ### 变式设计原则

1. **多维融合**: 每个变式都应体现多种属性的有机结合
2. **英文描述**: 所有task描述必须使用标准英文
3. **答案唯一**: 严格确保答案的准确性和唯一性
4. **分析挑战**: 每个变式都应提供有价值的数据挑战

### 质量保证

1. \*\*属性一致性\*\*: 确保各种属性的语义一致性
2. \*\*交互正确性\*\*: 验证属性间交互的正确性
3. \*\*分析可行性\*\*: 确保综合分析的可行性和合理性
4. \*\*答案验证\*\*: 严格验证答案的准确性和唯一性

## 输出格式要求

```
```json
[
  {
    "id": "PL-MIX-V001",
    "metadata": {
      "name": "PropertyLevel-Mix-VariantName",
      "category": "Property-Level",
      "subcategory": "Mix",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert/master",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
      "primary_focus": "主要关注的属性组合",
      "analysis_complexity": "low/medium/high/extreme"
    },
    "task": {
      "description": "English description of the mixed property analysis task",
      "code": "Code containing multiple property types to analyze",
      "answer": "Unique and accurate comprehensive analysis result"
    }
  },
  {下一个变式...}
]
```
```

特殊字段说明

**mixed\_properties**: 标识该变式包含的属性类型组合

**primary\_focus**: 标识主要的分析焦点

**analysis\_complexity**: 标识综合分析的复杂程度

生成目标

生成总计约50个代码属性推理大混合变式，分布如下：

简单混合变式：10个 (2种属性组合)

中等混合变式：15个 (2-3种属性组合)

复杂混合变式：15个 (3种属性组合)

专家级混合变式：8个 (3-4种属性组合)

大师级混合变式：2个 (4种属性完全融合)

重点测试场景

多属性协同分析：同时分析多种属性的协同效应

交互效应理解：理解不同属性间的相互影响

综合状态追踪：跟踪多个维度的程序状态变化

复杂控制流分析：分析复杂控制流中的多重属性

内存-作用域交互：理解内存操作与作用域的交互

循环-分支协同：分析循环与分支的协同效应

## 英文描述模板示例

"Analyze the comprehensive interaction of loop iterations, branch conditions, memory operations, and scope management..."

"Determine the final state considering the combined effects of conditional logic, pointer arithmetic, and variable scoping..."

"Trace the evolution of variables through nested loops with conditional memory access and scope changes..."

"Calculate the result of complex property interactions involving iteration counting, branch selection, and memory modifications..."

"Evaluate the

## 混合模式优先级

实用导向：优先设计实际编程中常见的属性组

教学价值：重点关注有教学和测试价值的组

渐进复杂：从简单组合逐步增加到复杂综合

平衡覆盖：确保各种属性组合的平衡覆盖  
请基于此提示词和提供的所有种子任务，生成约50个代码属性推理大混合变式集合，严格确保答案准确且唯一，所有描述使用英文。输出格式为包含所有变式的JSON数组

```
[
 {
 "id": "PL-MIX-V001",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchBasic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_branch_basic",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Conditional loop iteration counting",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Analyze the loop with conditional branching - how many times is the variable 'count' incremented?",
 "code": "int count = 0;\nfor (int i = 1; i <= 10; i++) {\n if (i % 3 == 0)
 count++;\n }\nprintf(\"Count: %d\\n\", count);",
 "answer": 3
 }
 },
 {
 "id": "PL-MIX-V002",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopMemorySimple",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_memory_simple",
 "mixed_properties": ["Loop", "Memory"],
 "primary_focus": "Conditional loop iteration counting",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Analyze the loop with conditional branching - how many times is the variable 'count' incremented?",
 "code": "int count = 0;\nfor (int i = 1; i <= 10; i++) {\n if (i % 3 == 0)
 count++;\n }\nprintf(\"Count: %d\\n\", count);",
 "answer": 3
 }
 }
]
```

```
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_memory_basic",
 "mixed_properties": ["Loop", "Memory"],
 "primary_focus": "Pointer arithmetic in loops",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Track pointer movement through the loop - what value does ptr point to after the loop completes?",
 "code": "int arr[5] = {10, 20, 30, 40, 50};\nint *ptr = arr;\nfor (int i = 0; i < 3; i++) {\n ptr++;\n}\nint result = *ptr;",
 "answer": 40
 }
},
{
 "id": "PL-MIX-V003",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchScopeBasic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "branch_scope_basic",
 "mixed_properties": ["Branch", "Scope"],
 "primary_focus": "Variable shadowing in branches",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Determine which variable value is used in the branch - what value is returned?",
 "code": "int x = 5;\nint test_function(int condition) {\n if (condition > 0)\n int x = 15;\n return x + 2;\n}\n return x * 2;\n}// Called with test_function(1)",
 "answer": 17
 }
},
{
 "id": "PL-MIX-V004",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryScopeBasic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "memory_scope_basic",
 "mixed_properties": ["Memory", "Scope"],
 "primary_focus": "Memory and scope interaction in loops",
 "analysis_complexity": "low"
 }
}
```

```

 "primary_focus": "Pointer access with variable scoping",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Analyze pointer access across scopes - what is the final value of the dereferenced pointer?",
 "code": "int global_val = 100;\nint *ptr;\nvoid set_pointer() {\n int local_val = 200;\n ptr = &global_val;\n}\nset_pointer();\nint result = *ptr;",
 "answer": 100
 }
},
{
 "id": "PL-MIX-V005",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchCondition",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_branch_condition",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Branch-controlled loop termination",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Analyze the loop with early termination - how many iterations are completed before the break?",
 "code": "int iterations = 0;\nfor (int i = 1; i <= 15; i++) {\n iterations++;\n if (i * i > 50) {\n break;\n }\n}\nprintf(\"Iterations: %d\\n\", iterations);",
 "answer": 8
 }
},
{
 "id": "PL-MIX-V006",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopScopeVariable",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_scope_variable",
 "mixed_properties": ["Loop", "Scope"],
 "primary_focus": "Variable scope in loop blocks",
 "analysis_complexity": "low"
 },
 "task": {

```

```

 "description": "Track variable values across loop scopes - what is the final
value of outer_sum?",
 "code": "int outer_sum = 0;\nfor (int i = 1; i <= 3; i++) {\n int inner_val =
i * 2;\n outer_sum += inner_val;\n}\nprintf(\"Sum: %d\\n\", outer_sum);",
 "answer": 12
 },
},
{
 "id": "PL-MIX-V007",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchMemoryAccess",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "branch_memory_access",
 "mixed_properties": ["Branch", "Memory"],
 "primary_focus": "Conditional pointer operations",
 "analysis_complexity": "Low"
 },
 "task": {
 "description": "Analyze conditional pointer modification - what value is stored
at arr[1] after the operations?",
 "code": "int arr[3] = {10, 20, 30};\nint *ptr = &arr[1];\nif (arr[0] < 15) {\n *ptr = *ptr + 5;\n} else {\n *ptr = *ptr * 2;\n}\nint result = arr[1];",
 "answer": 25
 }
},
{
 "id": "PL-MIX-V008",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryScopeLifetime",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "memory_scope_lifetime",
 "mixed_properties": ["Memory", "Scope"],
 "primary_focus": "Variable lifetime and pointer validity",
 "analysis_complexity": "Low"
 },
 "task": {
 "description": "Analyze static variable access through pointer - what value is
returned after the second function call?",
 "code": "int* get_static_ptr() {\n static int value = 10;\n value += 5;\n return &value;\n}\nint *ptr1 = get_static_ptr();\nint *ptr2 = get_static_ptr();\nint
result = *ptr2;",
 }
}

```

```

 "answer": 20
 }
},
{
 "id": "PL-MIX-V009",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchScopestatic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "branch_scope_static",
 "mixed_properties": ["Branch", "Scope"],
 "primary_focus": "Static variable modification in branches",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Track static variable through conditional branches - what value is returned by the third call?",
 "code": "int conditional_increment(int flag) {\n static int counter = 0;\n\n if (flag > 0) {\n counter += 2;\n } else {\n counter += 1;\n }\n\n return counter;\n}\n\n// Called three times: conditional_increment(1), conditional_increment(0), conditional_increment(1)",
 "answer": 5
 }
},
{
 "id": "PL-MIX-V010",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopMemoryModify",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "loop_memory_modify",
 "mixed_properties": ["Loop", "Memory"],
 "primary_focus": "Array modification through loops",
 "analysis_complexity": "low"
 },
 "task": {
 "description": "Calculate array element after loop modifications - what is the value of data[2] after the loop?",
 "code": "int data[4] = {1, 2, 3, 4};\n\nfor (int i = 0; i < 3; i++) {\n data[i + 1] = data[i] + data[i + 1];\n}\n\nint result = data[2];",
 "answer": 6
 }
},

```

```
{
 "id": "PL-MIX-V011",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchAccumulator",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_branch_accumulator",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Conditional accumulation in loops",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze conditional accumulation pattern - what is the final value of sum after processing all elements?",
 "code": "int numbers[6] = {2, 7, 4, 9, 6, 3};\nint sum = 0;\nfor (int i = 0; i < 6; i++) {\n if (numbers[i] % 2 == 0) {\n sum += numbers[i];\n } else {\n sum += numbers[i] * 2;\n }\n}\nprintf(\"Final sum: %d\\n\", sum);",
 "answer": 44
 }
},
{
 "id": "PL-MIX-V012",
 "metadata": {
 "name": "PropertyLevel-Mix-NestedLoopBranch",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "nested_loop_branch",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Nested loops with conditional logic",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Count conditional executions in nested loops - how many times is the inner operation executed?",
 "code": "int operations = 0;\nfor (int i = 1; i <= 4; i++) {\n for (int j = 1; j <= i; j++) {\n if ((i + j) % 2 == 0) {\n operations++;\n }\n }\n}\nprintf(\"Operations: %d\\n\", operations);",
 "answer": 4
 }
},
{
 "id": "PL-MIX-V013",
 "metadata": {

```

```

 "name": "PropertyLevel-Mix-LoopMemoryPointer",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_memory_pointer",
 "mixed_properties": ["Loop", "Memory"],
 "primary_focus": "Pointer traversal with modifications",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Track pointer-based array modification - what is the value of buffer[3] after the loop?",
 "code": "int buffer[5] = {5, 10, 15, 20, 25};\nint *ptr = buffer + 1;\nfor (int i = 0; i < 3; i++) {\n *ptr = *ptr + *(ptr - 1);\n ptr++;\n}\nint result = buffer[3];",
 "answer": 40
 }
},
{
 "id": "PL-MIX-V014",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchMemoryStruct",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "branch_memory_struct",
 "mixed_properties": ["Branch", "Memory"],
 "primary_focus": "Conditional struct member access",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze conditional struct modifications - what is the final value of point.y?",
 "code": "typedef struct { int x; int y; } Point;\nPoint point = {3, 4};\nif (point.x > 2) {\n point.y = point.y * 2;\n}\nif (point.y > 6) {\n point.x = point.x + point.y;\n}\nint result = point.y;",
 "answer": 8
 }
},
{
 "id": "PL-MIX-V015",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopScopeStatic",
 "category": "Property-Level",
 "subcategory": "Mix",

```

```
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_scope_static",
 "mixed_properties": ["Loop", "Scope"],
 "primary_focus": "Static variables in loop iterations",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Track static variable across loop iterations - what is the final value of the static variable?",
 "code": "int process_loop() {\n static int persistent = 1;\n for (int i = 0; i < 4; i++) {\n persistent = persistent * 2;\n }\n return persistent;\n}\nint result1 = process_loop();\nint result2 = process_loop();",
 "answer": 256
 }
},
{
 "id": "PL-MIX-V016",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchScopeNested",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "branch_scope_nested",
 "mixed_properties": ["Branch", "Scope"],
 "primary_focus": "Nested scopes with conditional access",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze nested scope variable access - what value is returned by the function?",
 "code": "int x = 100;\nint nested_scope_test(int flag) {\n int x = 50;\n if (flag > 0) {\n int x = 25;\n if (flag > 5) {\n return x + 10;\n }\n return x * 2;\n }\n return x + 5;\n}\n// called with nested_scope_test(3)",
 "answer": 50
 }
},
{
 "id": "PL-MIX-V017",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryScopeGlobal",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
```

```

 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "memory_scope_global",
 "mixed_properties": ["Memory", "Scope"],
 "primary_focus": "Global variable pointer access",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Track global variable modification through pointers - what is the final value of global_data?",
 "code": "int global_data = 10;\nvoid modify_through_pointer() {\n int *ptr =\n&global_data;\n *ptr = *ptr + 15;\n {\n int local_data = 5;\n *ptr =\n*ptr + local_data;\n }\n} \nmodify_through_pointer();\nint result = global_data;",
 "answer": 30
 }
},
{
 "id": "PL-MIX-V018",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchContinue",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_branch_continue",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Loop control with continue statements",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze loop with continue statements - how many times is the increment operation executed?",
 "code": "int counter = 0;\nfor (int i = 1; i <= 10; i++) {\n if (i % 3 == 0)\n continue;\n if (i % 2 == 0) {\n counter += 2;\n } else {\n counter += 1;\n }\n} \nprintf(\"Counter: %d\\n\", counter);",
 "answer": 11
 }
},
{
 "id": "PL-MIX-V019",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryArrayBounds",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,

```

```

 "variant_type": "memory_array_bounds",
 "mixed_properties": ["Memory", "Loop"],
 "primary_focus": "Array boundary handling in loops",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Calculate sum with boundary-aware array access - what is the final value of total?",
 "code": "int values[4] = {2, 4, 6, 8};\nint total = 0;\nfor (int i = 0; i < 6; i++) {\n if (i < 4) {\n total += values[i];\n } else {\n total += i;\n }\n}\nprintf(\"Total: %d\\n\", total);",
 "answer": 29
 }
},
{
 "id": "PL-MIX-V020",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchMemoryIndirect",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "branch_memory_indirect",
 "mixed_properties": ["Branch", "Memory"],
 "primary_focus": "Conditional indirect memory access",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze conditional pointer indirection - what value is stored in the target location?",
 "code": "int data1 = 100, data2 = 200;\nint *ptr1 = &data1, *ptr2 = &data2;\nint **indirect_ptr;\nif (data1 < data2) {\n indirect_ptr = &ptr1;\n **indirect_ptr =\n **indirect_ptr + 50;\n} else {\n indirect_ptr = &ptr2;\n **indirect_ptr =\n **indirect_ptr + 25;\n}\nint result = data1;",
 "answer": 150
 }
},
{
 "id": "PL-MIX-V021",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchMultiCondition",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_branch_multi_condition",
 "mixed_properties": ["Loop", "Branch"],
 "primary_focus": "Loop boundary handling in multi-condition branches"
 }
}

```

```

 "primary_focus": "Multiple conditional branches in loops",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Process multiple conditions in loop - what is the final value of result?",
 "code": "int result = 0;\nfor (int i = 1; i <= 8; i++) {\n if (i % 2 == 0 &&\n i % 4 != 0) {\n result += i;\n } else if (i % 3 == 0) {\n result += i * 2;\n } else {\n result += 1;\n }\n}\nprintf(\"Result: %d\\n\", result);",
 "answer": 30
 }
},
{
 "id": "PL-MIX-V022",
 "metadata": {
 "name": "PropertyLevel-Mix-ScopeMemoryLocal",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "scope_memory_local",
 "mixed_properties": ["Scope", "Memory"],
 "primary_focus": "Local variable pointer relationships",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze local variable pointer operations - what is the final value accessed through the pointer?",
 "code": "int outer_function() {\n int local_var = 20;\n int *ptr = &local_var;\n {\n int inner_var = 15;\n *ptr = *ptr + inner_var;\n }\n *ptr = *ptr * 2;\n return *ptr;\n}\nint result = outer_function();",
 "answer": 70
 }
},
{
 "id": "PL-MIX-V023",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopMemoryOverlap",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_memory_overlap",
 "mixed_properties": ["Loop", "Memory"],
 "primary_focus": "Overlapping memory access in loops",
 "analysis_complexity": "medium"
 },
}

```

```

"task": {
 "description": "Track overlapping array modifications - what is the value of array[1] after all operations?",
 "code": "int array[4] = {10, 20, 30, 40};\nfor (int i = 0; i < 3; i++) {\n array[i] = array[i] + array[i + 1];\n}\nfor (int i = 3; i > 0; i--) {\n array[i] = array[i] - array[i - 1];\n}\nint result = array[1];",
 "answer": -30
},
{
 "id": "PL-MIX-V024",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchScopeParameter",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "branch_scope_parameter",
 "mixed_properties": ["Branch", "Scope"],
 "primary_focus": "Parameter shadowing with conditionals",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Analyze parameter shadowing in conditional blocks - what value is returned?",
 "code": "int param_test(int value) {\n value = value + 10;\n if (value > 15) {\n int value = 5;\n value = value * 3;\n return value;\n }\n return value * 2;\n} // Called with param_test(8)",
 "answer": 15
 }
},
{
 "id": "PL-MIX-V025",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopScopeCapture",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "loop_scope_capture",
 "mixed_properties": ["Loop", "Scope"],
 "primary_focus": "Variable capture across loop iterations",
 "analysis_complexity": "medium"
 },
 "task": {
 "description": "Track variable modification across loop scopes - what is the final value of accumulator?"
 }
}

```

```

 "code": "int accumulator = 0;\nfor (int i = 1; i <= 4; i++) {\n {\n int multiplier = i * 2;\n accumulator += multiplier;\n }\n accumulator += i;\n}\nprintf(\"Accumulator: %d\\n\", accumulator);",
 "answer": 30
}
},
{
 "id": "PL-MIX-V026",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchMemoryArray",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "loop_branch_memory_array",
 "mixed_properties": ["Loop", "Branch", "Memory"],
 "primary_focus": "Conditional array processing in loops",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Analyze complex array processing with conditional logic - what is the final value of data[2]?",
 "code": "int data[5] = {1, 3, 5, 7, 9};\nint *ptr = data;\nfor (int i = 0; i < 4; i++) {\n if (*(ptr + i) % 2 == 1) {\n *(ptr + i + 1) = *(ptr + i + 1) + *(ptr + i);\n } else {\n *(ptr + i) = *(ptr + i) * 2;\n }\n}\nint result = data[2];",
 "answer": 9
 }
},
{
 "id": "PL-MIX-V027",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchScopeStatic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "loop_branch_scope_static",
 "mixed_properties": ["Loop", "Branch", "Scope"],
 "primary_focus": "Static variable management with conditional loops",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Track static variable through conditional loop iterations - what value is returned by the third function call?",

```

```
 "code": "int conditional_static_loop(int limit) {\n static int persistent =\n 2;\n for (int i = 0; i < limit; i++) {\n if (i % 2 == 0) {\n persistent\n += i;\n } else {\n persistent *= 2;\n }\n }\n return persistent;\n}\n// Called three times: conditional_static_loop(2),\nconditional_static_loop(1), conditional_static_loop(3)",\n "answer": 32\n }\n},\n{\n "id": "PL-MIX-V028",\n "metadata": {\n "name": "PropertyLevel-Mix-BranchMemoryScopeIndirect",\n "category": "Property-Level",\n "subcategory": "Mix",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "branch_memory_scope_indirect",\n "mixed_properties": ["Branch", "Memory", "Scope"],\n "primary_focus": "Conditional indirect access with scoping",\n "analysis_complexity": "high"\n },\n "task": {\n "description": "Analyze conditional pointer operations across scopes - what is the final value accessed through target_ptr?",\n "code": "int global_value = 100;\nint* target_ptr;\nvoid setup_pointers(int condition) {\n int local_value = 50;\n if (condition > 0) {\n target_ptr =\n &global_value;\n *target_ptr += 25;\n } else {\n target_ptr =\n &local_value;\n *target_ptr += 10;\n }\n}\nsetup_pointers(1);\nint result =\n*target_ptr;\n",\n "answer": 125\n }\n},\n{\n "id": "PL-MIX-V029",\n "metadata": {\n "name": "PropertyLevel-Mix-LoopMemoryScopeLifetime",\n "category": "Property-Level",\n "subcategory": "Mix",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "hard",\n "intervention": 2,\n "variant_type": "loop_memory_scope_lifetime",\n "mixed_properties": ["Loop", "Memory", "Scope"],\n "primary_focus": "Memory lifetime management in loops",\n "analysis_complexity": "high"\n },\n "task": {\n
```

```

 "description": "Track memory access patterns across loop scopes - what is the
final value of the static variable?",
 "code": "int* get_persistent_ptr() {\n static int persistent = 5;\n return
&persistent;\n}\nint process_persistent_loop() {\n int sum = 0;\n for (int i = 0; i <
3; i++) {\n int *ptr = get_persistent_ptr();\n *ptr = *ptr + i;\n sum
+= *ptr;\n }\n return sum;\n}\nint result = process_persistent_loop();",
 "answer": 21
 }
},
{
 "id": "PL-MIX-V030",
 "metadata": {
 "name": "PropertyLevel-Mix-NestedLoopBranchMemory",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_loop_branch_memory",
 "mixed_properties": ["Loop", "Branch", "Memory"],
 "primary_focus": "Nested loops with conditional memory operations",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Analyze nested loops with conditional pointer operations - what
is the final value of matrix[1][1]?",
 "code": "int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};\nfor (int i = 0;
i < 2; i++) {\n for (int j = 0; j < 2; j++) {\n if ((i + j) % 2 == 0) {\n matrix[i][j] = matrix[i][j] + matrix[i + 1][j + 1];\n } else {\n matrix[i + 1][j + 1] = matrix[i + 1][j + 1] * 2;\n }\n }\n}\nint result =
matrix[1][1];",
 "answer": 10
 }
},
{
 "id": "PL-MIX-V031",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchScopeMemoryStruct",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "branch_scope_memory_struct",
 "mixed_properties": ["Branch", "Scope", "Memory"],
 "primary_focus": "Conditional struct operations with scoping",
 "analysis_complexity": "high"
 },
 "task": {

```

```
 "description": "Analyze conditional struct modifications across scopes - what is the final value of result.value?",
 "code": "typedef struct { int value; int multiplier; } Data;\nData global_data = {10, 2};\nData process_conditional(int flag) {\n Data result = global_data;\n if (flag > 0) {\n Data local_data = {5, 3};\n result.value = result.value + local_data.value;\n result.multiplier = local_data.multiplier;\n }\n result.value = result.value * result.multiplier;\n return result;\n}\nData final = process_conditional(1);\nint answer = final.value;",
 "answer": 45
 },
 {
 "id": "PL-MIX-V032",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopBranchScopeNested",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "loop_branch_scope_nested",
 "mixed_properties": ["Loop", "Branch", "Scope"],
 "primary_focus": "Nested scopes with conditional loops",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Track variables through nested scopes and conditional loops - what is the final value of outer_var?",
 "code": "int outer_var = 1;\nfor (int i = 0; i < 3; i++) {\n int loop_var = i + 1;\n if (loop_var % 2 == 1) {\n int inner_var = loop_var * 2;\n outer_var += inner_var;\n } else {\n outer_var += loop_var;\n }\n}\nprintf(\"Final: %d\\n\", outer_var);",
 "answer": 9
 },
 },
 {
 "id": "PL-MIX-V033",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryLoopPointerChain",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "memory_loop_pointer_chain",
 "mixed_properties": ["Memory", "Loop"],
 "primary_focus": "Pointer chain traversal in loops",
 "analysis_complexity": "high"
 },
 },
```

```

"task": {
 "description": "Follow pointer chain modifications through loop - what is the final value pointed to by the last pointer?",
 "code": "int values[4] = {10, 20, 30, 40};\nint *ptrs[4];\nfor (int i = 0; i < 4; i++) {\n ptrs[i] = &values[i];\n}\nfor (int i = 0; i < 3; i++) {\n *ptrs[i] = *ptrs[i] + *ptrs[i + 1];\n}\nint result = *ptrs[2];",
 "answer": 90
},
{

"id": "PL-MIX-V034",
"metadata": {
 "name": "PropertyLevel-Mix-ScopeBranchMemoryGlobal",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "scope_branch_memory_global",
 "mixed_properties": ["Scope", "Branch", "Memory"],
 "primary_focus": "Global variable conditional access patterns",
 "analysis_complexity": "high"
},
"task": {
 "description": "Analyze global variable modification through conditional scopes - what is the final value of global_counter?",
 "code": "int global_counter = 0;\nvoid conditional_modify(int level) {\n if (level > 0) {\n int local_val = level * 5;\n global_counter += local_val;\n }\n if (level > 2) {\n int deep_val = 10;\n global_counter += deep_val;\n }\n else {\n global_counter -= 5;\n }\n}\nconditional_modify(3);\nconditional_modify(1);\nint result = global_counter;",
 "answer": 30
},
{

"id": "PL-MIX-V035",
"metadata": {
 "name": "PropertyLevel-Mix-LoopMemoryBranchArray",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "loop_memory_branch_array",
 "mixed_properties": ["Loop", "Memory", "Branch"],
 "primary_focus": "Array processing with conditional modifications",
 "analysis_complexity": "high"
},
"task": {

```

```

 "description": "Process array with conditional memory operations - what is the sum of all array elements after processing?",
 "code": "int arr[5] = {2, 4, 6, 8, 10};\nfor (int i = 0; i < 4; i++) {\n if (arr[i] < arr[i + 1]) {\n arr[i + 1] = arr[i + 1] - arr[i];\n } else {\n arr[i] = arr[i] + arr[i + 1];\n }\n}\nint sum = 0;\nfor (int i = 0; i < 5; i++) {\n sum += arr[i];\n}\nint result = sum;",
 "answer": 20
 },
},
{
 "id": "PL-MIX-V036",
 "metadata": {
 "name": "PropertyLevel-Mix-ComplexLoopBranchScope",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "complex_loop_branch_scope",
 "mixed_properties": ["Loop", "Branch", "Scope"],
 "primary_focus": "Complex scope management in conditional loops",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Analyze complex variable interactions across loops and scopes - what is the final value returned?",
 "code": "int complex_scope_analysis() {\n int base = 5;\n int total = 0;\n\n for (int i = 1; i <= 3; i++) {\n int inner = base + i;\n total += inner;\n }\n\n total += base;\n\n return total;\n}\nint result = complex_scope_analysis();",
 "answer": 25
 },
},
{
 "id": "PL-MIX-V037",
 "metadata": {
 "name": "PropertyLevel-Mix-MemoryScopeLoopStatic",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "memory_scope_loop_static",
 "mixed_properties": ["Memory", "Scope", "Loop"],
 "primary_focus": "Static memory management across loop iterations",
 "analysis_complexity": "high"
 },
 "task": {

```

```

 "description": "Track static memory access through multiple function calls - what is the final value stored in the static array?",

 "code": "int* get_static_array_element(int index) {\n static int static_array[3] = {1, 2, 3};\n return &static_array[index];\n}\nvoid modify_static_array() {\n for (int i = 0; i < 3; i++) {\n int *ptr = get_static_array_element(i);\n *ptr = *ptr * (i + 1);\n }\n}\nmodify_static_array();\nmodify_static_array();\nint *final_ptr = get_static_array_element(2);\nint result = *final_ptr;",

 "answer": 36
 }
},
{
 "id": "PL-MIX-V038",
 "metadata": {
 "name": "PropertyLevel-Mix-BranchMemoryLoopIndirect",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "branch_memory_loop_indirect",
 "mixed_properties": ["Branch", "Memory", "Loop"],
 "primary_focus": "Conditional indirect memory access in loops",
 "analysis_complexity": "high"
 },
 "task": {
 "description": "Analyze conditional indirect memory operations in loops - what is the final value of target_value?",

 "code": "int target_value = 10;\nint other_value = 5;\nint *active_ptr = &target_value;\nfor (int i = 0; i < 4; i++) {\n if (i % 2 == 0) {\n *active_ptr += i;\n } else {\n active_ptr = (active_ptr == &target_value) ? &other_value : &target_value;\n *active_ptr += i * 2;\n }\n}\nint result = target_value;",

 "answer": 16
 }
},
{
 "id": "PL-MIX-V039",
 "metadata": {
 "name": "PropertyLevel-Mix-LoopScopeMemoryComplex",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "loop_scope_memory_complex",
 "mixed_properties": ["Loop", "Scope", "Memory"],
 "primary_focus": "Complex memory-scope interactions in loops",
 "analysis_complexity": "high"
 }
},

```

```

"task": {
 "description": "Track complex memory and scope interactions through loops - what is the final value of the return?",
 "code": "int complex_memory_scope() {\n int external = 100;\n int *ptr = &external;\n for (int i = 0; i < 3; i++) {\n int local = i * 10;\n if (i == 1) {\n ptr = &local;\n }\n *ptr += i;\n if (i == 1) {\n external += local;\n }\n }\n return external;\n}\nint result = complex_memory_scope();",
 "answer": 112
},
{
 "id": "PL-MIX-V040",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertLoopBranchMemory",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_loop_branch_memory",
 "mixed_properties": ["Loop", "Branch", "Memory"],
 "primary_focus": "Expert-level loop-branch-memory interaction",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Analyze expert-level property interactions - what is the final value of the computed result?",
 "code": "int expert_analysis() {\n int data[6] = {1, 2, 3, 4, 5, 6};\n int *left = data, *right = data + 5;\n int operations = 0;\n while (*left < *right) {\n if (*left + *right > 7) {\n *left = *left + (*right / 2);\n }\n operations++;\n } else {\n *right = *right - (*left % 3);\n }\n left++; \n right--;\n if (operations > 2) break;\n}\nint sum = 0;\nfor (int i = 0; i < 6; i++) {\n sum += data[i];\n}\nreturn sum;\n}\nint result = expert_analysis();",
 "answer": 25
 }
},
{
 "id": "PL-MIX-V041",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertBranchScopeMemory",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_branch_scope_memory",
 "mixed_properties": ["Branch", "Scope", "Memory"],
 "primary_focus": "Expert-level branch-scope-memory interaction"
 }
}

```

```

 "primary_focus": "Expert-level branch-scope-memory interaction",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Analyze expert-level conditional scope and memory management - what is the final global state?",
 "code": "int global_state = 0;\nint* global_ptr = &global_state;\n\nvoid expert_conditional(int level, int *external) {\n static int call_count = 0;\n call_count++;\n if (level > 0) {\n int local_state = level * call_count;\n if (call_count % 2 == 1) {\n global_ptr = &local_state;\n *global_ptr += level;\n *external += local_state;\n } else {\n global_ptr = &global_state;\n *global_ptr += local_state;\n }\n }\n nint external_val = 0;\n expert_conditional(3, &external_val);\n expert_conditional(2, &external_val);\n expert_conditional(1, &external_val);\n nint result = global_state + external_val;\n}\n\n \"answer\": 10
 }
},
{
 "id": "PL-MIX-V042",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertLoopScopeMemory",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_loop_scope_memory",
 "mixed_properties": ["Loop", "Scope", "Memory"],
 "primary_focus": "Expert-level loop-scope-memory interaction",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Analyze expert-level loop with complex scope and memory patterns - what is the final accumulated value?",
 "code": "int expert_loop_scope_memory() {\n static int persistent_data[3] = {10, 20, 30};\n int accumulator = 0;\n int *working_ptr;\n for (int cycle = 0; cycle < 3; cycle++) {\n working_ptr = &persistent_data[cycle];\n int local_multiplier = cycle + 1;\n *working_ptr = *working_ptr * local_multiplier;\n accumulator += *working_ptr;\n if (cycle > 0) {\n persistent_data[cycle - 1] += local_multiplier;\n }\n }\n return accumulator;\n}\n\n nint first_call = expert_loop_scope_memory();\n nint second_call = expert_loop_scope_memory();\n nint result = second_call;\n\n \"answer\": 1140
 }
},
{
 "id": "PL-MIX-V043",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertMultiDimensional",
 "category": "Property-Level",
 "subcategory": "Multi-Dimensional"
 }
}

```

```

"category": "Property-Level",
"subcategory": "Mix",
"type": "variant",
"source": "Generated",
"language": "c",
"difficulty": "expert",
"intervention": 3,
"variant_type": "expert_multi_dimensional",
"mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
"primary_focus": "Multi-dimensional property interaction",
"analysis_complexity": "extreme"
},
"task": {
"description": "Analyze multi-dimensional property interactions - what is the final state of the central element?",
"code": "int multi_dimensional_analysis() {\n static int matrix[3][3] = {{1,\n2, 3}, {4, 5, 6}, {7, 8, 9}};\n int *center = &matrix[1][1];\n for (int round = 0; round < 2; round++) {\n for (int i = 0; i < 3; i++) {\n if (matrix[round][i] % 2 == round % 2) {\n int temp = matrix[round][i];\n int local_factor = i + 1;\n *center += temp * local_factor;\n }\n }\n }\n int adjustment = round * 5;\n *center -= adjustment;\n return *center;\n}\nint result = multi_dimensional_analysis();",
"answer": 25
},
{
"id": "PL-MIX-V044",
"metadata": {
"name": "PropertyLevel-Mix-ExpertRecursiveProperty",
"category": "Property-Level",
"subcategory": "Mix",
"type": "variant",
"source": "Generated",
"language": "c",
"difficulty": "expert",
"intervention": 3,
"variant_type": "expert_recursive_property",
"mixed_properties": ["Branch", "Memory", "Scope"],
"primary_focus": "Recursive property state management",
"analysis_complexity": "extreme"
},
"task": {
"description": "Analyze recursive property state management - what is the final value after all recursive calls?",
```

```

 "code": "int shared_state = 1;\n\nint recursive_property_analysis(int depth, int
*accumulator) {\n static int call_depth = 0;\n call_depth++; \n if (depth <
0) {\n call_depth--;\n return shared_state;\n } \n int local_contribution = depth * call_depth; \n *accumulator += local_contribution;\n if (depth % 2 == 1) {\n shared_state *= 2;\n }\n } \n int result = recursive_property_analysis(depth - 1, accumulator);\n call_depth--;\n return result + depth;\n}\n\nint accumulator = 0;\nint final_result = recursive_property_analysis(3,
&accumulator);\nint answer = shared_state;",
 "answer": 20
 }
},
{
 "id": "PL-MIX-V045",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertStateManagement",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_state_management",
 "mixed_properties": ["Loop", "Branch", "Scope"],
 "primary_focus": "Expert-level state management across properties",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Analyze expert-level state management across multiple property
types - what is the final computed state?",
 "code": "int expert_state_management() {\n static int global_history[5] = {0};\n static int history_index = 0;\n int local_state = 0;\n for (int phase = 1; phase <= 4; phase++) {\n int phase_multiplier = phase;\n if (phase % 2 == 0) {\n local_state += phase_multiplier * 3;\n } else {\n local_state += phase_multiplier * 2;\n }\n global_history[history_index % 5] = local_state;\n history_index++;\n }\n if (phase > 2) {\n local_state += global_history[(history_index - 3) % 5];\n }\n return local_state;\n}\n\nint result = expert_state_management();",
 "answer": 30
 }
},
{
 "id": "PL-MIX-V046",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertInteractionPattern",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_interaction_pattern",
 "mixed_properties": ["Loop", "Branch", "Scope"],
 "primary_focus": "Expert-level state management across properties",
 "analysis_complexity": "extreme"
 }
}

```

```

 "intervention": 3,
 "variant_type": "expert_interaction_pattern",
 "mixed_properties": ["Memory", "Scope", "Branch"],
 "primary_focus": "Expert-level property interaction patterns",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Analyze expert-level property interaction patterns - what is the final value of the tracked variable?",
 "code": "int expert_interaction_pattern() {\n static int static_tracker = 5;\n int dynamic_state = 0;\n int *active_pointer;\n for (int step = 0; step < 3; step++) {\n if (step == 0) {\n active_pointer = &static_tracker;\n } else {\n active_pointer = &dynamic_state;\n }\n int step_factor = (step + 1) * 2;\n *active_pointer += step_factor;\n if (*active_pointer > 10) {\n static_tracker += step;\n dynamic_state = static_tracker / 2;\n }\n }\n return static_tracker + dynamic_state;\n}\nint result = expert_interaction_pattern();",
 "answer": 20
 }
},
{
 "id": "PL-MIX-V047",
 "metadata": {
 "name": "PropertyLevel-Mix-ExpertComprehensive",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "expert_comprehensive",
 "mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
 "primary_focus": "Comprehensive expert-level property analysis",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Perform comprehensive expert-level property analysis - what is the final system state value?",
 "code": "typedef struct { int value; int *ref; } StateNode;\n\nint expert_comprehensive_analysis() {\n static StateNode nodes[3] = { {10, NULL}, {20, NULL}, {30, NULL} };\n static int initialization_done = 0;\n if (!initialization_done) {\n for (int i = 0; i < 3; i++) {\n nodes[i].ref = &nodes[(i + 1) % 3].value;\n }\n initialization_done = 1;\n }\n int system_state = 0;\n for (int cycle = 0; cycle < 2; cycle++) {\n for (int node_idx = 0; node_idx < 3; node_idx++) {\n StateNode *current = &nodes[node_idx];\n int local_modifier = cycle + node_idx + 1;\n if (current->value % 2 == 0) {\n *(current->ref) += local_modifier;\n }\n system_state += current->value;\n }\n system_state += local_modifier;\n }\n return system_state;\n}\nint result = expert_comprehensive_analysis();",
 "answer": 100
 }
}

```

```

 "answer": 226
 }
},
{
 "id": "PL-MIX-V048",
 "metadata": {
 "name": "PropertyLevel-Mix-MasterComplexSystem",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "master",
 "intervention": 4,
 "variant_type": "master_complex_system",
 "mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
 "primary_focus": "Master-level complex system simulation",
 "analysis_complexity": "extreme"
 },
 "task": {
 "description": "Master-level complex system analysis with full property integration - what is the final equilibrium state?",
 "code": "typedef struct { int energy; int *connections[3]; int connection_count; } SystemNode;\n\nint master_complex_system() {\n static SystemNode network[4];\n\n static int system_initialized = 0;\n static int global_time = 0;\n\n if (!system_initialized) {\n for (int i = 0; i < 4; i++) {\n network[i].energy = (i + 1) * 10;\n network[i].connection_count = 0;\n\n for (int i = 0; i < 4; i++) {\n for (int j = 0; j < 3 && j < 4; j++) {\n if ((i + j + 1) % 4 != i) {\n network[i].connections[network[i].connection_count] = &network[(i + j + 1) % 4].energy;\n network[i].connection_count++;\n }\n }\n }\n\n system_initialized = 1;\n }\n\n for (int time_step = 0; time_step < 3; time_step++) {\n global_time++;\n for (int node_id = 0; node_id < 4; node_id++) {\n SystemNode *current_node = &network[node_id];\n\n if (current_node->energy > 25) {\n for (int conn = 0; conn < current_node->connection_count; conn++) {\n int transfer_amount =\n *(current_node->connections[conn]) +=\n current_node->energy -= transfer_amount;\n\n if (current_node->energy > 25) {\n int local_boost = (node_id + 1) * time_step;\n current_node->energy += local_boost;\n }\n }\n }\n\n int total_energy = 0;\n for (int i = 0; i < 4; i++) {\n total_energy +=\n network[i].energy;\n }\n return total_energy;\n }\n }\n }\n\n result = master_complex_system();\n\n "answer": 145\n}\n\n},
{
 "id": "PL-MIX-V049",
 "metadata": {
 "name": "PropertyLevel-Mix-MasterUltimateChallenge",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 }
}
```

```

 "source": "Generated",
 "language": "c",
 "difficulty": "master",
 "intervention": 4,
 "variant_type": "master_ultimate_challenge",
 "mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
 "primary_focus": "Ultimate master-level property integration challenge",
 "analysis_complexity": "extreme"
},
"task": {
 "description": "Ultimate master-level challenge integrating all four property types - what is the final computed result after all transformations?",
 "code": "typedef struct { int data[3]; int *active_ptr; int state; }\nProcessorUnit;\n\nint master_ultimate_challenge() {\n static ProcessorUnit\nprocessors[2];\n static int master_state = 100;\n static int initialization_phase = 1;\n\n if (initialization_phase) {\n for (int p = 0; p < 2; p++) {\n for (int d = 0; d < 3; d++) {\n processors[p].data[d] = (p + 1) * (d + 1) * 5;\n }\n processors[p].active_ptr = &processors[p].data[0];\n }\n processors[p].state = p * 10 + 5;\n }\n initialization_phase = 0;\n\n for (int execution_round = 0; execution_round < 2; execution_round++) {\n for (int proc_id = 0; proc_id < 2; proc_id++) {\n ProcessorUnit *current_proc =\n&processors[proc_id];\n for (int operation = 0; operation < 3;\noperation++) {\n int operation_factor =\nexecution_round + operation + 1;\n if ((current_proc->state % 3 == operation % 3) {\n *(current_proc->active_ptr) += operation_factor * master_state / 10;\n current_proc->active_ptr = ¤t_proc->data[(operation + 1) % 3];\n current_proc->data[operation] *= 2;\n\n if (current_proc->data[operation] > 50)\n master_state +=\n current_proc->state +=\n operation;\n }\n current_proc->active_ptr = ¤t_proc->data[proc_id];\n }\n master_state = master_state % 1000;\n }\n int final_sum = master_state;\n for (int p = 0; p < 2; p++) {\n for (int d = 0; d < 3; d++) {\n final_sum += processors[p].data[d];\n }\n }\n return final_sum % 1000;\n }\n result = master_ultimate_challenge();\n}\n\n"answer": 168
}
},
{
 "id": "PL-MIX-V050",
 "metadata": {
 "name": "PropertyLevel-Mix-MasterPropertyFusion",
 "category": "Property-Level",
 "subcategory": "Mix",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "master",
 "intervention": 4,
 "variant_type": "master_property_fusion",
 "mixed_properties": ["Loop", "Branch", "Memory", "Scope"],
 "primary_focus": "Master-level property fusion analysis",
 }
}

```

```

 "analysis_complexity": "extreme"
},
"task": {
 "description": "Master-level property fusion analysis requiring deep
understanding of all four property types - what is the final synthesized value?",
 "code": "int master_property_fusion() {\n static int fusion_matrix[3][3] =\n {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};\n static int *access_pattern[9];\n static int\n pattern_initialized = 0;\n if (!pattern_initialized) {\n int index = 0;\n for (int i = 0; i < 3; i++) {\n for (int j = 0; j < 3; j++) {\n access_pattern[index++] = &fusion_matrix[i][j];\n }\n }\n pattern_initialized = 1;\n int synthesis_state = 0;\n int *current_focus =\n access_pattern[0];\n for (int fusion_cycle = 0; fusion_cycle < 3; fusion_cycle++) {\n int cycle_modifier = fusion_cycle * 2 + 1;\n for (int transformation = 0; transformation < 3; transformation++) {\n int pattern_index = (fusion_cycle * 3 + transformation) % 9;\n current_focus =\n access_pattern[pattern_index];\n if (*current_focus % 2 ==\n fusion_cycle % 2) {\n *current_focus += cycle_modifier;\n synthesis_state += *current_focus;\n if\n (transformation > 0) {\n int prev_index = (pattern_index - 1 + 9) %\n 9;\n *(access_pattern[prev_index]) += transformation;\n }\n else {\n synthesis_state += (*current_focus) *\n cycle_modifier;\n *current_focus = (*current_focus + transformation) %\n 20;\n }\n }\n int final_matrix_sum = 0;\n for (int i = 0; i < 3; i++) {\n for (int j = 0; j < 3; j++) {\n final_matrix_sum +=\n fusion_matrix[i][j];\n }\n }\n return (synthesis_state +\n final_matrix_sum) % 1000;\n }\n }\n }\n int result = master_property_fusion();\n \"answer\": 163\n}\n}\n]

```

## 4 - 跨函数推理 [ MultiFunc-Level ] (285)

### 4A - 函数调用链推理 [ Call Chain ] (64)

```
函数调用链推理变式生成提示词
```

```
任务目标
```

基于给定的函数调用链推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对跨函数调用链的理解能力，包括直接调用、链式调用、递归调用和回调函数等核心函数调用链推理技能。

```
函数调用链推理特征分析
```

函数调用链推理关注程序中函数间的调用关系和数据流转，重点测试模型对函数参数传递、返回值传播、调用栈管理和执行顺序的深层理解，强调对跨函数边界的追踪和控制流分析能力。

```
关键要求
```

- **描述语言**: 所有task描述必须使用英文
- **变式数量**: 每个种子任务严格生成8-12个变式，确保数量充足
- **答案唯一性**: 保证答案准确且唯一，避免歧义和多解情况
- **调用链聚焦**: 重点关注函数调用链的属性分析，强调跨函数推理

## ## 变式生成维度

### ### 1. 直接调用链分析变式

- \*\*单层调用变式\*\*：简单的函数调用和返回值传递
- \*\*两层调用变式\*\*：函数A调用函数B的直接调用链
- \*\*三层调用变式\*\*：A->B->C的三层直接调用链
- \*\*参数传递变式\*\*：参数在调用链中的传递和变换
- \*\*返回值传播变式\*\*：返回值在调用链中的传播分析
- \*\*错误处理调用变式\*\*：包含错误处理的直接调用链

### ### 2. 链式调用分析变式

- \*\*方法链式调用变式\*\*：对象方法的链式调用分析
- \*\*函数式链式调用变式\*\*：函数式编程的链式调用
- \*\*数据变换链变式\*\*：数据在链式调用中的逐步变换
- \*\*流式处理链变式\*\*：流式数据处理的调用链
- \*\*管道式调用变式\*\*：管道模式的函数调用链
- \*\*构建器模式调用变式\*\*：构建器模式的链式调用

### ### 3. 递归调用分析变式

- \*\*简单递归变式\*\*：基础递归函数的调用链分析
- \*\*尾递归变式\*\*：尾递归优化的调用链分析
- \*\*相互递归变式\*\*：函数间相互递归的调用链
- \*\*递归深度变式\*\*：递归调用深度的分析
- \*\*递归累积变式\*\*：递归过程中的值累积分析
- \*\*条件递归变式\*\*：带有复杂条件的递归调用

### ### 4. 回调函数分析变式

- \*\*同步回调变式\*\*：同步回调函数的调用链分析
- \*\*函数指针回调变式\*\*：通过函数指针的回调调用
- \*\*高阶函数变式\*\*：高阶函数作为参数的调用链
- \*\*闭包回调变式\*\*：闭包函数的回调调用分析
- \*\*事件回调变式\*\*：事件驱动的回调函数链
- \*\*回调链式变式\*\*：多个回调函数的链式执行

### ### 5. 函数组合分析变式

- \*\*函数组合变式\*\*：函数组合操作的调用链分析
- \*\*柯里化函数变式\*\*：柯里化函数的调用链
- \*\*偏函数应用变式\*\*：偏函数应用的调用分析
- \*\*函数装饰器变式\*\*：装饰器模式的调用链
- \*\*中间件模式变式\*\*：中间件模式的函数调用
- \*\*管道组合变式\*\*：管道式函数组合的分析

### ### 6. 调用栈分析变式

- \*\*栈帧分析变式\*\*：调用栈帧的状态分析
- \*\*调用深度变式\*\*：调用栈深度的计算和分析
- \*\*栈溢出检测变式\*\*：栈溢出风险的分析
- \*\*调用轨迹变式\*\*：完整调用轨迹的追踪
- \*\*返回路径变式\*\*：函数返回路径的分析
- \*\*异常传播变式\*\*：异常在调用栈中的传播

### ### 7. 参数传递模式变式

- \*\*值传递变式\*\*：值传递在调用链中的行为
- \*\*引用传递变式\*\*：引用传递的调用链分析

- **指针传递变式**: 指针传递的调用链效果
- **结构体传递变式**: 复杂数据结构的传递
- **数组传递变式**: 数组在调用链中的传递
- **可变参数变式**: 可变参数的调用链分析

#### ### 8. 返回值处理变式

- **多返回值变式**: 多返回值的处理和传递
- **条件返回变式**: 条件控制的返回值选择
- **错误码返回变式**: 错误码的返回和处理
- **状态返回变式**: 状态信息的返回传递
- **引用返回变式**: 引用类型的返回值分析
- **链式返回变式**: 支持链式调用的返回值

#### ### 9. 调用优化分析变式

- **内联优化变式**: 函数内联对调用链的影响
- **尾调用优化变式**: 尾调用优化的分析
- **调用消除变式**: 死代码消除对调用的影响
- **常量折叠变式**: 常量折叠在调用链中的应用
- **调用重排变式**: 编译器调用重排的分析
- **缓存优化变式**: 调用结果缓存的分析

#### ### 10. 高级调用模式变式

- **动态调用变式**: 运行时动态函数调用
- **反射调用变式**: 反射机制的函数调用
- **虚函数调用变式**: 虚函数的动态调用分析
- **泛型函数调用变式**: 泛型函数的调用实例化
- **宏展开调用变式**: 宏展开对调用链的影响
- **模板实例化变式**: 模板实例化的调用分析

## ## 复杂度层次设计

### ### 简单调用链 (Easy)

- 2-3层的直接函数调用
- 简单的参数传递和返回值
- 清晰的调用顺序和数据流
- 基础的错误处理

### ### 中等调用链 (Medium)

- 3-4层的调用链或简单递归
- 包含条件分支的调用路径
- 中等复杂度的参数和返回值处理
- 基础的回调和高阶函数

### ### 复杂调用链 (Hard)

- 4-5层的深度调用链或复杂递归
- 复杂的数据变换和处理
- 高阶函数和函数组合
- 复杂的错误处理和状态管理

### ### 专家级调用链 (Expert)

- 极深或极复杂的调用链
- 高级的函数式编程概念
- 复杂的异步和回调处理

- 需要深度编程知识的调用分析

## 生成策略

### 种子分析策略

1. \*\*识别调用模式\*\*: 分析种子任务的主要调用模式和特征
2. \*\*提取数据流\*\*: 识别参数传递和返回值传播的模式
3. \*\*确定复杂度层次\*\*: 评估调用链的深度和复杂程度
4. \*\*分析关键节点\*\*: 找出调用链中的关键决策和变换节点

### 变式设计原则

1. \*\*调用链导向\*\*: 每个变式都应密切关注函数调用链的分析
2. \*\*英文描述\*\*: 所有task描述必须使用标准英文
3. \*\*答案唯一\*\*: 严格确保答案的准确性和唯一性
4. \*\*数量保证\*\*: 严格确保每个种子生成8-12个变式

### 质量保证

1. \*\*调用语义验证\*\*: 验证函数调用的语义正确性
2. \*\*数据流检查\*\*: 确保参数和返回值的数据流分析正确
3. \*\*答案唯一性验证\*\*: 严格检查答案的唯一性和确定性
4. \*\*英文质量保证\*\*: 确保描述的英文表达准确清晰

## 输出格式要求

```
```json
[
  {
    "id": "MF-CC-S00X-V001",
    "metadata": {
      "name": "MultiFunc-Callchain-VariantName",
      "category": "MultiFunc-Level",
      "subcategory": "Call chain",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "call_pattern": "direct/chain/recursive/callback/composition",
      "call_depth": "shallow/medium/deep",
      "data_flow": "simple/complex/bidirectional"
    },
    "task": {
      "description": "English description of the function call chain analysis task",
      "code": "Code containing function call chains to analyze",
      "answer": "Unique and accurate call chain analysis result"
    }
  },
  {下一个变式...}
]
```

```

特殊字段说明

**call\_pattern**: 标识主要的调用模式类型

**call\_depth:** 标识调用链的深度层次  
**data\_flow:** 标识数据流的复杂程度  
生成目标  
为每个提供的种子任务严格生成8-12个函数调用链变式，确保：

每种难度等级至少包含2-3个变式  
涵盖至少6-8种不同的调用模式分析类型  
包含不同的调用深度和复杂度  
所有**task**描述使用标准英文表达  
严格保证答案的准确性和唯一性  
重点测试场景  
参数传递追踪：准确追踪参数在调用链中的传递和变换  
返回值传播：正确分析返回值在调用链中的传播  
递归展开理解：理解递归调用的展开和计算过程  
回调执行顺序：分析回调函数的执行顺序和效果  
调用栈管理：理解调用栈的构建和管理机制  
数据流分析：分析复杂调用链中的数据流转  
英文描述模板示例

```
"Trace the function call chain and determine the final return value..."
"Analyze the parameter passing through multiple function calls..."
"Follow the recursive call execution and calculate the result..."
"Determine the execution order and final outcome of callback functions..."
"Trace the data transformation through the function call chain..."
"Calculate the result after all function calls in the chain complete..."
```

答案唯一性保证策略

具体输入值：使用具体的输入参数，避免参数化结果  
确定性执行：确保函数调用的执行路径是确定的  
避免副作用：避免不可预测的副作用影响结果  
精确计算：确保数值计算的精确性和一致性  
清晰语义：确保函数调用的语义在目标语言中明确

常见调用链模式

验证-处理-返回：输入验证->数据处理->结果返回  
解析-转换-执行：数据解析->格式转换->操作执行  
递归分解：问题递归分解和结果合并  
管道处理：数据在多个处理函数间流转  
回调驱动：事件驱动的回调函数执行  
高阶组合：高阶函数的组合和应用

请基于此提示词，为给定的函数调用链推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

```
[
 {
 "id": "MF-CC-S001-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-DirectCallValidation",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
```

```

 "variant_type": "simple_direct",
 "call_pattern": "direct",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace the parameter validation through this two-level call chain. What is the return value when check_input is called with value = 50?",
 "code": "int is_valid_range(int val) {\n return (val >= 10 && val <= 100) ? 1 : 0;\n}\n\nint check_input(int value) {\n if (is_valid_range(value)) {\n return value * 2;\n }\n return -1;\n}\n\n// What is returned by check_input(50)?",
 "answer": 100
 }
},
{
 "id": "MF-CC-S001-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-ThreeLayerDirect",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "three_layer_direct",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow the three-layer direct call chain and determine the final output when main_process is called with input = 8.",
 "code": "int base_check(int x) {\n return x > 5 ? 1 : 0;\n}\n\nint middle_transform(int x) {\n if (base_check(x)) {\n return x * 3;\n }\n return 0;\n}\n\nint main_process(int input) {\n int result = middle_transform(input);\n return result + 7;\n}\n\n// What is returned by main_process(8)?",
 "answer": 31
 }
},
{
 "id": "MF-CC-S001-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-ConditionalFlow",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "conditional_flow",
 "call_pattern": "direct",
 "call_depth": "shallow",
 "data_flow": "simple"
 }
}

```

```

 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace the conditional call path through multiple functions. What does execute_operation return with mode = 2 and value = 15?",
 "code": "int operation_a(int x) {\n return x + 10;\n}\n\nint operation_b(int x) {\n return x * 2;\n}\n\nint select_operation(int mode, int value) {\n if (mode == 1) {\n return operation_a(value);\n } else if (mode == 2) {\n return operation_b(value);\n } else {\n return 0;\n }\n}\n\nint execute_operation(int mode, int value) {\n int result = select_operation(mode, value);\n return result + 5;\n}\n\n// What is returned by execute_operation(2, 15)?",
 "answer": 35
 }
},
{
 "id": "MF-CC-S001-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-ErrorPropagation",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "error_propagation",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze error code propagation through the call chain. What is the final return value when process_data is called with input = -5?",
 "code": "int validate_positive(int x) {\n return x > 0 ? x : -1;\n}\n\nint double_if_valid(int x) {\n int validated = validate_positive(x);\n if (validated == -1) {\n return -1;\n } else {\n return validated * 2;\n }\n}\n\nint process_data(int input) {\n int processed = double_if_valid(input);\n if (processed == -1) {\n return 0;\n } else {\n return processed + 1;\n }\n}\n\n// What is returned by process_data(-5)?",
 "answer": 0
 }
},
{
 "id": "MF-CC-S001-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-ParameterTransform",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 }
}

```

```

 "variant_type": "parameter_transform",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Track parameter transformation through the call chain. What is the return value when final_calc is called with x = 12?",
 "code": "int scale_input(int val) {\n return val / 3;\n}\n\nint add_offset(int val) {\n int scaled = scale_input(val);\n return scaled + 8;\n}\n\nint final_calc(int x) {\n int adjusted = add_offset(x);\n return adjusted * 5;\n}\n\n// what is returned by final_calc(12)?",
 "answer": 60
 }
},
{
 "id": "MF-CC-S001-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-MultipleValidation",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "multiple_validation",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace through multiple validation layers in the call chain. What does secure_process return with data = 150?",
 "code": "int check_minimum(int val) {\n return val >= 100 ? 1 : 0;\n}\n\nint check_maximum(int val) {\n return val <= 200 ? 1 : 0;\n}\n\nint validate_range(int val)\n{\n return check_minimum(val) && check_maximum(val);\n}\n\nint apply_security(int val)\n{\n if (validate_range(val)) {\n return val - 50;\n } else {\n return -1;\n }\n}\n\nint secure_process(int data)\n{\n int secured = apply_security(data);\n if (secured == -1)\n return 0;\n else\n return secured + 25;\n}\n\n// what is returned by secure_process(150)?",
 "answer": 125
 }
},
{
 "id": "MF-CC-S001-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-DeepDirect",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 }
}

```

```

 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "deep_direct",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow this five-layer deep call chain and calculate the final result when start_process is called with n = 6.",
 "code": "int level_one(int x) {\n return x + 1;\n}\n\nint level_two(int x)\n{\n return level_one(x) * 2;\n}\n\nint level_three(int x) {\n return level_two(x) -\n3;\n}\n\nint level_four(int x) {\n return level_three(x) + 4;\n}\n\nint start_process(int n) {\n return level_four(n) / 2;\n}// What is returned by start_process(6)?",
 "answer": 9
 }
},
{
 "id": "MF-CC-S001-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-BranchedFlow",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "branched_flow",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze the branched call flow with multiple paths. What is returned by complex_process with type = 1 and value = 20?",
 "code": "int path_alpha(int x) {\n return x * 3;\n}\n\nint path_beta(int x)\n{\n return x + 15;\n}\n\nint route_call(int type, int x) {\n if (type == 1) {\n return path_alpha(x);\n }\n return path_beta(x);\n}\n\nint complex_process(int type, int value) {\n int routed = route_call(type, value);\n int final_path = route_call(2, routed);\n return final_path - 10;\n}// What is returned by complex_process(1, 20)?",
 "answer": 65
 }
},
{
 "id": "MF-CC-S001-V009",
 "metadata": {
 "name": "MultiFunc-CallChain-ReturnValueChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "return_value_chain",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "simple"
 }
}

```

```
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "return_value_chain",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace return value propagation through this call chain. What is the final result when chain_process is called with input = 7?",
 "code": "int step_one(int x) {\n return x * 4;\n}\n\nint step_two(int x) {\n int result = step_one(x);\n return result / 2;\n}\n\nint chain_process(int input) {\n int intermediate = step_two(input);\n return intermediate + input;\n}\n\n// What is returned by chain_process(7)?",
 "answer": 21
 }
},
{
 "id": "MF-CC-S002-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonValidationChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "simple_validation",
 "call_pattern": "direct",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow the validation and processing chain. What is returned by compute_result when given num = 45?",
 "code": "def check_bounds(x):\n return 20 <= x <= 80\n\ndef compute_result(num):\n if check_bounds(num):\n return num + 15\n else:\n return 0\n\n# What is returned by compute_result(45)?",
 "answer": 60
 }
},
{
 "id": "MF-CC-S002-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonMultistep",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
```

```

 "variant_type": "multi_step",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace the multi-step processing chain. What is the final value returned by process_number with x = 18?",
 "code": "def is_even(n):\n return n % 2 == 0\n\ndef transform_number(n):\nif is_even(n):\n return n // 2\nelse:\n return n * 3\n\nprocess_number(x):\n transformed = transform_number(x)\n return transformed + 12\n\n# what is returned by process_number(18)?",
 "answer": 21
 }
},
{
 "id": "MF-CC-S002-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonConditionalError",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "conditional_error",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze the error handling flow through the call chain. What does safe_divide return with numerator = 100 and denominator = 0?",
 "code": "def check_non_zero(value):\n return value != 0\n\ndef perform_division(num, den):\n if check_non_zero(den):\n return num / den\n else:\n return None\n\ndef safe_divide(numerator, denominator):\n result = perform_division(numerator, denominator)\n if result is None:\n return -1\n else:\n return int(result)\n\n# what is returned by safe_divide(100, 0)?",
 "answer": -1
 }
},
{
 "id": "MF-CC-S002-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonNestedValidation",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 }
}

```

```

 "variant_type": "nested_validation",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow the nested validation chain with multiple checks. What is returned by validate_and_process with data = 75?",
 "code": "def check_minimum_threshold(val):\n return val >= 50\n\ndef check_maximum_threshold(val):\n return val <= 100\n\ndef validate_thresholds(val):\n return check_minimum_threshold(val) and check_maximum_threshold(val)\n\ndef apply_formula(val):\n if validate_thresholds(val):\n return val * 0.8 + 20\n else:\n return 0\n\ndef validate_and_process(data):\n result = apply_formula(data)\n return int(result) if result > 0 else -1\n\n# What is returned by validate_and_process(75)?",
 "answer": 80
 }
},
{
 "id": "MF-CC-S002-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonStringProcessing",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "string_processing",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace string processing through the call chain. What is returned by process_string with text = 'Hello'?",
 "code": "def count_chars(s):\n return len(s)\n\ndef calculate_score(s):\n length = count_chars(s)\n if length > 3:\n return length * 10\n else:\n return 0\n\ndef process_string(text):\n score = calculate_score(text)\n return score + 5\n\n# What is returned by process_string('Hello')?",
 "answer": 55
 }
},
{
 "id": "MF-CC-S002-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonListProcessing",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0
 }
}

```

```

 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "list_processing",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow list processing through the call chain. What is returned by analyze_list with numbers = [2, 4, 6, 8]?",
 "code": "def sum_elements(lst):\n return sum(lst)\n\ndef check_even_sum(lst):\n total = sum_elements(lst)\n return total % 2 == 0\n\ndef analyze_list(numbers):\n if check_even_sum(numbers):\n return len(numbers) * 5\n else:\n return 0\n\n# What is returned by analyze_list([2, 4, 6, 8])?",
 "answer": 20
 }
},
{
 "id": "MF-CC-S002-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonComplexFlow",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "complex_flow",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze the complex conditional flow. What does master_process return with mode = 'A' and value = 30?",
 "code": "def process_mode_a(x):\n return x * 2 + 10\n\ndef process_mode_b(x):\n return x + 25\n\ndef select_processor(mode, x):\n if mode == 'A':\n return process_mode_a(x)\n elif mode == 'B':\n return process_mode_b(x)\n else:\n return 0\n\ndef validate_result(result):\n return result if result > 50 else 0\n\ndef master_process(mode, value):\n processed = select_processor(mode, value)\n validated = validate_result(processed)\n return validated + 5 if validated > 0 else -1\n\n# What is returned by master_process('A', 30)?",
 "answer": 75
 }
},
{
 "id": "MF-CC-S002-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonMathChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "complex_flow",
 "call_pattern": "direct",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze the complex conditional flow. What does master_process return with mode = 'A' and value = 30?",
 "code": "def process_mode_a(x):\n return x * 2 + 10\n\ndef process_mode_b(x):\n return x + 25\n\ndef select_processor(mode, x):\n if mode == 'A':\n return process_mode_a(x)\n elif mode == 'B':\n return process_mode_b(x)\n else:\n return 0\n\ndef validate_result(result):\n return result if result > 50 else 0\n\ndef master_process(mode, value):\n processed = select_processor(mode, value)\n validated = validate_result(processed)\n return validated + 5 if validated > 0 else -1\n\n# What is returned by master_process('A', 30)?",
 "answer": 75
 }
}

```

```

 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "math_chain",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
},
"task": {
 "description": "Follow the mathematical transformation chain. what is returned by calculate_final with base = 12?",
 "code": "def square_root_int(n):\n return int(n ** 0.5)\n\n\napply_math_ops(n):\n sqrt_val = square_root_int(n)\n return sqrt_val + n\n\n\n\ncalculate_final(base):\n result = apply_math_ops(base)\n return result * 2\n\n\n# What is returned by calculate_final(12)?\n# Note: int(12**0.5) = int(3.46) = 3",
 "answer": 30
},
{
 "id": "MF-CC-S003-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-DatabaseLayered",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "database_layered",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "simple"
},
"task": {
 "description": "Trace through the database query processing layers. what is returned by query_processor with sql = 'UPDATE table SET x=1?'",
 "code": "int check_sql_syntax(char *sql) {\n return strlen(sql) > 5 ? 1 : 0;\n}\n\n\nint validate_permissions(char *sql) {\n int syntax_ok = check_sql_syntax(sql);\n\n if (!syntax_ok) return 0;\n return strlen(sql) < 50 ? 1 : 0;\n}\n\n\nint query_processor(char *sql) {\n int permitted = validate_permissions(sql);\n\n if (!permitted) return -1;\n return strlen(sql) + 10;\n}\n\n\n// Given sql = 'UPDATE table SET x=1', what does query_processor return?\n// Note: strlen('UPDATE table SET x=1') = 21",
 "answer": 31
},
{
 "id": "MF-CC-S003-V002",
 "metadata": {
 "name": "MultiFunc-Callchain-ParserChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "parser_chain",
 "call_pattern": "direct",
 "call_depth": "medium",
 "data_flow": "simple"
}
}

```

```

 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "parser_chain",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow the parsing and compilation chain. What does compile_statement return with code = 'int x = 10;?'",
 "code": "int tokenize(char *code) {\n return strlen(code) / 2;\n}\n\nint parse_tokens(char *code) {\n int tokens = tokenize(code);\n return tokens > 3 ? tokens : 0;\n}\n\nint semantic_analysis(char *code) {\n int parsed = parse_tokens(code);\n if (parsed == 0) return -1;\n return parsed + 5;\n}\n\nint compile_statement(char *code) {\n int analyzed = semantic_analysis(code);\n if (analyzed == -1) return 0;\n return analyzed * 2;\n}\n\n// Given code = 'int x = 10;', what does compile_statement return?\n\nNote: strlen('int x = 10;') = 11",
 "answer": 20
 }
},
{
 "id": "MF-CC-S003-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-NetworkProtocol",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "network_protocol",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze the network packet processing chain. What is returned by process_packet with data = 'GET /index.html?'",
 "code": "int validate_header(char *data) {\n return strlen(data) >= 8 ? 1 : 0;\n}\n\nint check_protocol(char *data) {\n int header_valid = validate_header(data);\n if (!header_valid) return 0;\n return (data[0] == 'G') ? 2 : 1;\n}\n\nint route_request(char *data) {\n int protocol_code = check_protocol(data);\n if (protocol_code == 0) return -1;\n return protocol_code * strlen(data);\n}\n\nint process_packet(char *data) {\n int routed = route_request(data);\n if (routed == -1) return 0;\n return routed + 100;\n}\n\n// Given data = 'GET /index.html', what does process_packet return?\n\nNote: strlen('GET /index.html') = 15",
 "answer": 130
 }
},
{

```

```
{
 "id": "MF-CC-S003-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-FileProcessing",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "file_processing",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace file processing through the chain. What does process_file return with filename = 'data.txt'?",
 "code": "int check_extension(char *filename) {\n int len =\n strlen(filename);\n return (len > 4 && filename[len-4] == '.') ? 1 : 0;\n}\n\nint\nget_file_size(char *filename) {\n int has_ext = check_extension(filename);\n if\n (!has_ext) return 0;\n return strlen(filename) * 10;\n}\n\nint\nprocess_file(char *\nfilename) {\n int size = get_file_size(filename);\n return size > 0 ? size + 50 :\n -1;\n}\n\n// Given filename = 'data.txt', what does process_file return?\n// Note:\nstrlen('data.txt') = 8",
 "answer": 130
 }
},
{
 "id": "MF-CC-S003-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-SecurityCheck",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "security_check",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow the security validation chain. What is returned by\nsecure_access with token = 'AUTH123'?",
 }
}
```

```

 "code": "int validate_token_format(char *token) {\n return (strlen(token) >= 6) ? 1 : 0;\n}\n\nint check_token_prefix(char *token) {\n int format_ok = validate_token_format(token);\n if (!format_ok) return 0;\n return (token[0] == 'A' && token[1] == 'U') ? 1 : 0;\n}\n\nint authenticate_user(char *token) {\n int prefix_ok = check_token_prefix(token);\n if (!prefix_ok) return -1;\n return strlen(token) + 20;\n}\n\nint secure_access(char *token) {\n int auth_result = authenticate_user(token);\n if (auth_result == -1) return 0;\n return auth_result * 3;\n}\n\n// Given token = 'AUTH123', what does secure_access return?\n// Note: strlen('AUTH123') = 7",
 "answer": 81
}
},
{
 "id": "MF-CC-S003-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-DataValidation",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "data_validation",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze complex data validation chain. What does validate_input return with input = 'JSON{data:value}'?",
 "code": "int check_format(char *input) {\n return strlen(input) > 10 ? 1 : 0;\n}\n\nint validate_structure(char *input) {\n int format_valid = check_format(input);\n if (!format_valid) return 0;\n return (input[0] == 'J') ? 2 : 1;\n}\n\nint sanitize_data(char *input) {\n int structure_code = validate_structure(input);\n if (structure_code == 0) return -1;\n return structure_code * strlen(input) / 2;\n}\n\nint validate_input(char *input) {\n int sanitized = sanitize_data(input);\n if (sanitized == -1) return 0;\n return sanitized + strlen(input);\n}\n\n// Given input = 'JSON{data:value}', what does validate_input return?\n// Note: strlen('JSON{data:value}') = 16",
 "answer": 32
 }
},
{
 "id": "MF-CC-S003-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-ConfigParser",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 }
}

```

```

 "intervention": 0,
 "variant_type": "config_parser",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace configuration parsing chain. What does parse_config return with config = 'port=8080'?",
 "code": "int find_equals(char *config) {\n for (int i = 0; i < strlen(config); i++) {\n if (config[i] == '=') return i;\n }\n return -1;\n}\n\nint extract_value(char *config) {\n int eq_pos = find_equals(config);\n if (eq_pos == -1) return 0;\n return strlen(config) - eq_pos;\n}\n\nint parse_config(char *config) {\n int value_len = extract_value(config);\n if (value_len == 0) return -1;\n return value_len * 100 + strlen(config);\n}\n\n// Given config = 'port=8080', what does parse_config return?\n// Note: strlen('port=8080') = 9, equals at position 4",
 "answer": 509
 }
},
{
 "id": "MF-CC-S003-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-MessageProcessor",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "message_processor",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow message processing through complex chain. What is returned by handle_message with msg = 'HELLO WORLD'?",
 "code": "int count_spaces(char *msg) {\n int count = 0;\n for (int i = 0; i < strlen(msg); i++) {\n if (msg[i] == ' ') count++;\n }\n return count;\n}\n\nint analyze_message(char *msg) {\n int spaces = count_spaces(msg);\n int length = strlen(msg);\n return length + spaces * 5;\n}\n\nint process_content(char *msg) {\n int analysis = analyze_message(msg);\n return analysis > 15 ? analysis - 5 : 0;\n}\n\nint handle_message(char *msg) {\n int processed = process_content(msg);\n return processed > 0 ? processed + 10 : -1;\n}\n\n// Given msg = 'HELLO WORLD', what does handle_message return?\n// Note: strlen('HELLO WORLD') = 11, contains 1 space",
 "answer": 27
 }
},
{
 "id": "MF-CC-S004-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonSimpleTransform",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "transform",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Transform a call chain using a simple Python function. What does the transformed chain return for the input 'HELLO WORLD'?",
 "code": "def transform_chain(chain):\n transformed_chain = chain\n for i in range(len(chain) - 1):\n if chain[i] == ' ':\n transformed_chain[i] = 'H'\n return transformed_chain\n\ntransformed_chain = transform_chain('HELLO WORLD')\nprint(transformed_chain)",
 "answer": "H E L L O W O R L D"
 }
}

```

```

 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "simple_transform",
 "call_pattern": "chain",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow the simple data transformation chain. What is returned by process_value with x = 20?",
 "code": "def double(x):\n return x * 2\n\ndef add_five(x):\n doubled =\n double(x)\n return doubled + 5\n\ndef process_value(x):\n result = add_five(x)\n return result\n\n# What is returned by process_value(20)?",
 "answer": 45
 }
},
{
 "id": "MF-CC-S004-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonMathPipeline",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "math_pipeline",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace the mathematical transformation pipeline. What is the final result when calculate_result is called with input = 16?",
 "code": "def square_root(x):\n return int(x ** 0.5)\n\ndef multiply_by_three(x):\n root = square_root(x)\n return root * 3\n\ndef subtract_two(x):\n tripled = multiply_by_three(x)\n return tripled - 2\n\ndef calculate_result(input):\n final = subtract_two(input)\n return final + 1\n\n# What is returned by calculate_result(16)?\n\n# Note: int(16**0.5) = 4",
 "answer": 11
 }
},
{
 "id": "MF-CC-S004-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonStringChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain"
 }
}

```

```
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "string_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow string processing through transformation chain. What does transform_text return with text = 'Python'?",
 "code": "def get_length(s):\n return len(s)\n\ndef double_length(s):\n length = get_length(s)\n return length * 2\n\ndef add_ten(s):\n doubled = double_length(s)\n return doubled + 10\n\ndef transform_text(text):\n result = add_ten(text)\n return result\n\n# what is returned by transform_text('Python')?",
 "answer": 22
 }
},
{
 "id": "MF-CC-S004-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonConditionChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "condition_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze conditional transformation chain. What is returned by complex_transform with num = 35?",
 "code": "def check_divisible_by_five(n):\n return n % 5 == 0\n\ndef apply_condition(n):\n if check_divisible_by_five(n):\n return n / 5\n else:\n return n * 2\n\ndef final_adjustment(n):\n adjusted = apply_condition(n)\n return int(adjusted) + 3\n\ndef complex_transform(num):\n result = final_adjustment(num)\n return result\n\n# what is returned by complex_transform(35)?",
 "answer": 10
 }
},
{
 "id": "MF-CC-S004-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonListChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "list_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze list processing through transformation chain. What does transform_text return with text = 'Python'?",
 "code": "def get_length(s):\n return len(s)\n\ndef double_length(s):\n length = get_length(s)\n return length * 2\n\ndef add_ten(s):\n doubled = double_length(s)\n return doubled + 10\n\ndef transform_text(text):\n result = add_ten(text)\n return result\n\n# what is returned by transform_text('Python')?",
 "answer": 22
 }
}
]
```

```

 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "list_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace list processing through transformation chain. What does process_list return with data = [1, 2, 3, 4]?",
 "code": "def calculate_sum(lst):\n return sum(lst)\n\ndef get_average(lst):\n total = calculate_sum(lst)\n return total / len(lst)\n\ndef scale_average(lst):\n avg = get_average(lst)\n return avg * 4\n\ndef process_list(data):\n result = scale_average(data)\n return int(result)\n\n# What is returned by process_list([1, 2, 3, 4])?",
 "answer": 10
 }
},
{
 "id": "MF-CC-S004-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonNestedTransform",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "nested_transform",
 "call_pattern": "chain",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow nested transformation chain. What is the final result when deep_process is called with value = 8?",
 "code": "def level_one_transform(x):\n return x + 2\n\ndef level_two_transform(x):\n step1 = level_one_transform(x)\n return step1 * 3\n\ndef level_three_transform(x):\n step2 = level_two_transform(x)\n return step2 - 5\n\ndef deep_process(value):\n final = level_three_transform(value)\n return final / 2\n\n# What is returned by deep_process(8)?",
 "answer": 12.5
 }
},
{
 "id": "MF-CC-S004-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonFloatChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "float_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace float processing through transformation chain. What does process_list return with data = [1.1, 2.2, 3.3, 4.4]?",
 "code": "def calculate_sum(lst):\n return sum(lst)\n\ndef get_average(lst):\n total = calculate_sum(lst)\n return total / len(lst)\n\ndef scale_average(lst):\n avg = get_average(lst)\n return avg * 4\n\ndef process_list(data):\n result = scale_average(data)\n return int(result)\n\n# What is returned by process_list([1.1, 2.2, 3.3, 4.4])?",
 "answer": 10
 }
}

```

```
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "float_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace floating-point transformations through the chain. What does compute_final return with x = 100?",
 "code": "def convert_to_decimal(x):\n return x / 10.0\n\ndef apply_multiplier(x):\n decimal = convert_to_decimal(x)\n return decimal * 1.5\n\ndef add_offset(x):\n multiplied = apply_multiplier(x)\n return multiplied + 2.5\n\ndef compute_final(x):\n result = add_offset(x)\n return round(result, 1)\n\n# What is returned by compute_final(100)?",
 "answer": 17.5
 }
},
{
 "id": "MF-CC-S004-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonBooleanChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "boolean_chain",
 "call_pattern": "chain",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow boolean logic through transformation chain. What is returned by logic_chain with a = True and b = False?",
 "code": "def logical_and(x, y):\n return x and y\n\ndef logical_or(x, y):\n and_result = logical_and(x, y)\n return x or and_result\n\ndef convert_to_number(x, y):\n or_result = logical_or(x, y)\n return 1 if or_result else 0\n\ndef logic_chain(a, b):\n result = convert_to_number(a, b)\n return result * 10 + 5\n\n# What is returned by logic_chain(True, False)?",
 "answer": 15
 }
},
{
 "id": "MF-CC-S005-V001",
 "metadata": {
```

```
 "name": "MultiFunc-CallChain-SimpleRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "simple_recursion",
 "call_pattern": "recursive",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace the simple recursive countdown. What is returned by countdown when called with n = 5?",
 "code": "int countdown(int n) {\n if (n <= 0) {\n return 0;\n }\n return n + countdown(n - 1);\n}\n\n// What is returned by countdown(5)?",
 "answer": 15
 }
},
{
 "id": "MF-CC-S005-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-PowerRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "power_recursion",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow recursive power calculation. What does power return with base = 3 and exp = 4?",
 "code": "int power(int base, int exp) {\n if (exp == 0) {\n return 1;\n }\n return base * power(base, exp - 1);\n}\n\n// What is returned by power(3, 4)?",
 "answer": 81
 }
},
{
 "id": "MF-CC-S005-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-DigitSum",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
```

```

 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "digit_sum",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
},
"task": {
 "description": "Trace recursive digit sum calculation. What is returned by digit_sum with number = 1234?",
 "code": "int digit_sum(int number) {\n if (number == 0) {\n return 0;\n }\n return (number % 10) + digit_sum(number / 10);\n}\n// What is returned by digit_sum(1234)?",
 "answer": 10
},
{
 "id": "MF-CC-S005-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-GCDRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "gcd_recursion",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
},
"task": {
 "description": "Follow recursive GCD algorithm. What does gcd return with a = 48 and b = 18?",
 "code": "int gcd(int a, int b) {\n if (b == 0) {\n return a;\n }\n return gcd(b, a % b);\n}\n// What is returned by gcd(48, 18)?",
 "answer": 6
},
{
 "id": "MF-CC-S005-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-TreeSum",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,

```

```

 "variant_type": "tree_sum",
 "call_pattern": "recursive",
 "call_depth": "deep",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace recursive tree-like calculation. What is returned by tree_calc with n = 4?",
 "code": "int tree_calc(int n) {\n if (n <= 1) {\n return 1;\n }\n return tree_calc(n - 1) + tree_calc(n - 2) + n;\n}\n// What is returned by tree_calc(4)?\n// Note: tree_calc(1)=1, tree_calc(2)=tree_calc(1)+tree_calc(0)+2=1+1+2=4",
 "answer": 12
 }
},
{
 "id": "MF-CC-S005-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-ArrayRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "array_recursion",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow recursive array processing with helper function. What does process_array return with arr = [2, 4, 6] and size = 3?",
 "code": "int sum_array_recursive(int arr[], int index, int size) {\n if (index >= size) {\n return 0;\n }\n return arr[index] + sum_array_recursive(arr, index + 1, size);\n}\nint process_array(int arr[], int size) {\n int sum = sum_array_recursive(arr, 0, size);\n return sum * 2;\n}\n// Given arr = [2, 4, 6] and size = 3, what does process_array return?",
 "answer": 24
 }
},
{
 "id": "MF-CC-S005-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-MutualRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "mutual_recursion",
 "call_pattern": "mutual"
 }
}

```

```

 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace mutual recursion between two functions. What is returned by is_even when called with n = 6?",
 "code": "int is_odd(int n);\n\nint is_even(int n) {\n if (n == 0) return 1;\n return is_odd(n - 1);\n}\n\nint is_odd(int n) {\n if (n == 0) return 0;\n return is_even(n - 1);\n}\n\nint compute_result(int n) {\n return is_even(n) ? n * 2 : n + 5;\n}\n\n// What is returned by compute_result(6)?",
 "answer": 12
 }
},
{
 "id": "MF-CC-S005-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-ComplexRecursive",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "complex_recursive",
 "call_pattern": "recursive",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze complex recursive calculation with multiple base cases. What does complex_func return with n = 5?",
 "code": "int complex_func(int n) {\n if (n <= 0) return 0;\n if (n == 1) return 1;\n if (n == 2) return 2;\n return complex_func(n - 1) + complex_func(n - 2) + complex_func(n - 3);\n}\n\nint wrapper_func(int n) {\n int result = complex_func(n);\n return result + n;\n}\n\n// What is returned by wrapper_func(5)?\n// Note: complex_func(3)=complex_func(2)+complex_func(1)+complex_func(0)=2+1+0=3",
 "answer": 16
 }
},
{
 "id": "MF-CC-S006-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-SimpleTailRecursion",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "simple_tail_recursion",
 }
}

```

```
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace tail-recursive summation with accumulator. What is returned by sum_up_to when called with n = 6?",
 "code": "def sum_helper(n, acc):\n if n <= 0:\n return acc\n return sum_helper(n - 1, acc + n)\n\ndef sum_up_to(n):\n return sum_helper(n, 0)\n\n# What is returned by sum_up_to(6)?",
 "answer": 21
 }
},
{
 "id": "MF-CC-S006-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-TailPower",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "tail_power",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow tail-recursive power calculation. What does compute_power return with base = 2 and exponent = 5?",
 "code": "def power_helper(base, exp, acc):\n if exp <= 0:\n return acc\n return power_helper(base, exp - 1, acc * base)\n\ndef compute_power(base, exponent):\n return power_helper(base, exponent, 1)\n\n# What is returned by compute_power(2, 5)?",
 "answer": 32
 }
},
{
 "id": "MF-CC-S006-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-TailStringReverse",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "tail_string_reverse",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
 }
}
```



```

 "call_depth": "deep",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace tail-recursive Fibonacci with two accumulators. What is returned by fibonacci when called with n = 7?",
 "code": "def fib_helper(n, a, b):\n if n <= 0:\n return a\n return fib_helper(n - 1, b, a + b)\n\ndef fibonacci(n):\n return fib_helper(n, 0, 1)\n\n# What is returned by calculate_result(7)?",
 "answer": 23
 }
},
{
 "id": "MF-CC-S006-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-TailGCD",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "tail_gcd",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow tail-recursive GCD with result processing. What does enhanced_gcd return with x = 36 and y = 24?",
 "code": "def gcd_helper(a, b):\n if b == 0:\n return a\n return gcd_helper(b, a % b)\n\ndef calculate_gcd(x, y):\n return gcd_helper(x, y)\n\ndef enhanced_gcd(x, y):\n gcd_value = calculate_gcd(x, y)\n return gcd_value * 5\n\n# What is returned by enhanced_gcd(36, 24)?",
 "answer": 60
 }
},
{
 "id": "MF-CC-S006-V007",
 "metadata": {
 "name": "MultiFunc-CallChain-TailMultiply",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "tail_multiply",
 "call_pattern": "recursive",
 "call_depth": "deep",
 }
}

```

```

 "data_flow": "simple"
 },
 "task": {
 "description": "Trace complex tail recursion with multiplication. What is returned by multiply_range with start = 3 and end = 6?",
 "code": "def multiply_helper(current, end, acc):\n if current > end:\n return acc\n return multiply_helper(current + 1, end, acc * current)\n\nmultiply_range(start, end):\n return multiply_helper(start, end, 1)\n\ncompute_final(start, end):\n product = multiply_range(start, end)\n return product + 100\n\n# What is returned by compute_final(3, 6)?\n# Note: multiply_range(3, 6) = 3 * 4 * 5 * 6",
 "answer": 460
 }
},
{
 "id": "MF-CC-S006-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-TailDigitProduct",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "tail_digit_product",
 "call_pattern": "recursive",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze tail-recursive digit product calculation. What does digit_product_calc return with number = 234?",
 "code": "def digit_product_helper(n, acc):\n if n == 0:\n return acc\n return digit_product_helper(n // 10, acc * (n % 10))\n\ndigit_product(number):\n return digit_product_helper(number, 1)\n\ndigit_product_calc(number):\n product = digit_product(number)\n return product * 3 + 5\n\n# What is returned by digit_product_calc(234)?\n# Note: digit_product(234) = 2 * 3 * 4 = 24",
 "answer": 77
 }
},
{
 "id": "MF-CC-S007-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-SimpleCallback",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "simple_callback",
 }
}

```

```
 "call_pattern": "callback",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace simple callback execution. What is returned by execute_operation with value = 12 and the triple function?",
 "code": "typedef int (*operation_func)(int);\n\nint triple(int x) {\n return x * 3;\n}\n\nint execute_operation(int value, operation_func op) {\n return op(value) + 5;\n}\n\n// What is returned by execute_operation(12, triple)?",
 "answer": 41
 }
},
{
 "id": "MF-CC-S007-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-ChainedCallbacks",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "chained_callbacks",
 "call_pattern": "callback",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow chained callback execution. What does process_chain return with input = 8?",
 "code": "typedef int (*callback_func)(int);\n\nint square_func(int x) {\n return x * x;\n}\n\nint subtract_five(int x) {\n return x - 5;\n}\n\nint apply_two_callbacks(int val, callback_func first, callback_func second) {\n int result1 = first(val);\n int result2 = second(result1);\n return result2;\n}\n\nint process_chain(int input) {\n return apply_two_callbacks(input, square_func, subtract_five);\n}\n\n// What is returned by process_chain(8)?",
 "answer": 59
 }
},
{
 "id": "MF-CC-S007-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-ConditionalCallback",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "conditional_callback",
 "call_pattern": "callback",
 "call_depth": "medium",
 "data_flow": "complex"
 }
}
```

```

 "call_pattern": "callback",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze conditional callback selection. What is returned by smart_processor with data = 15 and mode = 1?",
 "code": "typedef int (*processor_func)(int);\n\nint halve(int x) {\n return x / 2;\n}\n\nint add_twenty(int x) {\n return x + 20;\n}\n\nprocessor_func select_callback(int mode) {\n return (mode == 1) ? halve : add_twenty;\n}\n\nint smart_processor(int data, int mode) {\n processor_func selected =\n select_callback(mode);\n int result = selected(data);\n return result + 3;\n}\n\n// What is returned by smart_processor(15, 1)?",
 "answer": 10
 }
},
{
 "id": "MF-CC-S007-v004",
 "metadata": {
 "name": "MultiFunc-CallChain-CallbackArray",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "callback_array",
 "call_pattern": "callback",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace callback execution from function array. What does execute_callbacks return with value = 6 and index = 2?",
 "code": "typedef int (*math_func)(int);\n\nint increment(int x) { return x + 1; }\n\nint double_val(int x) { return x * 2; }\n\nint square_val(int x) { return x * x; }\n\nmath_func operations[3] = {increment, double_val, square_val};\n\nint execute_callbacks(int value, int index) {\n if (index < 0 || index >= 3) return 0;\n int result = operations[index](value);\n return result + 10;\n}\n\n// What is returned by execute_callbacks(6, 2)?",
 "answer": 46
 }
},
{
 "id": "MF-CC-S007-v005",
 "metadata": {
 "name": "MultiFunc-CallChain-NestedCallback",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 }
}

```

```

 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "nested_callback",
 "call_pattern": "callback",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow nested callback execution. What is returned by nested_apply with x = 10?",
 "code": "typedef int (*transform_func)(int);\nint add_five(int x) {\n return x + 5;\n}\nint multiply_by_two(int x) {\n return x * 2;\n}\nint apply_nested(int val, transform_func outer, transform_func inner) {\n int inner_result = inner(val);\n int outer_result = outer(inner_result);\n return outer_result;\n}\nint nested_apply(int x) {\n return apply_nested(x, multiply_by_two, add_five) + 7;\n}\n// What is returned by nested_apply(10)?",
 "answer": 37
 }
},
{
 "id": "MF-CC-S007-v006",
 "metadata": {
 "name": "MultiFunc-CallChain-CallbackComposition",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "callback_composition",
 "call_pattern": "callback",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace callback function composition. What does compose_and_execute return with input = 4?",
 "code": "typedef int (*unary_func)(int);\nint cube(int x) {\n return x * x * x;\n}\nint subtract_ten(int x) {\n return x - 10;\n}\nint compose_functions(int val, unary_func f1, unary_func f2, unary_func f3) {\n int step1 = f1(val);\n int step2 = f2(step1);\n int step3 = f3(step2);\n return step3;\n}\nint compose_and_execute(int input) {\n return compose_functions(input, cube, subtract_ten, cube);\n}\n// What is returned by compose_and_execute(4)?\n// Note: cube(4)=64, subtract_ten(64)=54, cube(54)=157464",
 "answer": 157464
 }
},
{
 "id": "MF-CC-S007-v007",
 "metadata": {
 "name": "MultiFunc-CallChain-CallbackValidation",
 "category": "MultiFunc-Level",

```

```

 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "callback_validation",
 "call_pattern": "callback",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze callback with validation. What is returned by validated_process with data = 25?",
 "code": "typedef int (*validator_func)(int);\n\n\nint is_positive(int x) {\n return x > 0 ? 1 : 0;\n}\n\n\nint scale_by_three(int x) {\n return x * 3;\n}\n\n\nint apply_with_validation(int val, validator_func validate, processor_func process) {\n if (validate(val)) {\n return process(val);\n }\n return -1;\n}\n\n\nint validated_process(int data) {\n int result =\n apply_with_validation(data, is_positive, scale_by_three);\n return result > 0 ? result + 5 : 0;\n}\n\n\n// What is returned by validated_process(25)?",
 "answer": 80
 }
},
{
 "id": "MF-CC-S007-V008",
 "metadata": {
 "name": "MultiFunc-CallChain-CallbackPipeline",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "callback_pipeline",
 "call_pattern": "callback",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow callback pipeline execution. What does execute_pipeline return with value = 7?",
 "code": "typedef int (*stage_func)(int);\n\n\nint stage_one(int x) { return x + 3;\n}\n\n\nint stage_two(int x) { return x * 4; }\n\n\nint stage_three(int x) { return x - 8; }\n\n\nint pipeline_executor(int val, stage_func s1, stage_func s2, stage_func s3) {\n int after_s1 = s1(val);\n int after_s2 = s2(after_s1);\n int after_s3 = s3(after_s2);\n return after_s3;\n}\n\n\nint execute_pipeline(int value) {\n int pipelined =\n pipeline_executor(value, stage_one, stage_two, stage_three);\n return pipelined / 2;\n}\n\n\n// What is returned by execute_pipeline(7)?",
 "answer": 16
 }
},
{

```

```

{
 "id": "MF-CC-S008-V001",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonSimpleCompose",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 0,
 "variant_type": "simple_compose",
 "call_pattern": "composition",
 "call_depth": "shallow",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace simple function composition. What is returned by apply_composed with x = 8?",
 "code": "def add_two(x):\n return x + 2\n\ndef multiply_by_five(x):\n return x * 5\n\ndef compose_simple(f, g):\n return lambda x: f(g(x))\n\napply_composed(x):\n composed = compose_simple(multiply_by_five, add_two)\n return composed(x)\n\n# What is returned by apply_composed(8)?",
 "answer": 50
 }
},
{
 "id": "MF-CC-S008-V002",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonTripleCompose",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "triple_compose",
 "call_pattern": "composition",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Follow triple function composition. What does triple_compose_result return with value = 5?",
 "code": "def subtract_one(x):\n return x - 1\n\ndef square(x):\n return x * x\n\ndef add_three(x):\n return x + 3\n\ndef compose_three(f, g, h):\n return lambda x: f(g(h(x)))\n\ndef triple_compose_result(value):\n result_func = compose_three(add_three, square, subtract_one)\n return result_func(value)\n\n# What is returned by triple_compose_result(5)?",
 "answer": 19
 }
},

```

```
{
 "id": "MF-CC-S008-V003",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonApplyTwiceChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "apply_twice_chain",
 "call_pattern": "composition",
 "call_depth": "medium",
 "data_flow": "simple"
 },
 "task": {
 "description": "Trace apply_twice with different functions. What is returned by complex_application with n = 6?",
 "code": "def double(x):\n return x * 2\n\ndef apply_twice(f):\n return lambda x: f(f(x))\n\ndef add_one(x):\n return x + 1\n\ndef complex_application(n):\n twice_double = apply_twice(double)\n twice_add_one = apply_twice(add_one)\n\n intermediate = twice_double(n)\n final = twice_add_one(intermediate)\n\n return final\n\n# what is returned by complex_application(6)?",
 "answer": 26
 }
},
{
 "id": "MF-CC-S008-V004",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonPartialApplication",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 0,
 "variant_type": "partial_application",
 "call_pattern": "composition",
 "call_depth": "medium",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow partial application and composition. What does partial_chain return with x = 12?",
 "code": "def multiply(a, b):\n return a * b\n\ndef partial_left(f, a):\n return lambda b: f(a, b)\n\ndef add(x, y):\n return x + y\n\ndef partial_chain(x):\n multiply_by_three = partial_left(multiply, 3)\n add_ten = partial_left(add, 10)\n\n step1 = multiply_by_three(x)\n step2 = add_ten(step1)\n\n return step2\n\n# what is returned by partial_chain(12)?",
 "answer": 46
 }
}
```

```

},
{
 "id": "MF-CC-S008-V005",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonCurryingChain",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "currying_chain",
 "call_pattern": "composition",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Trace curried function composition. What is returned by curry_composition with a = 4 and b = 7?",
 "code": "def curry_add(x):\n return lambda y: x + y\n\ndef curry_multiply(x):\n return lambda y: x * y\n\ndef compose(f, g):\n return lambda x:\n f(g(x))\n\ndef curry_composition(a, b):\n add_a = curry_add(a)\n multiply_b =\n curry_multiply(b)\n composed = compose(multiply_b, add_a)\n return composed(5)\n\n# What is returned by curry_composition(4, 7)?",
 "answer": 63
 }
},
{
 "id": "MF-CC-S008-V006",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonNestedHigherOrder",
 "category": "MultiFunc-Level",
 "subcategory": "Call Chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "nested_higher_order",
 "call_pattern": "composition",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze nested higher-order function application. What does nested_higher_order return with value = 3?",
 "code": "def create_adder(n):\n return lambda x: x + n\n\ndef create_multiplier(n):\n return lambda x: x * n\n\ndef apply_functions(value,\n func_creator1, func_creator2, param1, param2):\n f1 = func_creator1(param1)\n f2 =\n func_creator2(param2)\n result1 = f1(value)\n result2 = f2(result1)\n return\n result2\n\ndef nested_higher_order(value):\n return apply_functions(value, create_adder,\n create_multiplier, 8, 4)\n\n# What is returned by nested_higher_order(3)?",
 }
}

```

```
 "answer": 44
 },
},
{
 "id": "MF-CC-S008-v007",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonFunctionalPipeline",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "functional_pipeline",
 "call_pattern": "composition",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Follow functional programming pipeline. What is returned by pipeline_result with data = 15?",
 "code": "def pipe(*functions):\n def pipeline(x):\n result = x\n for func in functions:\n result = func(result)\n return result\n return pipeline\n\ndef halve(x):\n return x // 2\n\ndef power_of_two(x):\n return x ** 2\n\ndef subtract_five(x):\n return x - 5\n\ndef pipeline_result(data):\n processor = pipe(halve, power_of_two, subtract_five)\n return processor(data)\n\n# What is returned by pipeline_result(15)?",
 "answer": 44
 }
},
{
 "id": "MF-CC-S008-v008",
 "metadata": {
 "name": "MultiFunc-CallChain-PythonAdvancedComposition",
 "category": "MultiFunc-Level",
 "subcategory": "Call chain",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 0,
 "variant_type": "advanced_composition",
 "call_pattern": "composition",
 "call_depth": "deep",
 "data_flow": "complex"
 },
 "task": {
 "description": "Analyze advanced function composition with conditional logic. What does advanced_compose return with x = 9?",
 "code": "def advanced_compose(x):\n if x < 0:\n return x\n else:\n return advanced_compose(x - 1) + 1\n\nprint(advanced_compose(9))",
 "answer": 45
 }
}
]
```

```

 "code": "def conditional_func(predicate, true_func, false_func):\n return\nlambda x: true_func(x) if predicate(x) else false_func(x)\n\ndef is_even(x):\n return x %\n2 == 0\n\ndef triple(x):\n return x * 3\n\ndef add_ten(x):\n return x + 10\n\ndef\nsquare(x):\n return x * x\n\ndef advanced_compose(x):\n conditional =\nconditional_func(is_even, triple, add_ten)\n step1 = conditional(x)\n step2 =\nsquare(step1)\n return step2\n\n# what is returned by advanced_compose(9)?",\n "answer": 361\n}\n}\n]\n"

```

## 4B - 参数传递推理 [ Parameter Passing ] (57)

# 参数传递推理变式生成提示词

## 任务目标

基于给定的参数传递推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对跨函数参数传递机制的理解能力，包括值传递、引用传递、指针传递等核心参数传递推理技能。

## 参数传递推理特征分析

参数传递推理关注函数调用时参数的传递方式和对原始数据的影响，重点测试模型对不同传递方式的语义理解、内存模型认知和数据修改效果的准确判断，强调对参数传递机制的深层理解能力。

## 关键要求

- \*\*描述语言\*\*：所有task描述必须使用英文
- \*\*变式数量\*\*：每个种子任务严格生成8-12个变式，确保数量充足
- \*\*答案唯一性\*\*：保证答案准确且唯一，避免歧义和多解情况
- \*\*传递机制聚焦\*\*：重点关注参数传递机制的分析，强调传递方式对数据的影响

## 变式生成维度

### 1. 值传递分析变式

- \*\*基本类型值传递变式\*\*：基本数据类型的值传递行为分析
- \*\*结构体值传递变式\*\*：结构体作为值传递的拷贝行为
- \*\*数组值传递变式\*\*：数组退化为指针的值传递分析
- \*\*联合体值传递变式\*\*：联合体的值传递和内存共享
- \*\*大对象值传递变式\*\*：大型对象值传递的性能和行为
- \*\*嵌套结构值传递变式\*\*：嵌套结构体的深拷贝行为

### 2. 引用传递分析变式

- \*\*引用参数修改变式\*\*：通过引用参数修改原始数据
- \*\*引用返回变式\*\*：函数返回引用的行为分析
- \*\*常量引用变式\*\*：常量引用参数的限制和行为
- \*\*引用别名变式\*\*：引用创建别名的效果分析
- \*\*临时对象引用变式\*\*：临时对象的引用绑定
- \*\*引用链式传递变式\*\*：引用在多层调用中的传递

### 3. 指针传递分析变式

- \*\*单级指针传递变式\*\*：一级指针的传递和解引用
- \*\*多级指针传递变式\*\*：二级、三级指针的复杂传递
- \*\*指针修改目标变式\*\*：通过指针修改指向的数据
- \*\*指针重新指向变式\*\*：在函数中重新设置指针指向
- \*\*空指针处理变式\*\*：空指针的传递和检查处理

- **函数指针传递变式**: 函数指针作为参数的传递

#### ### 4. 数组传递分析变式

- **一维数组传递变式**: 一维数组作为参数的传递分析
- **多维数组传递变式**: 多维数组的传递和访问
- **数组大小信息变式**: 数组大小信息的传递和使用
- **动态数组传递变式**: 动态分配数组的传递
- **数组切片传递变式**: 数组部分片段的传递
- **字符串数组传递变式**: 字符串数组的特殊传递

#### ### 5. 复合类型传递变式

- **结构体指针传递变式**: 结构体指针的传递和成员访问
- **结构体数组传递变式**: 结构体数组的传递分析
- **嵌套结构传递变式**: 包含指针的结构体传递
- **位字段结构传递变式**: 位字段结构体的传递行为
- **变长结构传递变式**: 变长结构体的传递处理
- **自引用结构传递变式**: 自引用结构体的传递

#### ### 6. 语言特定传递变式

- **Python对象引用变式**: Python对象引用的传递机制
- **可变不可变对象变式**: 可变和不可变对象的传递差异
- **深拷贝浅拷贝变式**: 深拷贝和浅拷贝的传递效果
- **共享引用变式**: 多个变量共享同一对象引用
- **垃圾回收影响变式**: 垃圾回收对引用传递的影响
- **弱引用传递变式**: 弱引用的传递和生命周期

#### ### 7. 内存模型分析变式

- **栈参数传递变式**: 栈上参数的传递和内存布局
- **堆对象传递变式**: 堆上对象的引用传递
- **静态数据传递变式**: 静态数据的传递和共享
- **线程局部传递变式**: 线程局部存储的参数传递
- **内存对齐影响变式**: 内存对齐对参数传递的影响
- **缓存效应变式**: 参数传递的缓存效应分析

#### ### 8. 传递优化分析变式

- **编译器优化变式**: 编译器对参数传递的优化
- **内联函数传递变式**: 内联函数中的参数传递
- **寄存器传递变式**: 寄存器传递参数的分析
- **调用约定变式**: 不同调用约定的参数传递
- **尾调用优化变式**: 尾调用优化对参数的影响
- **逃逸分析变式**: 逃逸分析对参数传递的优化

#### ### 9. 异常和错误处理变式

- **异常传递变式**: 异常情况下的参数传递
- **错误码传递变式**: 错误码作为参数的传递
- **资源管理变式**: **RAII**模式的参数传递
- **析构函数影响变式**: 析构函数对参数传递的影响
- **异常安全变式**: 异常安全的参数传递设计
- **错误传播变式**: 错误在参数传递中的传播

#### ### 10. 高级传递模式变式

- **完美转发变式**: 完美转发的参数传递机制
- **移动语义变式**: 移动语义的参数传递优化

- **模板参数传递变式**: 模板参数的传递和实例化
- **可变参数传递变式**: 可变参数列表的传递
- **泛型约束传递变式**: 泛型约束的参数传递
- **元编程传递变式**: 元编程中的参数传递

## ## 复杂度层次设计

### ### 简单传递分析 (Easy)

- 基本类型的值传递和简单指针传递
- 清晰的传递语义和修改效果
- 直观的内存模型和数据流
- 基础的参数传递概念

### ### 中等传递分析 (Medium)

- 结构体和数组的传递分析
- 包含一定复杂度的指针操作
- 中等复杂度的内存模型
- 需要理解传递机制的差异

### ### 复杂传递分析 (Hard)

- 多级指针和复杂数据结构传递
- 深层的内存模型理解
- 复杂的传递语义和副作用
- 涉及优化和特殊情况的分析

### ### 专家级传递分析 (Expert)

- 极复杂的传递场景和边界情况
- 高级语言特性的传递分析
- 需要深度系统知识的传递理解
- 涉及编译器和运行时的传递机制

## ## 生成策略

### ### 种子分析策略

1. **识别传递模式**: 分析种子任务的主要参数传递模式
2. **提取修改效果**: 识别参数传递对原始数据的修改效果
3. **确定内存语义**: 分析涉及的内存模型和语义
4. **评估复杂度层次**: 评估传递分析的复杂程度

### ### 变式设计原则

1. **传递机制导向**: 每个变式都应密切关注参数传递机制
2. **英文描述**: 所有task描述必须使用标准英文
3. **答案唯一**: 严格确保答案的准确性和唯一性
4. **数量保证**: 严格确保每个种子生成8-12个变式

### ### 质量保证

1. **传递语义验证**: 验证参数传递的语义正确性
2. **内存模型检查**: 确保内存模型分析的准确性
3. **答案唯一性验证**: 严格检查答案的唯一性和确定性
4. **英文质量保证**: 确保描述的英文表达准确清晰

## ## 输出格式要求

```

```json
[
  {
    "id": "MF-PP-S00X-v001",
    "metadata": {
      "name": "MultiFunc-ParameterPassing-VariantName",
      "category": "MultiFunc-Level",
      "subcategory": "Parameter Passing",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "passing_type": "value/reference/pointer/array",
      "data_type": "primitive/struct/array/complex",
      "modification_effect": "none/local/original/both"
    },
    "task": {
      "description": "English description of the parameter passing analysis task",
      "code": "Code containing parameter passing scenarios to analyze",
      "answer": "Unique and accurate parameter passing analysis result"
    }
  },
  {下一个变式...}
]

```

特殊字段说明

passing_type: 标识主要的参数传递类型

data_type: 标识传递的数据类型复杂度

modification_effect: 标识参数传递的修改效果范围

生成目标

为每个提供的种子任务严格生成8-12个参数传递变式，确保：

每种难度等级至少包含2-3个变式

涵盖至少**6-8**种不同的传递模式分析类型

包含不同的数据类型和传递复杂度

所有**task**描述使用标准英文表达

严格保证答案的准确性和唯一性

重点测试场景

传递方式识别：准确识别不同的参数传递方式

修改效果分析：正确分析参数传递对原始数据的影响

内存模型理解：理解不同传递方式的内存模型

生存期管理：分析参数在不同作用域的生存期

性能影响评估：评估不同传递方式的性能影响

语言特性理解：理解语言特定的传递机制

英文描述模板示例

"Analyze the parameter passing mechanism and determine the final value..."

"Trace the effect of value/reference passing on the original data..."

"Determine what changes occur to the parameter after function execution..."

"Calculate the final state of variables after pointer parameter modification..."

"Evaluate the impact of parameter passing on memory contents..."

"Analyze the behavior of complex data structures in parameter passing..."

答案唯一性保证策略

明确传递语义：确保参数传递的语义在目标语言中明确

具体数据值：使用具体的数据值，避免符号化结果

确定性修改：确保参数修改的效果是确定的

避免未定义行为：避免产生未定义行为的传递操作

清晰内存模型：确保内存模型和数据布局清晰

常见传递模式

基本值传递：基本类型的拷贝传递，不影响原始数据

指针传递：传递地址，可以修改指向的数据

引用传递：创建别名，直接操作原始数据

数组退化：数组作为参数时退化为指针

结构体拷贝：结构体的完整拷贝传递

浅拷贝引用：对象引用的浅拷贝传递

请基于此提示词，为给定的参数传递推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

[

{

```
    "id": "MF-PP-S001-V001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-BasicValueDouble",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "basic_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze basic value passing with double type. What is the value of original_val after function execution?",
        "code": "void modify_double(double val) {\n    val = val / 2.0 + 3.5;\n}\nprintf(\"Inside function: %.1f\\n\", val);\n\nint main() {\n    double original_val = 14.0;\n    modify_double(original_val);\n    printf(\"After function: %.1f\\n\", original_val);\n    return 0;\n}\n// What is the value of original_val after modify_double is called?",
        "answer": "14.0"
    }
},
{
    "id": "MF-PP-S001-V002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-CharValuePass",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "basic_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze basic value passing with char type. What is the value of original_val after function execution?",
        "code": "void modify_char(char val) {\n    val = val + 'A';\n}\nprintf(\"Inside function: %c\\n\", val);\n\nint main() {\n    char original_val = 'A';\n    modify_char(original_val);\n    printf(\"After function: %c\\n\", original_val);\n    return 0;\n}\n// What is the value of original_val after modify_char is called?",
        "answer": "B"
    }
}
```

}

{

```
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "char_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Trace character value passing behavior. What is the ASCII value of original_char after function call?",
        "code": "void process_char(char ch) {\n    ch = ch + 5;\n    printf(\"Inside function: %c (%d)\\n\", ch, ch);\n}\n\nint main() {\n    char original_char = 'A';\n    process_char(original_char);\n    printf(\"After function: %c (%d)\\n\", original_char, original_char);\n    return 0;\n}\n\n// what is the ASCII value of original_char after process_char is called?",
        "answer": "65"
    }
},
{
    "id": "MF-PP-S001-v003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-Multiplevalues",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "multiple_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze multiple parameter value passing. What is the sum of a and b after function execution?",
        "code": "void swap_values(int x, int y) {\n    int temp = x;\n    x = y;\n    y = temp;\n}\n\nprintf(\"Inside function: x=%d, y=%d\\n\", x, y);\n\nint main() {\n    int a = 15, b = 25;\n    swap_values(a, b);\n    printf(\"After function: a=%d, b=%d\\n\", a, b);\n    return 0;\n}\n\n// what is the sum of a and b after swap_values is called?",
        "answer": "40"
    }
},
{
    "id": "MF-PP-S001-v004",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-NestedCalls",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "nested_value_calls",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Track value passing through nested function calls. What is the final value of num?",
        "code": "void inner_modify(int val) {\n    val = val * 3;\n}\n\nvoid outer_modify(int val) {\n    val = val + 10;\n    inner_modify(val);\n}\n\nint main() {\n    int num = 8;\n    outer_modify(num);\n    printf(\"Final value: %d\\n\", num);\n    return 0;\n}\n\n// What is the final value of num after all function calls?",
        "answer": "8"
    }
},
{
    "id": "MF-PP-S001-V005",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-GlobalInteraction",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "global_value_interaction",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "local"
    },
    "task": {
        "description": "Analyze value passing with global variable interaction. What is the value of local_var after function call?",
        "code": "int global_var = 100;\n\nvoid modify_with_global(int local_param) {\n    local_param = global_var + local_param;\n    global_var = local_param - 50;\n}\n\nint main() {\n    int local_var = 30;\n    modify_with_global(local_var);\n    printf(\"local_var: %d, global_var: %d\\n\", local_var, global_var);\n    return 0;\n}\n\n// What is the value of local_var after modify_with_global is called?",
        "answer": "30"
    }
},
{
    "id": "MF-PP-S001-V006",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ReturnValue",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "nested_value_calls",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze value passing with global variable interaction. What is the value of local_var after function call?",
        "code": "int global_var = 100;\n\nvoid modify_with_global(int local_param) {\n    local_param = global_var + local_param;\n    global_var = local_param - 50;\n}\n\nint main() {\n    int local_var = 30;\n    modify_with_global(local_var);\n    printf(\"local_var: %d, global_var: %d\\n\", local_var, global_var);\n    return 0;\n}\n\n// What is the value of local_var after modify_with_global is called?",
        "answer": "30"
    }
}
```

```

    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "return_value_passing",
    "passing_type": "value",
    "data_type": "primitive",
    "modification_effect": "none"
  },
  "task": {
    "description": "Examine value passing with return mechanism. What is the final value assigned to result?",
    "code": "int transform_value(int input) {\n    input = input * 4 + 7;\n\nreturn input;\n}\n\nint main() {\n    int original = 12;\n    int result =\n    transform_value(original);\n    printf(\"original: %d, result: %d\\n\", original, result);\n\n    return 0;\n}\n\n// What is the final value assigned to result?",
    "answer": "55"
  }
},
{
  "id": "MF-PP-S001-V007",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-ConditionalModify",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "conditional_value_passing",
    "passing_type": "value",
    "data_type": "primitive",
    "modification_effect": "none"
  },
  "task": {
    "description": "Analyze conditional value modification in function. What is the value of test_val after function execution?",
    "code": "void conditional_modify(int val) {\n    if (val > 20) {\n        val =\n        val / 2;\n    } else {\n        val = val * 3;\n    }\n    printf(\"Inside function: %d\\n\", val);\n}\n\nint main() {\n    int test_val = 18;\n    conditional_modify(test_val);\n    printf(\"After function: %d\\n\", test_val);\n\n    return 0;\n}\n\n// What is the value of test_val after conditional_modify is called?",
    "answer": "18"
  }
},
{
  "id": "MF-PP-S001-V008",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-LoopModification",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "loop_value_passing",
    "passing_type": "value",
    "data_type": "primitive",
    "modification_effect": "none"
  }
}

```

```
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "loop_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Track value passing with loop modification. What is the value of counter after function call?",
        "code": "void loop_modify(int count) {\n    for (int i = 0; i < 3; i++) {\n        count += i * 2;\n    }\n    printf(\"Inside function: %d\\n\", count);\n}\nint main()\n{\n    int counter = 10;\n    loop_modify(counter);\n    printf(\"After function: %d\\n\", counter);\n    return 0;\n}// what is the value of counter after loop_modify is called?",
        "answer": "10"
    }
},
{
    "id": "MF-PP-S001-v009",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-RecursiveValue",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "recursive_value_passing",
        "passing_type": "value",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze recursive value passing behavior. What is the value of initial_num after function execution?",
        "code": "void recursive_modify(int n, int depth) {\n    if (depth > 0) {\n        n = n + depth;\n        recursive_modify(n, depth - 1);\n    }\n    printf(\"Depth %d: n = %d\\n\", depth, n);\n}\nint main()\n{\n    int initial_num = 5;\n    recursive_modify(initial_num, 2);\n    printf(\"After recursion: %d\\n\", initial_num);\n    return 0;\n}// what is the value of initial_num after recursive_modify is called?",
        "answer": "5"
    }
},
{
    "id": "MF-PP-S001-v010",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-EnumValue",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
```

```

"source": "Generated",
"language": "c",
"difficulty": "medium",
"intervention": 1,
"variant_type": "enum_value_passing",
"passing_type": "value",
"data_type": "primitive",
"modification_effect": "none"
},
"task": {
"description": "Examine enum value passing mechanism. What is the numeric value of status after function call?",
"code": "typedef enum { IDLE = 0, RUNNING = 1, STOPPED = 2, ERROR = 3 } Status;\nvoid change_status(Status s) {\n    s = ERROR;\n    printf(\"Inside function: %d\\n\", s);\n}\nint main() {\n    Status status = RUNNING;\n    change_status(status);\n    printf(\"After function: %d\\n\", status);\n    return 0;\n}\n// What is the numeric value of status after change_status is called?",
"answer": "1"
},
{
"id": "MF-PP-S002-V001",
"metadata": {
"name": "MultiFunc-ParameterPassing-SimpleStructValue",
"category": "MultiFunc-Level",
"subcategory": "Parameter Passing",
"type": "variant",
"source": "Generated",
"language": "c",
"difficulty": "medium",
"intervention": 1,
"variant_type": "simple_struct_value",
"passing_type": "value",
"data_type": "struct",
"modification_effect": "none"
},
"task": {
"description": "Analyze simple struct value passing. What is the value of rect.width after function execution?",
"code": "typedef struct {\n    int width;\n    int height;\n} Rectangle;\n\nvoid resize_rectangle(Rectangle r) {\n    r.width *= 2;\n    r.height += 10;\n}\n\nint main() {\n    Rectangle rect = {15, 8};\n    resize_rectangle(rect);\n    printf(\"width: %d, height: %d\\n\", rect.width, rect.height);\n    return 0;\n}\n// What is the value of rect.width after resize_rectangle is called?",
"answer": "15"
},
{
"id": "MF-PP-S002-V002",
"metadata": {
"name": "MultiFunc-ParameterPassing-NestedStructValue",
"category": "MultiFunc-Level",
"subcategory": "Parameter Passing",

```

```
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_struct_value",
        "passing_type": "value",
        "data_type": "struct",
        "modification_effect": "none"
    },
    "task": {
        "description": "Track nested struct value passing. What is the value of person.address.zip after function call?",
        "code": "typedef struct {\n    int zip;\n    char city[20];\n} Address;\n\ntypedef struct {\n    char name[30];\n    int age;\n    Address address;\n} Person;\n\nvoid update_person(Person p) {\n    p.age += 5;\n    p.address.zip =\n12345;\n}\n\nint main() {\n    Person person = {"John", 25, {67890, \"NYC\"}};\n    update_person(person);\n    printf(\"age: %d, zip: %d\\n\", person.age,\n    person.address.zip);\n    return 0;\n}\n\n// What is the value of person.address.zip after update_person is called?",
        "answer": "67890"
    }
},
{
    "id": "MF-PP-S002-V003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ArrayInStruct",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "array_in_struct_value",
        "passing_type": "value",
        "data_type": "struct",
        "modification_effect": "none"
    },
    "task": {
        "description": "Examine struct with array member value passing. What is the value of scores.values[1] after function execution?",
        "code": "typedef struct {\n    int values[4];\n    int count;\n} ScoreList;\n\nvoid modify_scores(ScoreList s) {\n    s.values[0] = 95;\n    s.values[1] = 87;\n    s.count = 2;\n}\n\nint main() {\n    ScoreList scores = {{75, 82, 90, 88}, 4};\n    modify_scores(scores);\n    printf(\"values[1]: %d, count: %d\\n\", scores.values[1],\n    scores.count);\n    return 0;\n}\n\n// What is the value of scores.values[1] after modify_scores is called?",
        "answer": "82"
    }
},
{
    "id": "MF-PP-S002-V004",
```

```

"metadata": {
    "name": "MultiFunc-ParameterPassing-UnionValue",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "union_value_passing",
    "passing_type": "value",
    "data_type": "struct",
    "modification_effect": "none"
},
"task": {
    "description": "Analyze union value passing behavior. What is the integer value of data.i after function call?",
    "code": "typedef union {\n    int i;\n    float f;\n    char c[4];\n} Data;\n\nvoid modify_union(Data d) {\n    d.i = 0x12345678;\n    d.f = 3.14f;\n}\n\nint main() {\n    Data data;\n    data.i = 100;\n    modify_union(data);\n    printf(\"int value: %d\\n\", data.i);\n    return 0;\n} // what is the integer value of data.i after modify_union is called?",
    "answer": "100"
},
{
    "id": "MF-PP-S002-V005",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-StructReturn",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "struct_return_value",
        "passing_type": "value",
        "data_type": "struct",
        "modification_effect": "Local"
},
    "task": {
        "description": "Track struct return value behavior. What is the y coordinate of the returned point?",
        "code": "typedef struct {\n    int x;\n    int y;\n} Point;\n\nPoint create_point(Point base) {\n    base.x = base.x + 10;\n    base.y = base.y * 2;\n    return base;\n}\n\nint main() {\n    Point p1 = {5, 8};\n    Point p2 = create_point(p1);\n    printf(\"p1: (%d,%d), p2: (%d,%d)\\n\", p1.x, p1.y, p2.x, p2.y);\n    return 0;\n} // what is the y coordinate of p2 after create_point is called?",
        "answer": "16"
},
{
}

```

```

"id": "MF-PP-S002-V006",
"metadata": {
    "name": "MultiFunc-ParameterPassing-StructChain",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "struct_chain_passing",
    "passing_type": "value",
    "data_type": "struct",
    "modification_effect": "none"
},
"task": {
    "description": "Analyze chained struct value passing. What is the radius of circle after all function calls?",
    "code": "typedef struct {\n    double radius;\n    double area;\n} Circle;\n\nvoid scale_circle(Circle c) {\n    c.radius *= 1.5;\n    c.area = 3.14159 * c.radius * c.radius;\n}\n\nvoid process_circle(Circle c) {\n    scale_circle(c);\n    c.radius += 2.0;\n}\n\nint main() {\n    Circle circle = {5.0, 78.54};\n    process_circle(circle);\n    printf(\"radius: %.1f\\n\", circle.radius);\n    return 0;\n}\n\n// what is the radius of circle after process_circle is called?",
    "answer": "5.0"
},
{
    "id": "MF-PP-S002-V007",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-BitfieldStruct",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "bitfield_struct_value",
        "passing_type": "value",
        "data_type": "struct",
        "modification_effect": "none"
},
    "task": {
        "description": "Examine bitfield struct value passing. What is the value of flags.enabled after function execution?",
        "code": "typedef struct {\n    unsigned int enabled : 1;\n    unsigned int priority : 3;\n    unsigned int reserved : 4;\n} Flags;\n\nvoid modify_flags(Flags f) {\n    f.enabled = 0;\n    f.priority = 7;\n    f.reserved = 15;\n}\n\nint main() {\n    Flags flags = {1, 2, 0};\n    modify_flags(flags);\n    printf(\"enabled: %u, priority: %u\\n\", flags.enabled, flags.priority);\n    return 0;\n}\n\n// what is the value of flags.enabled after modify_flags is called?",
        "answer": "1"
}

```

```
        },
    },
    {
        "id": "MF-PP-S002-V008",
        "metadata": {
            "name": "MultiFunc-ParameterPassing-ConstStruct",
            "category": "MultiFunc-Level",
            "subcategory": "Parameter Passing",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "const_struct_value",
            "passing_type": "value",
            "data_type": "struct",
            "modification_effect": "none"
        },
        "task": {
            "description": "Analyze const struct parameter passing. What is the total value computed by the function?",
            "code": "typedef struct {\n    int a;\n    int b;\n    int c;\n} Triple;\n\nint compute_sum(const Triple t) {\n    // t.a = 100; // Would cause compilation error\n    return t.a + t.b + t.c;\n}\n\nint main() {\n    Triple triple = {10, 20, 30};\n    int result = compute_sum(triple);\n    printf(\"result: %d\\n\", result);\n    return 0;\n}\n\n// What value is returned by compute_sum?",
            "answer": "60"
        }
    },
    {
        "id": "MF-PP-S002-V009",
        "metadata": {
            "name": "MultiFunc-ParameterPassing-LargeStruct",
            "category": "MultiFunc-Level",
            "subcategory": "Parameter Passing",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "large_struct_value",
            "passing_type": "value",
            "data_type": "struct",
            "modification_effect": "none"
        },
        "task": {
            "description": "Track large struct value passing. What is the value of matrix.data[2][1] after function call?"
        }
    }
}
```

```

        "code": "typedef struct {\n    int data[3][3];\n    int rows;\n    int cols;\n} Matrix;\n\nvoid process_matrix(Matrix m) {\n    for (int i = 0; i < 3; i++) {\n        for (int j = 0; j < 3; j++) {\n            m.data[i][j] *= 2;\n        }\n    }\n}\n\nint main()\n{\n    Matrix matrix = {{{1,2,3},{4,5,6},{7,8,9}}, 3, 3};\n    process_matrix(matrix);\n    printf(\"matrix.data[2][1]: %d\\n\", matrix.data[2][1]);\n    return 0;\n}\n\n// What is the value of matrix.data[2][1] after process_matrix is called?",\n    "answer": "8"
}

},
{
    "id": "MF-PP-S002-V010",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-StructAssignment",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "struct_assignment_value",
        "passing_type": "value",
        "data_type": "struct",
        "modification_effect": "none"
    },
    "task": {
        "description": "Examine struct assignment in value passing. What is the final value of original.value?",\n        "code": "typedef struct {\n    int value;\n    char tag;\n} Item;\n\nvoid reassign_item(Item item) {\n    Item new_item = {999, 'z'};\n    item = new_item;\n}\n\nprintf(\"Inside function: %d\\n\", item.value);\n\nint main() {\n    Item original = {42, 'A'};\n    reassign_item(original);\n    printf(\"After function: %d\\n\", original.value);\n    return 0;\n}\n\n// What is the final value of original.value after reassign_item is called?",\n        "answer": "42"
    }
},
{
    "id": "MF-PP-S003-V001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-BasicPointerModify",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "basic_pointer_modify",
        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "original"
    },
}

```

```

"task": {
    "description": "Analyze basic pointer parameter modification. What is the final value of num?",
    "code": "void increment_value(int *ptr) {\n    *ptr = *ptr + 15;\n}\n\nint main() {\n    int num = 30;\n    increment_value(&num);\n    printf(\"After function: %d\\n\", num);\n    return 0;\n}\n\n// What is the final value of num after increment_value is called?",
    "answer": "45"
},
{
    "id": "MF-PP-S003-V002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-StringModification",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "string_pointer_modify",
        "passing_type": "pointer",
        "data_type": "array",
        "modification_effect": "original"
    },
    "task": {
        "description": "Track string modification through pointer. What is the second character of str after function call?",
        "code": "void modify_string(char *s) {\n    s[0] = 'H';\n    s[1] = 'i';\n    s[2] = '!';\n    s[3] = '\\0';\n}\n\nint main() {\n    char str[10] = \"Hello\";\n    modify_string(str);\n    printf(\"Modified string: %s\\n\", str);\n    return 0;\n}\n\n// What is the second character (index 1) of str after modify_string is called?",
        "answer": "i"
    }
},
{
    "id": "MF-PP-S003-V003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerArithmetic",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "pointer_arithmetic_modify",
        "passing_type": "pointer",
        "data_type": "array",
        "modification_effect": "original"
    },
    "task": {

```

```

    "description": "Examine pointer arithmetic in parameter passing. What is the value of numbers[3] after function execution?",
    "code": "void shift_values(int *arr) {\n    *(arr + 0) += 5;\n    *(arr + 1) += 10;\n    *(arr + 2) += 15;\n    *(arr + 3) += 20;\n}\n\nint main() {\n    int numbers[5] = {1, 2, 3, 4, 5};\n    shift_values(numbers);\n    printf(\"numbers[3]: %d\\n\", numbers[3]);\n    return 0;\n}\n\n// What is the value of numbers[3] after shift_values is called?",
    "answer": "24"
}
},
{
    "id": "MF-PP-S003-V004",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-StructPointer",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "struct_pointer_modify",
        "passing_type": "pointer",
        "data_type": "struct",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze struct pointer modification. What is the age field of person after function call?",
        "code": "typedef struct {\n    char name[20];\n    int age;\n    double salary;\n} Employee;\n\nvoid update_employee(Employee *emp) {\n    emp->age += 3;\n    emp->salary *= 1.1;\n}\n\nint main() {\n    Employee person = {"Alice", 28, 50000.0};\n    update_employee(&person);\n    printf(\"age: %d, salary: %.0f\\n\", person.age, person.salary);\n    return 0;\n}\n\n// What is the age field of person after update_employee is called?",
        "answer": "31"
    }
},
{
    "id": "MF-PP-S003-V005",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-MultiDimArray",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "multidim_array_pointer",
        "passing_type": "pointer",
        "data_type": "array",
        "modification_effect": "original"
    }
}

```

```

},
"task": {
    "description": "Track multidimensional array pointer modification. What is the value of grid[1][2] after function execution?",
    "code": "void fill_grid(int grid[][], int rows) {\n    for (int i = 0; i < rows; i++) {\n        for (int j = 0; j < 3; j++) {\n            grid[i][j] = i * 3 + j + 1;\n        }\n    }\n}\n\nint main() {\n    int grid[2][3] = {{0, 0, 0}, {0, 0, 0}};\n    fill_grid(grid, 2);\n    printf(\"grid[1][2]: %d\\n\", grid[1][2]);\n    return 0;\n}\n\n// what is the value of grid[1][2] after fill_grid is called?",
    "answer": "6"
},
},
{
    "id": "MF-PP-S003-V006",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-NullPointerCheck",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "null_pointer_handling",
        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze null pointer handling in parameter passing. What value is returned by the function?",
        "code": "int safe_increment(int *ptr) {\n    if (ptr != NULL) {\n        *ptr += 10;\n    }\n    return *ptr;\n}\n\nint main() {\n    int *null_ptr = NULL;\n    int result = safe_increment(null_ptr);\n    printf(\"Result: %d\\n\", result);\n    return 0;\n}\n\n// what value is returned by safe_increment when passed a NULL pointer?",
        "answer": "-1"
},
},
{
    "id": "MF-PP-S003-V007",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-Pointerswap",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "pointer_value_swap",
        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "original"
    }
}

```

```

},
"task": {
  "description": "Examine pointer-based value swapping. What is the value of b after function call?",
  "code": "void swap_values(int *x, int *y) {\n    int temp = *x;\n    *x = *y;\n    *y = temp;\n}\n\nint main() {\n    int a = 15, b = 25;\n    swap_values(&a, &b);\n    printf(\"a: %d, b: %d\\n\", a, b);\n    return 0;\n}\n\n// What is the value of b after swap_values is called?",
  "answer": "15"
},
{
  "id": "MF-PP-S003-V008",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-ConstPointer",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "const_pointer_param",
    "passing_type": "pointer",
    "data_type": "array",
    "modification_effect": "none"
  },
  "task": {
    "description": "Analyze const pointer parameter behavior. What is the sum computed by the function?",
    "code": "int calculate_sum(const int *arr, int size) {\n    int sum = 0;\n    for (int i = 0; i < size; i++) {\n        sum += arr[i];\n        // arr[i] = 0; // would cause compilation error\n    }\n    return sum;\n}\n\nint main() {\n    int data[4] = {10, 20, 30, 40};\n    int result = calculate_sum(data, 4);\n    printf(\"sum: %d\\n\", result);\n    return 0;\n}\n\n// What is the sum returned by calculate_sum?",
    "answer": "100"
},
{
  "id": "MF-PP-S003-V009",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-FunctionPointer",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "function_pointer_param",
    "passing_type": "pointer",
    "data_type": "complex",
    "modification_effect": "local"
  }
}

```

```

},
"task": {
    "description": "Track function pointer parameter usage. What value is returned by apply_operation?",
    "code": "int multiply_by_three(int x) {\n    return x * 3;\n}\n\nint add_ten(int x) {\n    return x + 10;\n}\n\nint apply_operation(int value, int (*func)(int)) {\n    return func(value);\n}\n\nint main() {\n    int result = apply_operation(7, multiply_by_three);\n    printf(\"result: %d\\n\", result);\n    return 0;\n}\n\n// what value is returned by apply_operation(7, multiply_by_three)?",
    "answer": "21"
},
{
    "id": "MF-PP-S003-V010",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-VoidPointer",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "void_pointer_param",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine void pointer parameter casting. What is the final value of number after function call?",
        "code": "void generic_increment(void *ptr, char type) {\n    if (type == 'i')\n        int *iptr = (int *)ptr;\n        *iptr += 5;\n    } else if (type == 'f') {\n        float *fptr = (float *)ptr;\n        *fptr += 2.5f;\n    }\n}\n\nint main() {\n    int number = 20;\n    generic_increment(&number, 'i');\n    printf(\"number: %d\\n\", number);\n    return 0;\n}\n\n// what is the final value of number after generic_increment is called?",
        "answer": "25"
    }
},
{
    "id": "MF-PP-S004-V001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-SimpleDoublePointer",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "simple_double_pointer",
        "passing_type": "pointer",
        "data_type": "primitive",
    }
}

```

```
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze simple double pointer manipulation. What is the final value pointed to by p?",
        "code": "void modify_through_double_ptr(int **dptr) {\n    **dptr = **dptr * 2 + 5;\n}\n\nint main() {\n    int value = 12;\n    int *p = &value;\n    modify_through_double_ptr(&p);\n    printf(\"Final value: %d\\n\", *p);\n    return 0;\n}\n\n// What is the final value pointed to by p after modify_through_double_ptr is called?",

        "answer": "29"
    }
},
{
    "id": "MF-PP-S004-V002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerReallocation",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "pointer_reallocation",
        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "both"
    },
    "task": {
        "description": "Track pointer reallocation through double pointer. What value does ptr point to after function call?",
        "code": "void reallocate_pointer(int **ptr) {\n    free(*ptr);\n    *ptr = malloc(sizeof(int));\n    **ptr = 88;\n}\n\nint main() {\n    int *ptr = malloc(sizeof(int));\n    *ptr = 42;\n    reallocate_pointer(&ptr);\n    int result = *ptr;\n    free(ptr);\n    return result;\n}\n\n// What value does ptr point to after reallocate_pointer is called?",

        "answer": "88"
    }
},
{
    "id": "MF-PP-S004-V003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-TriplePointer",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "triple_pointer_manip",
        "passing_type": "pointer",
        "modification_effect": "original"
    }
}
```

```
        "data_type": "primitive",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine triple pointer manipulation. What is the final value of the integer?",
        "code": "void modify_triple_ptr(int ***tptr) {\n    ***tptr = ***tptr +\n100;\n}\n\nint main() {\n    int value = 15;\n    int *p1 = &value;\n    int **p2 = &p1;\n    int ***p3 = &p2;\n    modify_triple_ptr(p3);\n    printf(\"Final value: %d\\n\", value);\n}\n\n// What is the final value of the integer after modify_triple_ptr is called?",
        "answer": "115"
    }
},
{
    "id": "MF-PP-S004-V004",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerArray",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "pointer_array_manip",
        "passing_type": "pointer",
        "data_type": "array",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze array of pointers manipulation. What is the value pointed to by ptrs[1] after function execution?",
        "code": "void update_pointer_array(int **ptrs, int count) {\n    for (int i = 0; i < count; i++) {\n        *(ptrs[i]) += i * 10;\n    }\n}\n\nint main() {\n    int a = 5, b = 10, c = 15;\n    int *ptrs[3] = {&a, &b, &c};\n    update_pointer_array(ptrs, 3);\n    printf(\"b = %d\\n\", b);\n}\n\n// What is the value pointed to by ptrs[1] (i.e., value of b) after update_pointer_array is called?",
        "answer": "20"
    }
},
{
    "id": "MF-PP-S004-V005",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerChain",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "pointer_chain_modify",
        "passing_type": "pointer"
    }
}
```

```

        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "both"
    },
    "task": {
        "description": "Track complex pointer chain modification. What is the final value after all operations?",
        "code": "void redirect_pointer(int **ptr, int *new_target) {\n    *ptr = new_target;\n}\nvoid modify_target(int **ptr) {\n    **ptr *= 3;\n}\nint main() {\n    int original = 10;\n    int replacement = 20;\n    int *p = &original;\n    modify_target(&p);\n    redirect_pointer(&p, &replacement);\n    modify_target(&p);\n    printf(\"replacement = %d\\n\", replacement);\n    return 0;\n}\n// What is the final value of replacement after all operations?",
        "answer": "60"
    }
},
{
    "id": "MF-PP-S004-V006",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-DynamicStruct",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "dynamic_struct_pointer",
        "passing_type": "pointer",
        "data_type": "struct",
        "modification_effect": "both"
    },
    "task": {
        "description": "Examine dynamic struct allocation through double pointer. What is the id field after function execution?",
        "code": "typedef struct {\n    int id;\n    char name[20];\n} Record;\nvoid create_record(Record **rec, int id) {\n    *rec = malloc(sizeof(Record));\n    (*rec)->id = id * 2;\n    strcpy((*rec)->name, \"Dynamic\");\n}\nint main() {\n    Record *record = NULL;\n    create_record(&record, 25);\n    int result = record->id;\n    free(record);\n    return result;\n}\n// What is the id field of the dynamically created record?",
        "answer": "50"
    }
},
{
    "id": "MF-PP-S004-V007",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-NestedPointers",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
    }
}

```

```

    "intervention": 3,
    "variant_type": "nested_pointer_struct",
    "passing_type": "pointer",
    "data_type": "struct",
    "modification_effect": "original"
},
"task": {
    "description": "Analyze nested pointer structures. What is the final value of node->data after function call?",
    "code": "typedef struct Node {\n    int data;\n    struct Node *next;\n} Node;\n\nvoid update_linked_node(Node **head) {\n    (*head)->data += 50;\n    if ((*head)->next != NULL) {\n        (*head)->next->data *= 2;\n    }\n}\n\nint main() {\n    Node node2 = {20, NULL};\n    Node node1 = {10, &node2};\n    Node *head = &node1;\n\n    update_linked_node(&head);\n\n    printf(\"node1.data = %d\\n\", head->data);\n\n    return 0;\n}\n\n// what is the final value of node1.data (head->data) after update_linked_node is called?",
    "answer": "60"
},
{
    "id": "MF-PP-S004-v008",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ConditionalAllocation",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "conditional_allocation",
        "passing_type": "pointer",
        "data_type": "primitive",
        "modification_effect": "both"
    },
    "task": {
        "description": "Track conditional memory allocation. What value is returned by the test function?",
        "code": "int allocate_if_needed(int **ptr, int threshold) {\n    if (*ptr ==\n        NULL && threshold > 50) {\n        *ptr = malloc(sizeof(int));\n        **ptr = threshold *\n        2;\n        return **ptr;\n    } else if (*ptr != NULL) {\n        **ptr += 10;\n        return **ptr;\n    }\n    return -1;\n}\n\nint test_conditional() {\n    int *p = NULL;\n    int result = allocate_if_needed(&p, 60);\n    if (p != NULL) free(p);\n    return result;\n}\n\n// what value is returned by test_conditional()?",
        "answer": "120"
    }
},
{
    "id": "MF-PP-S005-v001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ListAppendExtend",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "list_append_extend",
        "passing_type": "pointer",
        "data_type": "list",
        "modification_effect": "original"
    }
}

```

```
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "list_mutation_methods",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze list mutation through multiple methods. What is the length of my_data after function execution?",
        "code": "def manipulate_list(data):\n    data.append(50)\n    data.insert(1, 25)\n    data.extend([75, 100])\n    data.remove(10)\n\ndef process_data():\n    my_data = [10, 20, 30]\n    manipulate_list(my_data)\n    return len(my_data)\n\n# What is returned by process_data()?",
        "answer": "6"
    }
},
{
    "id": "MF-PP-S005-V002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-DictModification",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "dict_reference_modify",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Track dictionary modification through reference. What is the value of scores['math'] after function call?",
        "code": "def update_scores(score_dict):\n    score_dict['math'] = score_dict['math'] + 15\n    score_dict['science'] = 95\n    score_dict.pop('english', None)\n\ndef test_dict_update():\n    scores = {'math': 80, 'english': 75, 'history': 90}\n    update_scores(scores)\n    return scores['math']\n\n# What is returned by test_dict_update()?",
        "answer": "95"
    }
},
{
    "id": "MF-PP-S005-V003",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-NestedListModify",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "list_mutation_modify",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze list mutation through multiple methods. What is the length of my_data after function execution?",
        "code": "def manipulate_list(data):\n    data.append(50)\n    data.insert(1, 25)\n    data.extend([75, 100])\n    data.remove(10)\n\ndef process_data():\n    my_data = [10, 20, 30]\n    manipulate_list(my_data)\n    return len(my_data)\n\n# What is returned by process_data()?",
        "answer": "6"
    }
}
```

```
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_list_reference",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine nested list modification behavior. What is the value of matrix[1][0] after function execution?",
        "code": "def modify_matrix(mat):\n    mat[0][1] = 99\n    mat[1].append(44)\nmat.append([77, 88])\n\ndef test_nested_lists():\n    matrix = [[1, 2, 3], [4, 5, 6]]\n    modify_matrix(matrix)\n    return matrix[1][0]\n\n# What is returned by test_nested_lists()?",
        "answer": "4"
    }
},
{
    "id": "MF-PP-S005-V004",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-SetOperations",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "set_reference_modify",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze set modification through reference passing. What is the size of numbers after function call?",
        "code": "def modify_set(s):\n    s.add(100)\n    s.discard(20)\n    s.update([200, 300])\n    s.remove(10)\n\ndef test_set_operations():\n    numbers = {10, 20, 30, 40}\n    modify_set(numbers)\n    return len(numbers)\n\n# What is returned by test_set_operations()?",
        "answer": "5"
    }
},
{
    "id": "MF-PP-S005-V005",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ObjectAttribute",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
```

```
"source": "Generated",
"language": "python",
"difficulty": "hard",
"intervention": 2,
"variant_type": "object_attribute_modify",
"passing_type": "reference",
"data_type": "complex",
"modification_effect": "original"
},
"task": {
    "description": "Track object attribute modification through reference. What is the final value of person.age?",
    "code": "class Person:\n    def __init__(self, name, age):\n        self.name = name\n        self.age = age\n    def update_person(p):\n        p.age += 5\n        p.name = p.name.upper()\n        p.city = \"New York\"\n    def test_object_modification():\n        person = Person(\"alice\", 25)\n        update_person(person)\n        return person.age\n\n# What is returned by test_object_modification()?",
    "answer": "30"
},
{
    "id": "MF-PP-S005-V006",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-StringImmutable",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "immutable_string_reference",
        "passing_type": "reference",
        "data_type": "primitive",
        "modification_effect": "none"
    },
    "task": {
        "description": "Examine immutable string behavior in parameter passing. What is the final value of text?",
        "code": "def modify_string(s):\n    s = s.upper()\n    s += \" MODIFIED\"\n\nreturn s\n\ndef test_string_immutability():\n    text = \"hello world\"\n    result = modify_string(text)\n    return len(result)\n\n# What is returned by test_string_immutability()?",
        "answer": "11"
    }
},
{
    "id": "MF-PP-S005-V007",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-TupleImmutable",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
```

```
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "immutable_tuple_reference",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze tuple immutability in reference passing. What is the first element of coordinates after function call?",
        "code": "def try_modify_tuple(t):\n    # t[0] = 999 # would raise TypeError\n    t = (999, 888, 777)\n    return t\n\ndef test_tuple_immutability():\n    coordinates = (10,\n    20, 30)\n    new_tuple = try_modify_tuple(coordinates)\n    return coordinates[0]\n\n# What is returned by test_tuple_immutability()?",
        "answer": "10"
    }
},
{
    "id": "MF-PP-S005-v008",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ListSlicing",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "list_slice_modification",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Track list slice assignment effects. What is the length of numbers after function execution?",
        "code": "def modify_slice(lst):\n    lst[1:3] = [99, 88, 77]\n    lst[:2] = [111, 222]\n\ndef test_slice_modification():\n    numbers = [1, 2, 3, 4, 5]\n    modify_slice(numbers)\n    return len(numbers)\n\n# What is returned by test_slice_modification()?",
        "answer": "5"
    }
},
{
    "id": "MF-PP-S005-v009",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ListClear",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "list_clear",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "none"
    },
    "task": {
        "description": "Analyze list clearing in reference passing. What is the length of numbers after function execution?",
        "code": "def clear_list(lst):\n    lst.clear()\n\ndef test_list_clear():\n    numbers = [1, 2, 3, 4, 5]\n    clear_list(numbers)\n    return len(numbers)\n\n# What is returned by test_list_clear()?",
        "answer": "0"
    }
}
```

```
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "list_clear_reference",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine list clearing through reference. What is the length of items after function call?",
        "code": "def process_and_clear(lst):\n    total = sum(lst)\n    lst.clear()\n    lst.append(total)\n    return total\n\ndef test_list_clear():\n    items = [10, 20, 30, 40]\n    result = process_and_clear(items)\n    return len(items)\n\n# What is returned by test_list_clear()?",
        "answer": "1"
    }
},
{
    "id": "MF-PP-S005-V010",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-SharedReference",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "shared_reference_modify",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze shared reference modification effects. What is the sum of all elements in list_b after function execution?",
        "code": "def modify_shared_list(lst):\n    lst.extend([100, 200])\n    lst[0] = 999\n\ndef test_shared_reference():\n    list_a = [1, 2, 3]\n    list_b = list_a # Same reference\n    modify_shared_list(list_a)\n    return sum(list_b)\n\n# What is returned by test_shared_reference()?",
        "answer": "1304"
    }
},
{
    "id": "MF-PP-S006-V001",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ArrayStructMixed",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "array_struct_mixed",
        "passing_type": "reference",
        "data_type": "complex",
        "modification_effect": "original"
    }
}
```

```

        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "array_struct_mixed",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze mixed array and struct parameter passing. What is the final sum computed?",
        "code": "typedef struct {\n    int values[3];\n    int multiplier;\n} NumberSet;\n\nvoid scale_numbers(NumberSet *ns, int factor) {\n    ns->multiplier *= factor;\n    for (int i = 0; i < 3; i++) {\n        ns->values[i] *= ns->multiplier;\n    }\n}\n\nint calculate_total(NumberSet set) {\n    int sum = 0;\n    for (int i = 0; i < 3; i++) {\n        sum += set.values[i];\n    }\n    return sum;\n}\n\nint test_mixed_operations() {\n    NumberSet ns = {{2, 4, 6}, 3};\n    scale_numbers(&ns, 2);\n    return calculate_total(ns);\n}\n\n// What value is returned by test_mixed_operations()?",
        "answer": "72"
    }
},
{
    "id": "MF-PP-S006-V002",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-PointerStructArray",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "pointer_struct_array",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Track pointer struct array manipulation. What is the score of students[1] after function execution?",
        "code": "typedef struct {\n    char name[20];\n    int score;\n} Student;\n\nvoid update_students(Student *students, int count) {\n    for (int i = 0; i < count; i++) {\n        students[i].score += 10;\n        if (students[i].grades != NULL) {\n            students[i].grades[0] += 5;\n        }\n    }\n}\n\nint test_student_update() {\n    int grades1[] = {85, 90};\n    int grades2[] = {75, 80};\n    Student students[2] = {{\"Alice\", 80, grades1}, {"Bob", 70, grades2}};\n    update_students(students, 2);\n    return students[1].score;\n}\n\n// What value is returned by test_student_update()?",
        "answer": "80"
    }
},
{
    "id": "MF-PP-S006-V003",
    "metadata": {

```

```

        "name": "MultiFunc-ParameterPassing-RecursiveStruct",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "recursive_struct_modify",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine recursive struct modification. What is the final value of root node data?",
        "code": "typedef struct TreeNode {\n    int data;\n    struct TreeNode *left;\n    struct TreeNode *right;\n} TreeNode;\n\nvoid increment_tree(TreeNode *node, int increment) {\n    if (node == NULL) return;\n    node->data += increment;\n    increment_tree(node->left, increment + 1);\n    increment_tree(node->right, increment + 1);\n}\n\nint test_tree_modification() {\n    TreeNode leaf1 = {10, NULL, NULL};\n    TreeNode leaf2 = {20, NULL, NULL};\n    TreeNode root = {5, &leaf1, &leaf2};\n    increment_tree(&root, 5);\n    return root.data;\n}\n\n// What value is returned by test_tree_modification()?",
        "answer": "10"
    }
},
{
    "id": "MF-PP-S006-V004",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-FunctionStructParam",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "function_struct_param",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Analyze function pointer within struct parameter. What is the final result value?",
        "code": "typedef struct {\n    int value;\n    int (*operation)(int);\n} Calculator;\n\nint double_value(int x) {\n    return x * 2;\n}\n\nint square_value(int x) {\n    return x * x;\n}\n\nvoid apply_operation(Calculator *calc) {\n    calc->value = calc->operation(calc->value);\n}\n\nint test_calculator() {\n    Calculator calc = {6, square_value};\n    apply_operation(&calc);\n    return calc.value;\n}\n\n// What value is returned by test_calculator()?",
        "answer": "36"
    }
}

```

```

    }
},
{
  "id": "MF-PP-S006-V005",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-UnionComplexParam",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "union_complex_param",
    "passing_type": "pointer",
    "data_type": "complex",
    "modification_effect": "original"
  },
  "task": {
    "description": "Track complex union parameter modification. What is the integer interpretation after function call?",
    "code": "typedef union {\n    int integer;\n    float floating;\n} struct {\n    short low;\n    short high;\n} parts;\n\nvoid\nmodify_union_value(value *v, int mode) {\n    if (mode == 1) {\n        v->integer =\n0x12345678;\n    } else if (mode == 2) {\n        v->parts.low = 0x1234;\n        v->parts.high = 0x5678;\n    }\n}\n\nint test_union_modification() {\n    value val;\n    val.integer = 0;\n    modify_union_value(&val, 2);\n    return val.integer;\n}\n\n// What value is returned by test_union_modification()?",
    "answer": "1450744916"
  }
},
{
  "id": "MF-PP-S006-V006",
  "metadata": {
    "name": "MultiFunc-ParameterPassing-variableLengthArray",
    "category": "MultiFunc-Level",
    "subcategory": "Parameter Passing",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "vla_parameter_passing",
    "passing_type": "pointer",
    "data_type": "array",
    "modification_effect": "original"
  },
  "task": {
    "description": "Examine variable length array parameter passing. What is the sum of the middle row?"
  }
}

```



```

        "modification_effect": "original"
    },
    "task": {
        "description": "Track flexible array member modification. What is the sum of all data elements?",

        "code": "typedef struct {\n    int count;\n    int data[];\n} FlexibleArray;\n\nvoid double_elements(FlexibleArray *fa) {\n    for (int i = 0; i < fa->count; i++) {\n        fa->data[i] *= 2;\n    }\n}\n\nint test_flexible_array() {\n    // Allocate struct with 4 additional integers\n    FlexibleArray *fa =\n    malloc(sizeof(FlexibleArray) + 4 * sizeof(int));\n    fa->count = 4;\n    fa->data[0] = 10;\n    fa->data[1] = 20;\n    fa->data[2] = 30;\n    fa->data[3] = 40;\n\n    double_elements(fa);\n\n    int sum = 0;\n    for (int i = 0; i < fa->count; i++) {\n        sum += fa->data[i];\n    }\n\n    free(fa);\n    return sum;\n}\n\n// What value is returned by test_flexible_array()?",

        "answer": "200"
    }
},
{
    "id": "MF-PP-S006-V009",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ComplexPointerMath",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "complex_pointer_arithmetic",
        "passing_type": "pointer",
        "data_type": "complex",
        "modification_effect": "original"
    },
    "task": {
        "description": "Examine complex pointer arithmetic in parameter passing. What is the final value at position 2?",

        "code": "typedef struct {\n    int x, y;\n} Point;\n\nvoid manipulate_points(Point *points, int count) {\n    Point *ptr = points;\n    for (int i = 0; i < count; i++, ptr++) {\n        ptr->x += (ptr - points) * 10;\n        ptr->y *= 2;\n    }\n}\n\nint test_pointer_arithmetic() {\n    Point points[4] = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};\n    manipulate_points(points, 4);\n    return points[2].x;\n}\n\n// What value is returned by test_pointer_arithmetic()?",

        "answer": "25"
    }
},
{
    "id": "MF-PP-S006-V010",
    "metadata": {
        "name": "MultiFunc-ParameterPassing-ConstCorrectness",
        "category": "MultiFunc-Level",
        "subcategory": "Parameter Passing",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "const_correctness",
        "passing_type": "pointer",
        "data_type": "const",
        "modification_effect": "original"
    }
}

```

```

    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "const_correctness_param",
    "passing_type": "pointer",
    "data_type": "complex",
    "modification_effect": "none"
  },
  "task": {
    "description": "Analyze const correctness in parameter passing. What is the maximum value found?",
    "code": "typedef struct {\n    const int *data;\n    int size;\n} ReadOnlyContainer;\n\nint find_maximum(const ReadOnlyContainer *container) {\n    int max = container->data[0];\n    for (int i = 1; i < container->size; i++) {\n        if (container->data[i] > max) {\n            max = container->data[i];\n        }\n    }\n    return max;\n}\n\nint test_const_container() {\n    const int values[] = {15, 42, 8, 99, 27};\n    ReadOnlyContainer container = {values, 5};\n    return find_maximum(&container);\n}\n\n// what value is returned by test_const_container()?",
    "answer": "99"
  }
}
]

```

4C - 状态传播推理 [State Propagation] (64)

状态传播推理变式生成提示词

任务目标

基于给定的状态传播推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对跨函数状态传播机制的理解能力，包括全局变量、静态变量、闭包捕获和对象状态等核心状态传播推理技能。

状态传播推理特征分析

状态传播推理关注程序中状态信息在函数调用间的传递、维护和变化，重点测试模型对不同存储类型、生存期管理、状态共享机制和数据持久性的深层理解，强调对跨函数状态管理的精确分析能力。

关键要求

- **描述语言**：所有task描述必须使用英文
- **变式数量**：每个种子任务严格生成8-12个变式，确保数量充足
- **答案唯一性**：保证答案准确且唯一，避免歧义和多解情况
- **状态传播聚焦**：重点关注状态在函数间的传播和变化，强调状态管理机制

变式生成维度

1. 全局变量状态分析变式

- **基本全局变量变式**：简单全局变量的跨函数修改分析
- **全局数组状态变式**：全局数组在多函数中的状态变化
- **全局结构体变式**：全局结构体成员的状态传播
- **多全局变量协同变式**：多个全局变量的协同状态变化
- **全局变量竞争变式**：多函数同时修改全局变量的分析
- **全局常量状态变式**：全局常量和全局变量的混合使用

2. 静态变量持久化分析变式

- **函数静态变量变式**：函数内静态变量的持久化分析

- **静态数组管理变式**：静态数组的状态管理和访问
- **静态计数器变式**：静态变量作为计数器的使用
- **静态缓存变式**：静态变量实现缓存机制的分析
- **多静态变量协作变式**：多个静态变量的协作状态管理
- **静态变量初始化变式**：静态变量初始化时机和影响

3. 闭包捕获状态分析变式

- **简单闭包捕获变式**：基础闭包变量捕获的状态分析
- **嵌套闭包状态变式**：多层嵌套闭包的状态传播
- **闭包修改状态变式**：闭包内部修改捕获变量的效果
- **多闭包共享变式**：多个闭包共享同一外部变量
- **闭包生存期变式**：闭包的生存期和状态持久性
- **闭包状态隔离变式**：不同闭包实例的状态隔离

4. 对象状态管理变式

- **对象属性状态变式**：对象属性在方法调用间的状态变化
- **对象状态链式变式**：链式方法调用的对象状态变化
- **对象状态复制变式**：对象复制和引用的状态传播
- **对象状态继承变式**：继承关系中的状态传播
- **对象状态组合变式**：对象组合中的状态管理
- **对象状态序列化变式**：对象状态的序列化和恢复

5. 状态同步和竞争分析变式

- **状态竞争条件变式**：多函数访问共享状态的竞争分析
- **状态同步机制变式**：状态同步机制的效果分析
- **原子操作状态变式**：原子操作对状态的影响
- **锁保护状态变式**：锁机制保护的状态变化
- **无锁状态管理变式**：无锁数据结构的状态管理
- **状态一致性变式**：状态一致性保证的分析

6. 状态生存期分析变式

- **状态生命周期变式**：不同存储类型的状态生命周期
- **状态销毁时机变式**：状态销毁的时机和影响
- **状态持久化机制变式**：状态持久化的实现机制
- **临时状态管理变式**：临时状态的创建和销毁
- **状态内存泄漏变式**：状态管理中的内存泄漏分析
- **状态垃圾回收变式**：垃圾回收对状态的影响

7. 复合状态传播变式

- **多层状态传播变式**：多层函数调用的状态传播
- **状态变换链变式**：状态在传播中的变换处理
- **状态聚合变式**：多个状态的聚合和合并
- **状态分发变式**：状态的分发和分散处理
- **状态缓存变式**：状态缓存机制的实现和分析
- **状态版本管理变式**：状态版本控制的实现

8. 函数式状态管理变式

- **不可变状态变式**：不可变状态的传播和管理
- **函数式状态更新变式**：函数式状态更新的模式
- **状态单子变式**：状态单子模式的应用分析
- **纯函数状态变式**：纯函数中的状态处理
- **状态组合子变式**：状态组合子的使用分析
- **函数式状态机变式**：函数式状态机的实现

9. 状态模式应用变式

- **状态机模式变式**：状态机模式的实现和状态转换
- **观察者状态变式**：观察者模式中的状态传播
- **单例状态变式**：单例模式的状态管理
- **工厂状态变式**：工厂模式中的状态创建
- **策略状态变式**：策略模式中的状态管理
- **命令状态变式**：命令模式中的状态记录

10. 高级状态传播变式

- **分布式状态变式**：分布式系统中的状态传播
- **异步状态变式**：异步操作中的状态管理
- **事件驱动状态变式**：事件驱动的状态传播
- **响应式状态变式**：响应式编程的状态管理
- **状态流处理变式**：状态流的处理和传播
- **状态快照变式**：状态快照的创建和恢复

复杂度层次设计

简单状态传播 (Easy)

- 单一全局变量或静态变量的基础状态传播
- 清晰的状态变化和传播路径
- 直观的状态管理机制
- 基础的状态生存期概念

中等状态传播 (Medium)

- 2-3个状态变量的协同传播
- 包含条件分支的状态变化
- 中等复杂度的对象状态管理
- 基础的闭包状态捕获

复杂状态传播 (Hard)

- 多变量复杂状态传播网络
- 深层的状态管理机制
- 复杂的闭包和对象状态
- 涉及状态优化和模式的分析

专家级状态传播 (Expert)

- 极复杂的状态传播场景
- 高级状态管理模式和机制
- 并发和异步状态管理
- 需要深度架构知识的状态分析

生成策略

种子分析策略

1. **识别状态类型**：分析种子任务涉及的主要状态类型
2. **提取传播模式**：识别状态传播的核心模式和机制
3. **确定状态生存期**：分析状态的生存期和持久性特征
4. **评估复杂度层次**：评估状态传播的复杂程度

变式设计原则

1. **状态传播导向**：每个变式都应密切关注状态传播机制

2. **英文描述**: 所有**task**描述必须使用标准英文
3. **答案唯一**: 严格确保答案的准确性和唯一性
4. **数量保证**: 严格确保每个种子生成8-12个变式

质量保证

1. **状态语义验证**: 验证状态传播的语义正确性
2. **生存期一致性**: 确保状态生存期分析的一致性
3. **答案唯一性验证**: 严格检查答案的唯一性和确定性
4. **英文质量保证**: 确保描述的英文表达准确清晰

输出格式要求

```
```json
[
 {
 "id": "MF-SP-S00X-V001",
 "metadata": {
 "name": "MultiFunc-StatePropagation-VariantName",
 "category": "MultiFunc-Level",
 "subcategory": "State Propagation",
 "type": "variant",
 "source": "Generated",
 "language": "target_language",
 "difficulty": "easy/medium/hard/expert",
 "intervention": 0,
 "variant_type": "variant_type_label",
 "state_type": "global/static/closure/object",
 "propagation_pattern": "linear/branched/circular/complex",
 "persistence_level": "temporary/session/persistent/permanent"
 },
 "task": {
 "description": "English description of the state propagation analysis task",
 "code": "Code containing state propagation scenarios to analyze",
 "answer": "Unique and accurate state propagation analysis result"
 }
 },
 {下一个变式...}
]
```

```

特殊字段说明

state_type: 标识主要的状态类型

propagation_pattern: 标识状态传播的模式

persistence_level: 标识状态的持久性级别

生成目标

为每个提供的种子任务严格生成8-12个状态传播变式，确保：

每种难度等级至少包含2-3个变式

涵盖至少6-8种不同的状态传播分析类型

包含不同的状态类型和传播复杂度

所有**task**描述使用标准英文表达

严格保证答案的准确性和唯一性

重点测试场景

状态变化追踪：准确追踪状态在函数调用间的变化

生存期管理：理解不同存储类型的状态生存期
状态共享机制：分析状态在多个函数间的共享
闭包状态捕获：理解闭包的状态捕获和维护机制
对象状态演化：分析对象状态在方法调用中的演化
状态持久化：理解状态的持久化和恢复机制

英文描述模板示例

```
"Track the state changes across multiple function calls and determine..."  
"Analyze the propagation of global/static state through the function chain..."  
"Determine the final state value after all function executions..."  
"Trace the closure state capture and modification effects..."  
"Calculate the object state after a series of method invocations..."  
"Evaluate the persistent state changes across function boundaries..."
```

答案唯一性保证策略

确定性状态变化：确保状态变化是确定性的
明确初始状态：明确定义所有状态的初始值
清晰操作顺序：确保状态操作的顺序是明确的
避免竞争条件：避免产生竞争条件的状态访问
具体数值结果：使用具体的数值结果，避免符号化答案
常见状态传播模式
累积模式：状态在函数调用中逐步累积
重置模式：状态的重置和重新初始化
转换模式：状态在传播中的转换和变换
缓存模式：状态作为缓存的使用模式
计数模式：状态作为计数器的递增递减
标记模式：状态作为标记位的设置和检查

请基于此提示词，为给定的状态传播推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

```
[  
  {  
    "id": "MF-SP-S001-V001",  
    "metadata": {  
      "name": "MultiFunc-StatePropagation-GlobalCounterSequence",  
      "category": "MultiFunc-Level",  
      "subcategory": "State Propagation",  
      "type": "variant",  
      "source": "Generated",  
      "language": "c",  
      "difficulty": "easy",  
      "intervention": 0,  
      "variant_type": "simple_global_sequence",  
      "state_type": "global",  
      "propagation_pattern": "linear",  
      "persistence_level": "persistent"  
    },  
    "task": {  
      "description": "Track the global counter modifications through a linear sequence of function calls and determine the final value."  
    }  
  }]
```

```

    "code": "int global_sum = 5;\n\nvoid add_ten() {\n    global_sum +=\n10;\n}\n\nvoid multiply_by_two() {\n    global_sum *= 2;\n}\n\nvoid subtract_five() {\n    global_sum -= 5;\n}\n\nint execute_sequence() {\n    add_ten();\n    multiply_by_two();\n    subtract_five();\n    return global_sum;\n}\n\n// What value is returned by\nexecute_sequence()?",\n    "answer": "25"
}
},
{
    "id": "MF-SP-S001-V002",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ConditionalGlobalUpdate",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "conditional_global_modification",
        "state_type": "global",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze conditional global variable modifications across\nmultiple function calls and determine the final state.",
        "code": "int global_value = 10;\nint global_flag = 1;\n\nvoid\nconditional_update(int condition) {\n    if (condition > 0) {\n        global_value +=\n        condition * 3;\n    } else {\n        global_value /= 2;\n    }\n}\n\nvoid\ntoggle_and_process() {\n    global_flag = !global_flag;\n    if (global_flag) {\n        global_value += 5;\n    } else {\n        global_value -= 3;\n    }\n}\n\nint\ntest_conditional_flow() {\n    conditional_update(4);\n    toggle_and_process();\n    conditional_update(-1);\n    return global_value;\n}\n\n// What value is returned by\ntest_conditional_flow()?",\n        "answer": "12"
    }
},
{
    "id": "MF-SP-S001-V003",
    "metadata": {
        "name": "MultiFunc-StatePropagation-MultiGlobalCoordination",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "multi_global_coordination",
        "state_type": "global",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    }
}

```

```

    },
    "task": {
        "description": "Track coordinated modifications of multiple global variables through interconnected function calls.",
        "code": "int global_a = 8;\nint global_b = 12;\nint global_c = 3;\n\nvoid cross_update() {\n    global_a = global_a + global_b;\n    global_b = global_a - global_c;\n    global_c = global_b / 4;\n}\n\nvoid scale_variables(int factor) {\n    global_a *= factor;\n    global_b += global_a;\n    global_c = global_a % global_b;\n}\n\nint compute_final_result() {\n    cross_update();\n    scale_variables(2);\n    return global_a + global_b + global_c;\n}\n\n// What value is returned by compute_final_result()?",
        "answer": "108"
    }
},
{
    "id": "MF-SP-S001-v004",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalArraySum",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "global_array_manipulation",
        "state_type": "global",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze global array state changes through sequential function calls and calculate the final sum.",
        "code": "int numbers[4] = {2, 4, 6, 8};\n\nvoid increment_all() {\n    for (int i = 0; i < 4; i++) {\n        numbers[i] += 2;\n    }\n}\n\nvoid multiply_even_indices() {\n    for (int i = 0; i < 4; i += 2) {\n        numbers[i] *= 3;\n    }\n}\n\nint calculate_array_sum() {\n    increment_all();\n    multiply_even_indices();\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += numbers[i];\n    }\n    return sum;\n}\n\n// What value is returned by calculate_array_sum()?",
        "answer": "58"
    }
},
{
    "id": "MF-SP-S001-v005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalStateReset",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "global_reset_pattern",
    }
}

```

```

        "state_type": "global",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global variable state through reset operations and subsequent modifications.",
        "code": "int accumulator = 0;\nint base_value = 7;\n\nvoid add_base() {\n    accumulator += base_value;\n}\n\nvoid reset_accumulator() {\n    accumulator = 0;\n    base_value += 3;\n}\n\nvoid double_accumulator() {\n    accumulator *= 2;\n}\n\nint test_reset_sequence() {\n    add_base();\n    add_base();\n    reset_accumulator();\n    add_base();\n    double_accumulator();\n    return accumulator;\n}\n\n// What value is returned by test_reset_sequence()?",
        "answer": "20"
    }
},
{
    "id": "MF-SP-S001-v006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalCounterChain",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "global_chain_operations",
        "state_type": "global",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze chained global variable operations where each function builds upon the previous state.",
        "code": "int global_result = 1;\nint step_counter = 0;\n\nvoid step_one() {\n    global_result = global_result + 4;\n    step_counter++;\n}\n\nvoid step_two() {\n    global_result = global_result * step_counter;\n    step_counter++;\n}\n\nvoid step_three() {\n    global_result = global_result + (step_counter * 2);\n    step_counter++;\n}\n\nint execute_chain() {\n    step_one();\n    step_two();\n    step_three();\n    return global_result;\n}\n\n// What value is returned by execute_chain()?",
        "answer": "11"
    }
},
{
    "id": "MF-SP-S001-v007",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalStructState",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
    }
}

```

```

        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "global_struct_modification",
        "state_type": "global",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global structure member modifications across multiple function calls and calculate the final aggregate value.",
        "code": "struct GlobalData {\n    int x;\n    int y;\n    int sum;\n} data = {3,\n7, 0};\n\nvoid update_sum() {\n    data.sum = data.x + data.y;\n}\n\nvoid\nmodify_coordinates(int dx, int dy) {\n    data.x += dx;\n    data.y += dy;\n}\n\nvoid\napply_transform() {\n    data.x *= 2;\n    data.y -= 1;\n    update_sum();\n}\n\nint\n calculate_total() {\n    modify_coordinates(2, -3);\n    apply_transform();\n    return data.sum;\n}\n\n// What value is returned by calculate_total()?",\n        "answer": "13"
    }
},
{
    "id": "MF-SP-S001-v008",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalLoopAccumulation",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "global_loop_accumulation",
        "state_type": "global",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze global variable accumulation through repeated function calls in a loop structure.",
        "code": "int total = 0;\nint multiplier = 2;\n\nvoid add_product(int value) {\n    total += value * multiplier;\n}\n\nvoid increment_multiplier() {\n    multiplier++;\n}\n\nint process_values() {\n    for (int i = 1; i <= 3; i++) {\n        add_product(i);\n        if (i % 2 == 0) {\n            increment_multiplier();\n        }\n    }\n    return total;\n}\n\n// What value is returned by process_values()?",\n        "answer": "11"
    }
},
{
    "id": "MF-SP-S001-v009",
    "metadata": {
        "name": "MultiFunc-StatePropagation-GlobalBitwiseOps",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",

```

```
"source": "Generated",
"language": "c",
"difficulty": "expert",
"intervention": 3,
"variant_type": "global_bitwise_operations",
"state_type": "global",
"propagation_pattern": "complex",
"persistence_level": "persistent"
},
"task": {
  "description": "Track global variable state through bitwise operations across multiple function calls.",
  "code": "int global_bits = 0x0F;\nint mask = 0x03;\n\nvoid apply_xor() {\n  global_bits ^= mask;\n}\n\nvoid shift_and_or() {\n  global_bits = (global_bits << 1) | 0x01;\n  mask = mask << 1;\n}\n\nvoid final_and_operation() {\n  global_bits &= 0x1F;\n}\n\nint compute_bitwise_result() {\n  apply_xor();\n  shift_and_or();\n  apply_xor();\n  final_and_operation();\n  return global_bits;\n}\n\n// what value is returned by compute_bitwise_result()?",
  "answer": "25"
},
{
  "id": "MF-SP-S002-V001",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ArrayElementShift",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "array_element_shift",
    "state_type": "global",
    "propagation_pattern": "Linear",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Analyze global array element shifting and modification across function calls to determine the final sum.",
    "code": "int values[4] = {5, 10, 15, 20};\n\nvoid shift_left() {\n  for (int i = 0; i < 3; i++) {\n    values[i] = values[i + 1];\n  }\n  values[3] = 0;\n}\n\nvoid add_constant() {\n  for (int i = 0; i < 4; i++) {\n    values[i] += 3;\n  }\n}\n\nint compute_array_sum() {\n  shift_left();\n  add_constant();\n  int sum = 0;\n  for (int i = 0; i < 4; i++) {\n    sum += values[i];\n  }\n  return sum;\n}\n\n// what value is returned by compute_array_sum()?",
    "answer": "60"
  }
},
{
  "id": "MF-SP-S002-V002",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ArrayMatrixOps",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "array_matrix_ops",
    "state_type": "global",
    "propagation_pattern": "Linear",
    "persistence_level": "persistent"
  }
}
```

```

        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "array_matrix_operations",
        "state_type": "global",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global 2D array modifications through matrix operations and calculate the final element sum.",
        "code": "int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};\n\nvoid transpose_matrix() {\n    for (int i = 0; i < 3; i++) {\n        for (int j = i + 1; j < 3; j++) {\n            int temp = matrix[i][j];\n            matrix[i][j] = matrix[j][i];\n            matrix[j][i] = temp;\n        }\n    }\n}\n\nvoid scale_diagonal(int factor) {\n    for (int i = 0; i < 3; i++) {\n        matrix[i][i] *= factor;\n    }\n}\n\nint calculate_matrix_sum() {\n    transpose_matrix();\n    scale_diagonal(2);\n    int sum = 0;\n    for (int i = 0; i < 3; i++) {\n        for (int j = 0; j < 3; j++) {\n            sum += matrix[i][j];\n        }\n    }\n    return sum;\n}\n\n// What value is returned by calculate_matrix_sum()?",\n        "answer": "60"
    }
},
{
    "id": "MF-SP-S002-V003",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayRotation",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "array_rotation",
        "state_type": "global",
        "propagation_pattern": "circular",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze global array rotation operations and determine the final array configuration sum.",
        "code": "int data[5] = {2, 4, 6, 8, 10};\n\nvoid rotate_right() {\n    int temp = data[4];\n    for (int i = 4; i > 0; i--) {\n        data[i] = data[i - 1];\n    }\n    data[0] = temp;\n}\n\nvoid multiply_ends() {\n    data[0] *= 2;\n    data[4] *= 2;\n}\n\nint process_rotation() {\n    rotate_right();\n    multiply_ends();\n    rotate_right();\n    int sum = 0;\n    for (int i = 0; i < 5; i++) {\n        sum += data[i];\n    }\n    return sum;\n}\n\n// What value is returned by process_rotation()?",\n        "answer": "50"
    }
}

```

```
        }
    },
{
    "id": "MF-SP-S002-V004",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayCondensation",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "array_condensation",
        "state_type": "global",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global array condensation operations where adjacent elements are combined and calculate the final result.",
        "code": "int sequence[6] = {1, 3, 5, 7, 9, 11};\nint length = 6;\n\nvoid condense_pairs()\n{\n    for (int i = 0; i < length - 1; i += 2) {\n        sequence[i/2] = sequence[i] + sequence[i + 1];\n    }\n    length = length / 2;\n}\n\nvoid square_elements()\n{\n    for (int i = 0; i < length; i++) {\n        sequence[i] = sequence[i] * sequence[i];\n    }\n}\n\nint compute_condensed_sum()\n{\n    condense_pairs();\n    square_elements();\n    condense_pairs();\n    return sequence[0];\n}\n\n// What value is returned by compute_condensed_sum()?",
        "answer": "400"
    }
},
{
    "id": "MF-SP-S002-V005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayFiltering",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "array_filtering",
        "state_type": "global",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze global array filtering operations where elements are conditionally modified and calculate the remaining sum."
    }
}
```

```

        "code": "int numbers[6] = {12, 15, 18, 21, 24, 27};\n\nvoid
filter_divisible_by_three() {\n    for (int i = 0; i < 6; i++) {\n        if (numbers[i] % 3
== 0) {\n            numbers[i] = numbers[i] / 3;\n        } else {\n            numbers[i]
= 0;\n        }\n    }\n}\n\nvoid remove_zeros() {\n    for (int i = 0; i < 6; i++) {\n        if (numbers[i] == 0) {\n            numbers[i] = -1;\n        }\n    }\n}\n\nint
calculate_filtered_sum() {\n    filter_divisible_by_three();\n    remove_zeros();\n    int
sum = 0;\n    for (int i = 0; i < 6; i++) {\n        if (numbers[i] > 0) {\n            sum
+= numbers[i];\n        }\n    }\n    return sum;\n}\n\n// What value is returned by
calculate_filtered_sum()?",\n        "answer": "33"
    },
},
{
    "id": "MF-SP-S002-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayAccumulation",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "array_accumulation",
        "state_type": "global",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global array accumulation where each element becomes the
sum of previous elements.",\n        "code": "int accumulator[4] = {3, 5, 2, 8};\n\nvoid create_cumulative() {\nfor (int i = 1; i < 4; i++) {\n    accumulator[i] += accumulator[i - 1];\n}\n}\n\nvoid subtract_first() {\n    int first = accumulator[0];\n    for (int i = 0; i < 4; i++) {\n        accumulator[i] -= first;\n    }\n}\n\nint get_final_sum() {\n    create_cumulative();\n    subtract_first();\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += accumulator[i];\n    }\n    return sum;\n}\n\n// What value is
returned by get_final_sum()?",\n        "answer": "18"
    },
},
{
    "id": "MF-SP-S002-V007",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayInterleaving",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "array_interleaving",
        "persistence_level": "persistent"
    }
}

```

```

        "state_type": "global",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze complex global array interleaving operations with conditional modifications.",
        "code": "int primary[4] = {6, 12, 18, 24};\nint secondary[4] = {1, 2, 3, 4};\n\nvoid interleave_arrays() {\n    for (int i = 0; i < 4; i++) {\n        if (i % 2 == 0) {\n            primary[i] = primary[i] + secondary[i];\n        } else {\n            primary[i] = primary[i] * secondary[i];\n        }\n    }\n}\n\nvoid reverse_and_modify() {\n    for (int i = 0; i < 2; i++) {\n        int temp = primary[i];\n        primary[i] = primary[3 - i] + 5;\n        primary[3 - i] = temp - 5;\n    }\n}\n\nint compute_interleaved_result() {\n    interleave_arrays();\n    reverse_and_modify();\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += primary[i];\n    }\n    return sum;\n}\n\n// What value is returned by compute_interleaved_result()?",\n        "answer": "94"
    }
},
{
    "id": "MF-SP-S002-v008",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ArrayPartitioning",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "array_partitioning",
        "state_type": "global",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track global array partitioning operations where elements are redistributed based on conditions.",
        "code": "int dataset[8] = {4, 7, 2, 9, 6, 1, 8, 3};\n\nvoid partition_even_odd() {\n    int even_sum = 0, odd_sum = 0;\n    for (int i = 0; i < 8; i++) {\n        if (dataset[i] % 2 == 0) {\n            even_sum += dataset[i];\n        } else {\n            odd_sum += dataset[i];\n        }\n    }\n    dataset[0] = even_sum;\n    dataset[1] = odd_sum;\n    for (int i = 2; i < 8; i++) {\n        dataset[i] = 0;\n    }\n}\n\nvoid apply_difference() {\n    dataset[2] = dataset[0] - dataset[1];\n}\n\nint calculate_partition_result() {\n    partition_even_odd();\n    apply_difference();\n    return dataset[0] + dataset[1] + dataset[2];\n}\n\n// What value is returned by calculate_partition_result()?",\n        "answer": "60"
    }
},
{
    "id": "MF-SP-S003-v001",
    "metadata": {

```

```
        "name": "MultiFunc-StatePropagation-StaticIncrement",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "static_increment_pattern",
        "state_type": "static",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track static variable incrementation across multiple function calls and determine the final accumulated value.",
        "code": "int get_sequence_number() {\n    static int sequence = 0;\n    sequence += 5;\n    return sequence;\n}\n\nint calculate_sequence_sum() {\n    int total = 0;\n    for (int i = 0; i < 4; i++) {\n        total += get_sequence_number();\n    }\n    return total;\n}\n\n// What value is returned by calculate_sequence_sum()?",
        "answer": "50"
    }
},
{
    "id": "MF-SP-S003-V002",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticToggle",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "static_toggle_state",
        "state_type": "static",
        "propagation_pattern": "circular",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static variable toggle behavior across function calls and calculate the final state value.",
        "code": "int toggle_state() {\n    static int state = 1;\n    static int multiplier = 2;\n    state = state * multiplier;\n    multiplier = (multiplier == 2) ? 3 : 2;\n    return state;\n}\n\nint process_toggles() {\n    int result = 0;\n    result += toggle_state();\n    result += toggle_state();\n    result += toggle_state();\n    return result;\n}\n\n// What value is returned by process_toggles()?",
        "answer": "26"
    }
},
{
    "id": "MF-SP-S003-V003",
    "metadata": {
```

```
        "name": "MultiFunc-StatePropagation-NestedstaticCounters",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_static_counters",
        "state_type": "static",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track multiple nested static counters across interconnected function calls and determine the final combined result.",
        "code": "int inner_counter() {\n    static int count = 0;\n    count += 3;\n    return count;\n}\n\nint outer_counter() {\n    static int total = 0;\n    total += inner_counter();\n    return total;\n}\n\nint combined_counter() {\n    static int combined = 0;\n    combined = outer_counter() + inner_counter();\n    return combined;\n}\n\nint test_nested_counters() {\n    int result1 = combined_counter();\n    int result2 = combined_counter();\n    return result1 + result2;\n}\n\n// What value is returned by test_nested_counters()?",
        "answer": "36"
    }
},
{
    "id": "MF-SP-S003-V004",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticFactorial",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "static_factorial_accumulation",
        "state_type": "static",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static variable factorial accumulation across function calls and determine the final result.",
        "code": "int factorial_step() {\n    static int result = 1;\n    static int factor = 1;\n    result *= factor;\n    factor++;\n    return result;\n}\n\nint compute_factorial_sequence() {\n    int sum = 0;\n    for (int i = 0; i < 5; i++) {\n        sum += factorial_step();\n    }\n    return sum;\n}\n\n// What value is returned by compute_factorial_sequence()?",
        "answer": "34"
    }
},
{
    "id": "MF-SP-S004-V005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticFactorial",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "static_factorial_accumulation",
        "state_type": "static",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static variable factorial accumulation across function calls and determine the final result.",
        "code": "int factorial_step() {\n    static int result = 1;\n    static int factor = 1;\n    result *= factor;\n    factor++;\n    return result;\n}\n\nint compute_factorial_sequence() {\n    int sum = 0;\n    for (int i = 0; i < 5; i++) {\n        sum += factorial_step();\n    }\n    return sum;\n}\n\n// What value is returned by compute_factorial_sequence()?",
        "answer": "34"
    }
},
```

```
{
  "id": "MF-SP-S003-V005",
  "metadata": {
    "name": "MultiFunc-StatePropagation-StaticFibonacci",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "static_fibonacci_sequence",
    "state_type": "static",
    "propagation_pattern": "complex",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Track static variables implementing Fibonacci sequence generation and calculate the sum of generated numbers.",
    "code": "int fibonacci_next() {\n    static int prev = 0;\n    static int curr = 1;\n    int next = prev + curr;\n    prev = curr;\n    curr = next;\n    return next;\n}\nint generate_fibonacci_sum() {\n    int sum = 1;\n    for (int i = 0; i < 5; i++) {\n        sum += fibonacci_next();\n    }\n    return sum;\n}// what value is returned by generate_fibonacci_sum()?",
    "answer": "21"
  }
},
{
  "id": "MF-SP-S003-V006",
  "metadata": {
    "name": "MultiFunc-StatePropagation-StaticBitshift",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "static_bitwise_shift",
    "state_type": "static",
    "propagation_pattern": "complex",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Analyze static variable bitwise shift operations across function calls and determine the final bit pattern value.",
    "code": "int bit_shifter() {\n    static int value = 1;\n    static int direction = 1;\n    if (direction == 1) {\n        value = value << 1;\n        if (value >= 16) direction = -1;\n    } else {\n        value = value >> 1;\n        if (value <= 1) direction = 1;\n    }\n    return value;\n}\nint test_bit_shifting() {\n    int sum = 0;\n    for (int i = 0; i < 7; i++) {\n        sum += bit_shifter();\n    }\n    return sum;\n}// what value is returned by test_bit_shifting()?",
    "answer": "46"
  }
}
```

```

        },
    },
    {
        "id": "MF-SP-S003-V007",
        "metadata": {
            "name": "MultiFunc-StatePropagation-StaticModuloCounter",
            "category": "MultiFunc-Level",
            "subcategory": "State Propagation",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "medium",
            "intervention": 1,
            "variant_type": "static_modulo_counter",
            "state_type": "static",
            "propagation_pattern": "circular",
            "persistence_level": "persistent"
        },
        "task": {
            "description": "Track static variable modulo counter behavior and calculate the accumulated results across function calls.",
            "code": "int modulo_counter() {\n    static int counter = 0;\n    static int base = 7;\n    counter = (counter + base) % 20;\n    return counter;\n}\n\nvoid reset_base()\n{\n    static int reset_count = 0;\n    reset_count++;\n    if (reset_count == 2) {\n        modulo_counter();\n    }\n}\n\nint test_modulo_sequence()\n{\n    int sum = 0;\n    sum += modulo_counter();\n    reset_base();\n    sum += modulo_counter();\n    reset_base();\n    sum += modulo_counter();\n    return sum;\n}\n\n// what value is returned by test_modulo_sequence()?",
            "answer": "35"
        }
    },
    {
        "id": "MF-SP-S003-V008",
        "metadata": {
            "name": "MultiFunc-StatePropagation-StaticPrimeGenerator",
            "category": "MultiFunc-Level",
            "subcategory": "State Propagation",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "expert",
            "intervention": 3,
            "variant_type": "static_prime_generation",
            "state_type": "static",
            "propagation_pattern": "complex",
            "persistence_level": "persistent"
        },
        "task": {
            "description": "Analyze static variables implementing prime number generation and calculate the sum of generated primes."
        }
    }
}

```

```
        "code": "int next_prime() {\n    static int current = 2;\n    static int found = 0;\n    while (!found) {\n        found = 1;\n        for (int i = 2; i * i <= current; i++) {\n            if (current % i == 0) {\n                found = 0;\n                break;\n            }\n        }\n        if (!found) current++;\n    }\n    int result = current;\n    current++;\n    found = 0;\n    return result;\n}\n\nint generate_prime_sum() {\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += next_prime();\n    }\n    return sum;\n}\n\n// What value is returned by generate_prime_sum()?",\n        "answer": "17"\n    },\n    {\n        "id": "MF-SP-S004-V001",\n        "metadata": {\n            "name": "MultiFunc-StatePropagation-StaticArrayFIFO",\n            "category": "MultiFunc-Level",\n            "subcategory": "State Propagation",\n            "type": "variant",\n            "source": "Generated",\n            "language": "c",\n            "difficulty": "medium",\n            "intervention": 1,\n            "variant_type": "static_array_fifo",\n            "state_type": "static",\n            "propagation_pattern": "circular",\n            "persistence_level": "persistent"\n        },\n        "task": {\n            "description": "Analyze static array implementing FIFO queue behavior across function calls and determine the final sum.",\n            "code": "int enqueue_value(int value) {\n    static int queue[3] = {0, 0, 0};\n    static int front = 0;\n    queue[front] = value;\n    front = (front + 1) % 3;\n    int sum = 0;\n    for (int i = 0; i < 3; i++) {\n        sum += queue[i];\n    }\n    return sum;\n}\n\nint test_fifo_operations() {\n    enqueue_value(5);\n    enqueue_value(10);\n    enqueue_value(15);\n    return enqueue_value(20);\n}\n\n// What value is returned by test_fifo_operations()?",\n            "answer": "45"\n        },\n    },\n    {\n        "id": "MF-SP-S004-V002",\n        "metadata": {\n            "name": "MultiFunc-StatePropagation-StaticArrayStack",\n            "category": "MultiFunc-Level",\n            "subcategory": "State Propagation",\n            "type": "variant",\n            "source": "Generated",\n            "language": "c",\n            "difficulty": "hard",\n            "intervention": 2,\n            "variant_type": "static_array_stack",\n            "state_type": "static",\n            "propagation_pattern": "linear",\n            "persistence_level": "persistent"\n        }\n    }
```

```

    },
    "task": {
        "description": "Track static array implementing stack operations with push and pop functionality across function calls.",
        "code": "int stack_operation(int value, int is_push) {\n    static int stack[4] = {0, 0, 0, 0};\n    static int top = -1;\n    if (is_push && top < 3) {\n        top++;\n        stack[top] = value;\n    } else if (!is_push && top >= 0) {\n        value = stack[top];\n        stack[top] = 0;\n        top--;\n    }\n    int sum = 0;\n    for (int i = 0; i <= top; i++) {\n        sum += stack[i];\n    }\n    return sum;\n}\n\nint test_stack_operations() {\n    stack_operation(8, 1);\n    stack_operation(12, 1);\n    stack_operation(6, 1);\n    stack_operation(0, 0);\n    return stack_operation(4, 1);\n}\n\n// what value is returned by test_stack_operations()?",\n        "answer": "24"
    }
},
{
    "id": "MF-SP-S004-V003",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayHistogram",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "static_array_histogram",
        "state_type": "static",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static array implementing histogram counting across function calls and calculate the final distribution sum.",
        "code": "int add_to_histogram(int value) {\n    static int histogram[5] = {0, 0, 0, 0, 0};\n    static int total_count = 0;\n    if (value >= 0 && value < 5) {\n        histogram[value]++;\n        total_count++;\n    }\n    int weighted_sum = 0;\n    for (int i = 0; i < 5; i++) {\n        weighted_sum += i * histogram[i];\n    }\n    return weighted_sum;\n}\n\nint test_histogram_operations() {\n    add_to_histogram(1);\n    add_to_histogram(3);\n    add_to_histogram(1);\n    add_to_histogram(4);\n    return add_to_histogram(2);\n}\n\n// what value is returned by test_histogram_operations()?",\n        "answer": "12"
    }
},
{
    "id": "MF-SP-S004-V004",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayMatrix",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
    }
}

```

```

        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "static_array_matrix_ops",
        "state_type": "static",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track static 2D array matrix operations across function calls and determine the final trace value.",
        "code": "int matrix_transform(int row, int col, int value) {\n    static int\nmatrix[2][2] = {{1, 2}, {3, 4}};\n    if (row >= 0 && row < 2 && col >= 0 && col < 2)\n        matrix[row][col] += value;\n    int trace = matrix[0][0] + matrix[1][1];\n    return trace;\n}\n\nint test_matrix_operations() {\n    matrix_transform(0, 0, 3);\n    matrix_transform(1, 1, 2);\n    matrix_transform(0, 1, 5);\n    return matrix_transform(1, 0, 1);\n}\n\n// What value is returned by test_matrix_operations()?",\n        "answer": "11"
    }
},
{
    "id": "MF-SP-S004-V005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayBubbleSort",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "static_array_sorting",
        "state_type": "static",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static array bubble sort implementation across function calls and determine the final sorted sum.",
        "code": "int sort_step() {\n    static int array[4] = {8, 3, 5, 1};\n    static int pass = 0;\n    for (int i = 0; i < 3 - pass; i++) {\n        if (array[i] > array[i + 1]) {\n            int temp = array[i];\n            array[i] = array[i + 1];\n            array[i + 1] = temp;\n        }\n    }\n    pass++;\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += array[i] * (i + 1);\n    }\n    return sum;\n}\n\nint test_sorting_process() {\n    sort_step();\n    sort_step();\n    return sort_step();\n}\n\n// What value is returned by test_sorting_process()?",\n        "answer": "30"
    }
},
{
    "id": "MF-SP-S004-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayBuffer",
        "category": "MultiFunc-Level",

```

```
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "static_array_circular_buffer",
        "state_type": "static",
        "propagation_pattern": "circular",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track static array circular buffer operations and calculate the final buffer state sum.",
        "code": "int circular_buffer(int value) {\n    static int buffer[3] = {0, 0, 0};\n    static int position = 0;\n    static int count = 0;\n    buffer[position] = value;\n    position = (position + 1) % 3;\n    if (count < 3) count++;\n    int sum = 0;\n    for (int i = 0; i < count; i++) {\n        sum += buffer[i];\n    }\n    return sum;\n}\n\nint test_circular_operations() {\n    circular_buffer(7);\n    circular_buffer(14);\n    circular_buffer(21);\n    return circular_buffer(28);\n}\n\n// What value is returned by test_circular_operations()?",\n        "answer": "63"
    }
},
{
    "id": "MF-SP-S004-V007",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayLookup",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "static_array_lookup_table",
        "state_type": "static",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static array lookup table modifications and access patterns across function calls.",
        "code": "int lookup_table(int key, int value, int is_set) {\n    static int table[4] = {10, 20, 30, 40};\n    static int access_count = 0;\n    access_count++;\n\n    if (key >= 0 && key < 4) {\n        if (is_set) {\n            table[key] = value;\n        }\n        return table[key] + access_count;\n    }\n    return access_count;\n}\n\nint test_lookup_operations() {\n    int result = 0;\n    result += lookup_table(1, 25, 1);\n    result += lookup_table(1, 0, 0);\n    result += lookup_table(3, 50, 1);\n    return result;\n}\n\n// What value is returned by test_lookup_operations()?",\n        "answer": "80"
    }
},
{
    "id": "MF-SP-S005-V007",
    "metadata": {
        "name": "MultiFunc-StatePropagation-StaticArrayLookup",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "static_array_lookup_table",
        "state_type": "static",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze static array lookup table modifications and access patterns across function calls.",
        "code": "int lookup_table(int key, int value, int is_set) {\n    static int table[4] = {10, 20, 30, 40};\n    static int access_count = 0;\n    access_count++;\n\n    if (key >= 0 && key < 4) {\n        if (is_set) {\n            table[key] = value;\n        }\n        return table[key] + access_count;\n    }\n    return access_count;\n}\n\nint test_lookup_operations() {\n    int result = 0;\n    result += lookup_table(1, 25, 1);\n    result += lookup_table(1, 0, 0);\n    result += lookup_table(3, 50, 1);\n    return result;\n}\n\n// What value is returned by test_lookup_operations()?",\n        "answer": "80"
    }
},
```

```
{
  "id": "MF-SP-S004-V008",
  "metadata": {
    "name": "MultiFunc-StatePropagation-StaticArrayPriorityQueue",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "static_array_priority_queue",
    "state_type": "static",
    "propagation_pattern": "complex",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Track static array implementing priority queue operations and calculate the final queue state sum.",
    "code": "int priority_queue(int value, int is_insert) {\n    static int queue[3] = {0, 0, 0};\n    static int size = 0;\n    if (is_insert && size < 3) {\n        queue[size] = value;\n        size++;\n        for (int i = size - 1; i > 0 && queue[i] > queue[i-1]; i--) {\n            int temp = queue[i];\n            queue[i] = queue[i-1];\n            queue[i-1] = temp;\n        }\n    } else if (!is_insert && size > 0) {\n        for (int i = 0; i < size - 1; i++) {\n            queue[i] = queue[i + 1];\n        }\n        queue[size - 1] = 0;\n        size--;\n    }\n    int sum = 0;\n    for (int i = 0; i < size; i++) {\n        sum += queue[i];\n    }\n    return sum;\n}\n\nint test_priority_operations() {\n    priority_queue(15, 1);\n    priority_queue(10, 1);\n    priority_queue(20, 1);\n    priority_queue(0, 0);\n    return priority_queue(5, 1);\n}\n\n// what value is returned by test_priority_operations()?",\n    "answer": "30"
  }
},
{
  "id": "MF-SP-S005-V001",
  "metadata": {
    "name": "MultiFunc-StatePropagation-SimpleClosure",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "simple_closure_capture",
    "state_type": "closure",
    "propagation_pattern": "linear",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Track simple closure variable capture and modification to determine the final captured state value."
  }
}
```

```
        "code": "def create_multiplier(factor):\n            def multiply(value):\n                return factor * value\n                return multiply\n            def test_simple_closure():\n                mult_by_3 = create_multiplier(3)\n                mult_by_5 = create_multiplier(5)\n                result1 = mult_by_3(4)\n                result2 = mult_by_5(2)\n                return result1 + result2\n            # what value is returned by test_simple_closure()?",\n        "answer": "22"\n    }\n},\n{\n    "id": "MF-SP-S005-V002",\n    "metadata": {\n        "name": "MultiFunc-StatePropagation-ClosureModification",\n        "category": "MultiFunc-Level",\n        "subcategory": "State Propagation",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "medium",\n        "intervention": 1,\n        "variant_type": "closure_state_modification",\n        "state_type": "closure",\n        "propagation_pattern": "linear",\n        "persistence_level": "persistent"\n    },\n    "task": {\n        "description": "Analyze closure variable modification across multiple function calls and determine the final modified value.",\n        "code": "def create_counter(start):\n            count = start\n            def increment(step=1):\n                nonlocal count\n                count += step\n                return count\n            def get_count():\n                return count\n            return increment, get_count\n        def test_closure_modification():\n            inc, get = create_counter(5)\n            inc(3)\n            inc(2)\n            result = get()\n            inc(4)\n            return result\n        # what value is returned by test_closure_modification()?",\n        "answer": "10"\n    }\n},\n{\n    "id": "MF-SP-S005-V003",\n    "metadata": {\n        "name": "MultiFunc-StatePropagation-ClosureFactoryPattern",\n        "category": "MultiFunc-Level",\n        "subcategory": "State Propagation",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "hard",\n        "intervention": 2,\n        "variant_type": "closure_factory_pattern",\n        "state_type": "closure",\n        "propagation_pattern": "complex",\n        "persistence_level": "persistent"\n    },\n    "task": {\n
```

```

"description": "Track closure factory pattern with multiple instances and shared state modifications.",
"code": "def create_calculator(initial_value):\n    state = {'value': initial_value, 'operations': 0}\n    def add(x):\n        state['value'] += x\n        state['operations'] += 1\n        return state['value']\n    def multiply(x):\n        state['value'] *= x\n        state['operations'] += 1\n        return state['value']\n    def get_operations():\n        return state['operations']\n    return add, multiply, get_operations\n\ndef test_calculator_closures():\n    add1, mult1, ops1 = create_calculator(10)\n    add2, mult2, ops2 = create_calculator(5)\n    add1(5)\n    mult2(3)\n    mult1(2)\n    return add1(0) + mult2(1) + ops1() + ops2()\n\n# What value is returned by test_calculator_closures()?",\n"answer": "49"
}
},
{
"id": "MF-SP-S005-V004",
"metadata": {
"name": "MultiFunc-StatePropagation-ClosureChaining",
"category": "MultiFunc-Level",
"subcategory": "State Propagation",
"type": "variant",
"source": "Generated",
"language": "python",
"difficulty": "hard",
"intervention": 2,
"variant_type": "closure_chaining",
"state_type": "closure",
"propagation_pattern": "complex",
"persistence_level": "persistent"
},
"task": {
"description": "Analyze chained closure operations where output of one closure becomes input to another.",
"code": "def create_processor(operation):\n    def process(value):\n        if operation == 'add':\n            return value + 10\n        elif operation == 'multiply':\n            return value * 2\n        elif operation == 'square':\n            return value * value\n    return process\n\ndef chain_processors(*processors):\n    def execute(initial_value):\n        result = initial_value\n        for processor in processors:\n            result = processor(result)\n        return result\n    return execute\n\ndef test_chained_closures():\n    add_proc = create_processor('add')\n    mult_proc = create_processor('multiply')\n    square_proc = create_processor('square')\n    chain = chain_processors(add_proc, mult_proc, square_proc)\n    return chain(3)\n\n# What value is returned by test_chained_closures()?",\n"answer": "676"
}
},
{
"id": "MF-SP-S005-V005",
"metadata": {
"name": "MultiFunc-StatePropagation-ClosureMemorization",
"category": "MultiFunc-Level",
"subcategory": "State Propagation",
"type": "variant",
}
}

```

```
"source": "Generated",
"language": "python",
"difficulty": "expert",
"intervention": 3,
"variant_type": "closure_memoization",
"state_type": "closure",
"propagation_pattern": "complex",
"persistence_level": "persistent"
},
"task": {
    "description": "Track closure implementing memoization pattern and analyze cached value propagation.",
    "code": "def create_memoized_function():\n    cache = {}\n    call_count = 0\n\n    def fibonacci(n):\n        nonlocal call_count\n        call_count += 1\n\n        if n in cache:\n            return cache[n]\n\n            if n <= 1:\n                result = n\n            else:\n                result = fibonacci(n-1) + fibonacci(n-2)\n\n                cache[n] = result\n            return result\n\n    def get_cache_size():\n        return len(cache)\n\n    def get_call_count():\n        return call_count\n\n    return fibonacci, get_cache_size, get_call_count\n\ndef test_memoized_closure():\n    fib, cache_size, call_count = create_memoized_function()\n\n    result1 = fib(5)\n    result2 = fib(4)\n\n    return result1 + cache_size() + call_count()\n\n# What value is returned by test_memoized_closure()?",\n    "answer": "20"
},
{
    "id": "MF-SP-S005-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ClosureStateMachine",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "closure_state_machine",
        "state_type": "closure",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
},
    "task": {
        "description": "Analyze closure implementing state machine pattern with state transitions and value accumulation."
    }
}
```

```

"code": "def create_state_machine():\n    state = 'idle'\n    value = 0\n\n    transitions = 0\n    def process(action):\n        nonlocal state, value,\n        transitions\n        if state == 'idle' and action == 'start':\n            state = 'running'\n            value = 10\n            transitions += 1\n        elif state == 'running' and action == 'increment':\n            value += 5\n            transitions += 1\n        elif state == 'running' and action == 'stop':\n            state = 'stopped'\n            value *= 2\n            transitions += 1\n        elif state == 'stopped' and action == 'reset':\n            state = 'idle'\n            value = 0\n            transitions += 1\n\n    return value\n\ndef get_transitions():\n    return transitions\n\nprocess, get_transitions\n\nndef test_state_machine_closure():\n    process, get_trans = create_state_machine()\n    process('start')\n    process('increment')\n    process('increment')\n    result = process('stop')\n    return result + get_trans()\n\n# What value is returned by test_state_machine_closure()?",\n    \"answer\": \"42\"\n}\n},\n{\n    \"id\": \"MF-SP-S005-V007\",\n    \"metadata\": {\n        \"name\": \"MultiFunc-StatePropagation-ClosureEventHandler\",\n        \"category\": \"MultiFunc-Level\",\n        \"subcategory\": \"State Propagation\",\n        \"type\": \"variant\",\n        \"source\": \"Generated\",\n        \"language\": \"python\",\n        \"difficulty\": \"hard\",\n        \"intervention\": 2,\n        \"variant_type\": \"closure_event_handler\",\n        \"state_type\": \"closure\",\n        \"propagation_pattern\": \"complex\",\n        \"persistence_level\": \"persistent\"\n    },\n    \"task\": {\n        \"description\": \"Track closure implementing event handler pattern with event counting and state accumulation.\",\n        \"code\": \"def create_event_handler():\n            events = []\n            total_value = 0\n\n            def handle_event(event_type, value):\n                nonlocal total_value\n                events.append(event_type)\n                if event_type == 'add':\n                    total_value += value\n                elif event_type == 'multiply':\n                    total_value *= value\n                elif event_type == 'reset':\n                    total_value = value\n\n            return total_value\n\n            def get_event_count():\n                return len(events)\n\n        return handle_event, get_event_count()\n\nndef test_event_handler_closure():\n    handle,\n    count = create_event_handler()\n    handle('reset', 8)\n    handle('add', 4)\n    handle('multiply', 3)\n    result = handle('add', 2)\n    return result + count()\n\n# What value is returned by test_event_handler_closure()?",\n        \"answer\": \"42\"\n    }\n},\n{\n    \"id\": \"MF-SP-S005-V008\",\n    \"metadata\": {\n        \"name\": \"MultiFunc-StatePropagation-ClosureConfigurable\",\n        \"category\": \"MultiFunc-Level\",\n        \"subcategory\": \"State Propagation\"\n    }\n}\n
```

```
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "closure_configurable_behavior",
        "state_type": "closure",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze closure with configurable behavior that changes based on captured configuration values.",
        "code": "def create_configurable_function(config):\n    result =\nconfig['initial']\n    mode = config['mode']\n    \n    def execute(value):\n        nonlocal result\n        \n        if mode == 'accumulate':\n            result += value\n        elif mode == 'maximum':\n            result = max(result, value)\n        elif mode == 'product':\n            result *= value\n        \n        return result\n    \n    return execute\n\ndef test_configurable_closure():\n    func1 =\n        create_configurable_function({'initial': 5, 'mode': 'accumulate'})\n    func2 =\n        create_configurable_function({'initial': 2, 'mode': 'product'})\n    \n    result1 = func1(7)\n    result2 = func2(4)\n    result2 = func2(3)\n    \n    return result1 + result2\n\n# what value is returned by test_configurable_closure()?",\n    "answer": "39"
}
},
{
    "id": "MF-SP-S006-V001",
    "metadata": {
        "name": "MultiFunc-StatePropagation-TripleNestedClosure",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "triple_nested_closure",
        "state_type": "closure",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze triple-nested closure state propagation with multiple levels of variable capture."
    }
}
```



```

        "code": "def create_event_system():\n    global_events = []\n        \n    def\ncreate_event_handler(handler_id):\n        local_events = []\n            \n            def\nhandle_event(event_type, value):\n                nonlocal global_events, local_events\n\n                global_events.append(event_type)\n                local_events.append(value)\n\n                if event_type == 'increment':\n                    return\nsum(local_events) + len(global_events)\n                elif event_type == 'reset':\n                    local_events.clear()\n                    return len(global_events)\n                else:\n\n                    return value + handler_id\n\n            return handle_event\n        \n    return create_event_handler\n\n    def test_nested_event_system():\n        create_handler =\n        create_event_system()\n            \n            handler1 = create_handler(10)\n            handler2 =\n        create_handler(20)\n            \n            result1 = handler1('increment', 5)\n            result2 =\n        handler2('increment', 3)\n            result3 = handler1('other', 2)\n            \n            return result1 +\n        result2 + result3\n\n    # What value is returned by test_nested_event_system()?",\n\n    "answer": "31"\n\n}\n\n{\n    "id": "MF-SP-S006-v005",\n    "metadata": {\n        "name": "MultiFunc-StatePropagation-NestedFactoryChain",\n        "category": "MultiFunc-Level",\n        "subcategory": "State Propagation",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "nested_factory_chain",\n        "state_type": "closure",\n        "propagation_pattern": "complex",\n        "persistence_level": "persistent"\n    },\n    "task": {\n        "description": "Analyze nested factory pattern with chained closure creation and\nshared state propagation.",\n        "code": "def create_factory_chain(initial_multiplier):\n    chain_state =\n{'multiplier': initial_multiplier, 'calls': 0}\n\n    def\ncreate_transformer(transform_type):\n        transform_count = 0\n            \n            def\ncreate_processor(processor_value):\n                nonlocal chain_state, transform_count\n\n                chain_state['calls'] += 1\n                transform_count += 1\n\n                if transform_type == 'multiply':\n                    result = input_value *\nchain_state['multiplier'] * processor_value\n                elif transform_type == 'add':\n                    result = input_value + chain_state['multiplier'] + processor_value\n\n                else:\n                    result = input_value + processor_value\n\n                chain_state['multiplier'] += transform_count\n\n                return\nresult\n\n            return process\n        \n    return create_transformer\n\n    def test_factory_chain():\n        factory = create_factory_chain(2)\n            \n            multiply_transformer = factory('multiply')\n            add_transformer = factory('add')\n            \n            mult_processor = multiply_transformer(3)\n            add_processor = add_transformer(5)\n            \n            result1 = mult_processor(4)\n            result2 =\n            add_processor(6)\n            \n            return result1 + result2\n\n    # What value is returned by test_factory_chain()?",\n\n    "answer": "16"

```

```
        "answer": "57"
    }
},
{
    "id": "MF-SP-S006-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-NestedCounter",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "nested_counter_system",
        "state_type": "closure",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track nested counter system where inner counters affect outer counter state.",
        "code": "def create_counter_system(base):\n    total_count = base\n    def create_counter(increment_value):\n        local_count = 0\n        def count():\n            nonlocal total_count, local_count\n            local_count += increment_value\n            total_count += local_count\n            return total_count\n        return count()\n    return create_counter\n\ndef test_nested_counters():\n    create_counter = create_counter_system(5)\n    counter1 = create_counter(3)\n    counter2 = create_counter(2)\n    result1 = counter1()\n    result2 = counter2()\n    result3 = counter1()\n    return result3\n\n# what value is returned by test_nested_counters()?",
        "answer": "19"
    }
},
{
    "id": "MF-SP-S006-V007",
    "metadata": {
        "name": "MultiFunc-StatePropagation-NestedValidator",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_validator_system",
        "state_type": "closure",
        "propagation_pattern": "branched",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze nested validation system with cascading validation rules and state tracking."
    }
}
```

```

        "code": "def create_validator_system(base_threshold):\n    validation_count =\n0\\n    \\n    def create_validator(validator_type):\n        local_validations = 0\\n\n        def validate(value):\n            nonlocal validation_count, local_validations\n            validation_count += 1\\n            local_validations += 1\\n\n            if validator_type == 'min' and value >= base_threshold:\n                return value + validation_count\n            elif validator_type == 'max' and value <=\n                base_threshold:\n                return value + local_validations\n            else:\n                return 0\\n\n        return validate\\n    \\n    return\ncreate_validator\\n\\ndef test_nested_validators():\n    create_validator =\ncreate_validator_system(10)\n    min_validator = create_validator('min')\n    max_validator = create_validator('max')\n    result1 = min_validator(15)\n    result2 = max_validator(8)\n    result3 = min_validator(12)\n    \\n    return result1 + result2 +\nresult3\\n\\n# What value is returned by test_nested_validators()?",\n        "answer": "36"
    }
},
{
    "id": "MF-SP-S006-v008",
    "metadata": {
        "name": "MultiFunc-StatePropagation-NestedGenerator",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "nested_generator_system",
        "state_type": "closure",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track nested generator system with shared sequence state and\nmultiple generation patterns.",\n        "code": "def create_generator_system(seed):\n    global_sequence = [seed]\n\n    def create_generator(pattern):\n        local_position = 0\n        \\n        def\ngenerate():\n            nonlocal global_sequence, local_position\n            \\n            if pattern == 'fibonacci':\n                if len(global_sequence) < 2:\n                    next_val = 1\n                else:\n                    next_val = global_sequence[-1] +\nglobal_sequence[-2]\n            elif pattern == 'double':\n                next_val =\nglobal_sequence[-1] * 2\n            else:\n                next_val = global_sequence[-1] +\n1\n            \\n            global_sequence.append(next_val)\n            local_position +=\n1\n            \\n            return global_sequence[local_position]\n        \\n        return generate\n    \\n    return create_generator\\n\\ndef test_nested_generators():\n    create_gen = create_generator_system(1)\n    fib_gen = create_gen('fibonacci')\n    double_gen = create_gen('double')\n    result1 = fib_gen()\n    result2 =\ndouble_gen()\n    result3 = fib_gen()\n    \\n    return result1 + result2 + result3\\n\\n# What value is returned by test_nested_generators()?",\n        "answer": "6"
    }
},
{

```

```
"id": "MF-SP-S007-V001",
"metadata": {
    "name": "MultiFunc-StatePropagation-SimpleObjectState",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "simple_object_state",
    "state_type": "object",
    "propagation_pattern": "linear",
    "persistence_level": "persistent"
},
"task": {
    "description": "Track simple object state changes through method calls and determine the final attribute value.",
    "code": "class SimpleCounter:\n    def __init__(self, start):\n        self.value = start\n    def add(self, amount):\n        self.value += amount\n    return self.value\n    def multiply(self, factor):\n        self.value *= factor\n    return self.value\n\ndef test_simple_object():\n    counter = SimpleCounter(5)\n    counter.add(7)\n    counter.multiply(2)\n    counter.add(3)\n    return counter.value\n\n# what value is returned by test_simple_object()?",
    "answer": "27"
}
},
{
    "id": "MF-SP-S007-V002",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectStateChaining",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "object_method_chaining",
        "state_type": "object",
        "propagation_pattern": "linear",
        "persistence_level": "persistent"
},
    "task": {
        "description": "Analyze object state propagation through method chaining and calculate the final accumulated result."
    }
}
```

```

        "code": "class calculator:\n    def __init__(self, initial=0):\n        self.result = initial\n        self.operations = 0\n        def add(self, value):\n            self.result += value\n            self.operations += 1\n            return self\n        def multiply(self, value):\n            self.result *= value\n            self.operations += 1\n            return self\n        def subtract(self, value):\n            self.result -= value\n            self.operations += 1\n            return self\n        def get_total(self):\n            return self.result + self.operations\n\n    def test_chained_operations():\n        calc = calculator(10)\n        result = calc.add(5).multiply(3).subtract(8).get_total()\n        return result\n\n    # What value is returned by test_chained_operations()?",\n    "answer": "40"
    }
},
{
    "id": "MF-SP-S007-V003",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectInventorySystem",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "object_inventory_system",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track object inventory system state across multiple operations and calculate the final inventory value.",
        "code": "class Inventory:\n    def __init__(self):\n        self.items = {'apples': 10, 'oranges': 5, 'bananas': 8}\n        self.total_operations = 0\n\n    def add_item(self, item, quantity):\n        if item in self.items:\n            self.items[item] += quantity\n        else:\n            self.items[item] = quantity\n\n        self.total_operations += 1\n        return self.items[item]\n\n    def remove_item(self, item, quantity):\n        if item in self.items:\n            self.items[item] = max(0, self.items[item] - quantity)\n\n            self.total_operations += 1\n            return\n\n        self.items.get(item, 0)\n\n    def get_total_items(self):\n        return sum(self.items.values()) + self.total_operations\n\n    def test_inventory_operations():\n        inventory = Inventory()\n        inventory.add_item('apples', 5)\n        inventory.remove_item('oranges', 3)\n        inventory.add_item('grapes', 12)\n        return\n\n        inventory.get_total_items()\n\n    # What value is returned by test_inventory_operations()?",\n    "answer": "40"
    }
},
{
    "id": "MF-SP-S007-V004",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectStateMachine",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",

```

```

"source": "Generated",
"language": "python",
"difficulty": "expert",
"intervention": 3,
"variant_type": "object_state_machine",
"state_type": "object",
"propagation_pattern": "complex",
"persistence_level": "persistent"
},
"task": {
    "description": "Analyze object implementing state machine pattern with conditional state transitions and value tracking.",
    "code": "class StateMachine:\n    def __init__(self):\n        self.state = 'init'\n        self.value = 0\n        self.transitions = 0\n    def process(self, event, data=0):\n        self.transitions += 1\n        if self.state == 'init' and event == 'start':\n            self.state = 'processing'\n            self.value = data\n        elif self.state == 'processing' and event == 'add':\n            self.value += data\n        elif self.state == 'processing' and event == 'multiply':\n            self.value *= data\n        elif self.state == 'processing' and event == 'finish':\n            self.state = 'complete'\n            self.value += self.transitions\n        elif self.state == 'complete' and event == 'reset':\n            self.state = 'init'\n            self.value = 0\n            self.transitions = 0\n    return self.value\n\ndef test_state_machine_object():\n    sm = StateMachine()\n    sm.process('start', 10)\n    sm.process('add', 5)\n    sm.process('multiply', 2)\n    result = sm.process('finish')\nreturn result\n\n# What value is returned by test_state_machine_object()?",

    "answer": "34"
}
},
{
    "id": "MF-SP-S007-v005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-Objectobserver",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "object_observer_pattern",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track object implementing observer pattern with state change notifications and accumulated tracking."
    }
}

```

```

        "code": "class Observable:\n    def __init__(self, initial_value):\n        self.value = initial_value\n        self.observers = []\n        self.change_count = 0\n\n    def add_observer(self, observer):\n        self.observers.append(observer)\n\n    def set_value(self, new_value):\n        old_value = self.value\n        self.value = new_value\n        self.change_count += 1\n        for observer in self.observers:\n            observer.notify(old_value, new_value)\n\n    def get_total(self):\n        return self.value + self.change_count\n\nclass Observer:\n    def __init__(self):\n        self.notifications = 0\n        self.total_change = 0\n\n    def notify(self, old_val, new_val):\n        self.notifications += 1\n        self.total_change += abs(new_val - old_val)\n\n    def get_stats(self):\n        return self.notifications + self.total_change\n\n\ndef test_observer_pattern():\n    observable = Observable(5)\n    observer = Observer()\n    observable.add_observer(observer)\n    observable.set_value(10)\n    observable.set_value(7)\n    observable.set_value(15)\n\n    return observable.get_total() + observer.get_stats()\n\n# What value is returned by test_observer_pattern()?",\n    "answer": "36"
}
},
{
    "id": "MF-SP-S007-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectComposition",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "object_composition",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze object composition with shared state propagation between composed objects.",
        "code": "class Engine:\n    def __init__(self, power):\n        self.power = power\n        self.running = False\n        self.runtime = 0\n\n    def start(self):\n        self.running = True\n        return self.power\n\n    def run(self, duration):\n        if self.running:\n            self.runtime += duration\n            return self.power\n        return 0\n\n    def stop(self):\n        self.running = False\n        return self.runtime\n\nclass Vehicle:\n    def __init__(self, engine):\n        self.engine = engine\n        self.distance = 0\n        self.fuel_consumed = 0\n\n    def drive(self, time):\n        power_output = self.engine.run(time)\n        if power_output > 0:\n            self.distance += power_output // 10\n            self.fuel_consumed += time\n\n        return self.distance\n\n    def get_efficiency(self):\n        return self.distance + self.fuel_consumed + self.engine.runtime\n\n\ndef test_object_composition():\n    engine = Engine(50)\n    vehicle = Vehicle(engine)\n    engine.start()\n    vehicle.drive(3)\n    vehicle.drive(2)\n    engine.stop()\n\n    return vehicle.get_efficiency()\n\n# What value is returned by test_object_composition()?",\n    "answer": "35"
}
}

```

```

},
{
  "id": "MF-SP-S007-V007",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ObjectFactory",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "object_factory_pattern",
    "state_type": "object",
    "propagation_pattern": "linear",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Track object creation through factory pattern with shared state tracking across instances.",
    "code": "class Counter:\n    instances_created = 0\n    total_operations = 0\n\n    def __init__(self, initial):\n        Counter.instances_created += 1\n\n    self.value = initial\n        self.local_ops = 0\n        def increment(self, amount=1):\n            self.value += amount\n            self.local_ops += 1\n            Counter.total_operations\n            += 1\n            return self.value\n\n        @classmethod\n        def get_global_stats(cls):\n            return cls.instances_created + cls.total_operations\n\n    def create_counter(initial):\n        return Counter(initial)\n\n    def test_factory_pattern():\n        counter1 = create_counter(5)\n        counter2 = create_counter(10)\n        counter1.increment(3)\n        counter2.increment(2)\n        counter1.increment(1)\n        return Counter.get_global_stats() + counter1.value +\n        counter2.value\n\n    # what value is returned by test_factory_pattern()?",
    "answer": "32"
  }
},
{
  "id": "MF-SP-S007-V008",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ObjectPrototype",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "object_prototype_pattern",
    "state_type": "object",
    "propagation_pattern": "complex",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Analyze object prototype pattern with cloning and independent state evolution tracking."
  }
}

```

```

        "code": "class Prototype:\n    def __init__(self, data):\n        self.data =\n        data.copy() if isinstance(data, dict) else data\n        self.modifications = 0\n\n    self.creation_id = id(self)\n\n    def clone(self):\n        new_obj =\n        Prototype(self.data)\n        new_obj.modifications = self.modifications\n        return\n        new_obj\n\n    def modify(self, key, value):\n        if isinstance(self.data, dict):\n            self.data[key] = value\n        else:\n            self.data = value\n\n    self.modifications += 1\n    return self.get_checksum()\n\n    def get_checksum(self):\n        if isinstance(self.data, dict):\n            return\n            sum(self.data.values()) + self.modifications\n        else:\n            return self.data +\n            self.modifications\n\n    def test_prototype_pattern():\n        original = Prototype({'x': 5, 'y': 10})\n        clone1 = original.clone()\n        clone2 = original.clone()\n\n        original.modify('x', 8)\n        clone1.modify('y', 15)\n        result = clone2.modify('x', 12)\n\n        return result\n\n    # What value is returned by test_prototype_pattern()?",\n\n    \"answer\": \"23\"\n\n}\n\n{\n    \"id\": \"MF-SP-S008-v001\",\n\n    \"metadata\": {\n        \"name\": \"MultiFunc-StatePropagation-MultiObjectInteraction\",\n        \"category\": \"MultiFunc-Level\",\n        \"subcategory\": \"State Propagation\",\n        \"type\": \"variant\",\n        \"source\": \"Generated\",\n        \"language\": \"python\",\n        \"difficulty\": \"medium\",\n        \"intervention\": 1,\n        \"variant_type\": \"multi_object_interaction\",\n        \"state_type\": \"object\",\n        \"propagation_pattern\": \"complex\",\n        \"persistence_level\": \"persistent\"\n    },\n\n    \"task\": {\n        \"description\": \"Track shared object state modifications across multiple\n        interconnected objects and determine the final aggregate value.\",\n\n        \"code\": \"class SharedResource:\n            def __init__(self, initial_value):\n                self.value = initial_value\n                self.access_count = 0\n\n            def modify(self, delta):\n                self.value += delta\n                self.access_count += 1\n                return\n                self.value\n\n            def get_total(self):\n                return self.value +\n                self.access_count\n\n        class Worker:\n            def __init__(self, worker_id, resource):\n                self.worker_id = worker_id\n                self.resource = resource\n                self.work_done = 0\n\n            def do_work(self, amount):\n                self.work_done += amount\n                return\n                self.resource.modify(amount * self.worker_id)\n\n        def test_shared_object_state():\n            resource = SharedResource(20)\n            worker1 = Worker(2, resource)\n            worker2 = Worker(3, resource)\n\n            worker1.do_work(5)\n            worker2.do_work(4)\n            worker1.do_work(3)\n\n            return resource.get_total()\n\n        # What value is returned by\n        test_shared_object_state()?\",\n\n        \"answer\": \"45\"\n    }\n\n    {\n        \"id\": \"MF-SP-S008-v002\",\n\n        \"metadata\": {\n\n\n
```

```
        "name": "MultiFunc-StatePropagation-ObjectStateSync",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "object_state_synchronization",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Analyze object state synchronization across multiple objects with bidirectional state updates.",
        "code": "class SyncedCounter:\n    def __init__(self, initial_value,\n     sync_partner=None):\n        self.value = initial_value\n        self.sync_partner = sync_partner\n        self.sync_operations = 0\n        if sync_partner:\n            sync_partner.sync_partner = self\n\n    def increment(self, amount):\n        self.value += amount\n        if self.sync_partner:\n            self.sync_partner.value += amount // 2\n        self.sync_operations += 1\n        return self.value\n\n    def get_combined_value(self):\n        partner_value = self.sync_partner.value if\n        self.sync_partner else 0\n        return self.value + partner_value +\n        self.sync_operations\n\ndef test_synced_objects():\n    counter1 = SyncedCounter(10)\n    counter2 = SyncedCounter(5, counter1)\n    counter1.increment(8)\n    counter2.increment(6)\n    counter1.increment(4)\n    return\n    counter1.get_combined_value()\n\n# What value is returned by test_synced_objects()?",\n    "answer": "37"
},
{
    "id": "MF-SP-S008-V003",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectEventBus",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "object_event_bus",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track object state propagation through event bus system with multiple subscribers and event handling."
    }
}
```

```

        "code": "class EventBus:\n    def __init__(self):\n        self.subscribers = []\n        self.events_published = 0\n    def subscribe(self, subscriber):\n        self.subscribers.append(subscriber)\n    def publish(self, event_type, data):\n        self.events_published += 1\n        for subscriber in self.subscribers:\n            subscriber.handle_event(event_type, data)\n        return self.events_published\n\nclass Subscriber:\n    def __init__(self, subscriber_id):\n        self.subscriber_id = subscriber_id\n        self.events_received = 0\n        self.accumulated_value = 0\n    def handle_event(self, event_type, data):\n        self.events_received += 1\n        if event_type == 'increment':\n            self.accumulated_value += data *\n        self.subscriber_id\n        elif event_type == 'reset':\n            self.accumulated_value = 0\n    def get_total(self):\n        return self.accumulated_value +\n        self.events_received\n\ndef test_event_bus_system():\n    bus = EventBus()\n    sub1 =\n    Subscriber(2)\n    sub2 = Subscriber(3)\n    bus.subscribe(sub1)\n    bus.subscribe(sub2)\n    bus.publish('increment', 5)\n    bus.publish('increment', 3)\n    bus.publish('reset', 0)\n    return sub1.get_total() + sub2.get_total() +\n    bus.events_published\n\n# What value is returned by test_event_bus_system()?",\n        \"answer\": \"15\"\n    }\n},\n{\n    \"id\": \"MF-SP-S008-v004\", \n    \"metadata\": {\n        \"name\": \"MultiFunc-StatePropagation-ObjectChainOfResponsibility\", \n        \"category\": \"MultiFunc-Level\", \n        \"subcategory\": \"State Propagation\", \n        \"type\": \"variant\", \n        \"source\": \"Generated\", \n        \"language\": \"python\", \n        \"difficulty\": \"hard\", \n        \"intervention\": 2, \n        \"variant_type\": \"object_chain_responsibility\", \n        \"state_type\": \"object\", \n        \"propagation_pattern\": \"linear\", \n        \"persistence_level\": \"persistent\"\n    }, \n    \"task\": {\n        \"description\": \"Analyze object state propagation through chain of responsibility pattern with sequential processing.\",\n        \"code\": \"class Handler:\n            def __init__(self, handler_id, threshold):\n                self.handler_id = handler_id\n                self.threshold = threshold\n                self.next_handler = None\n                self.processed_count = 0\n                self.accumulated_value = 0\n            def set_next(self, handler):\n                self.next_handler = handler\n                return handler\n            def handle(self, request):\n                self.processed_count += 1\n                if request >= self.threshold:\n                    self.accumulated_value += request *\n                if self.handler_id:\n                    return self.accumulated_value\n                elif self.next_handler:\n                    return self.next_handler.handle(request)\n                else:\n                    return 0\n            def get_stats(self):\n                return self.accumulated_value +\n                self.processed_count\n\ndef test_chain_of_responsibility():\n    handler1 = Handler(1, 20)\n    handler2 = Handler(2, 10)\n    handler3 = Handler(3, 5)\n    handler1.set_next(handler2).set_next(handler3)\n    result1 = handler1.handle(25)\n    result2 = handler1.handle(15)\n    result3 = handler1.handle(8)\n    return\n    handler1.get_stats() + handler2.get_stats() + handler3.get_stats()\n\n# What value is returned by test_chain_of_responsibility()?\",\n        \"answer\": \"45\"\n    }\n}\n
```

```

        "answer": "84"
    }
},
{
    "id": "MF-SP-S008-V005",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectMediator",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "object_mediator_pattern",
        "state_type": "object",
        "propagation_pattern": "complex",
        "persistence_level": "persistent"
    },
    "task": {
        "description": "Track object state coordination through mediator pattern with centralized communication management.",
        "code": "class Mediator:\n    def __init__(self):\n        self.components = []\n        self.messages_handled = 0\n        self.total_data = 0\n    def register(self, component):\n        self.components.append(component)\n    component.mediator = self\n    def notify(self, sender, event, data):\n        self.messages_handled += 1\n        self.total_data += data\n        for component in self.components:\n            if component != sender:\n                component.receive_notification(event, data)\n        return\n    self.total_data\n\nclass Component:\n    def __init__(self, component_id):\n        self.component_id = component_id\n        self.mediator = None\n        self.notifications_received = 0\n        self.local_value = 0\n    def send_message(self, event, data):\n        if self.mediator:\n            return\n        self.mediator.notify(self, event, data)\n        return 0\n    def receive_notification(self, event, data):\n        self.notifications_received += 1\n        self.local_value += data // self.component_id\n    def get_state(self):\n        return self.local_value + self.notifications_received\n\ndef test_mediator_pattern():\n    mediator = Mediator()\n    comp1 = Component(2)\n    comp2 = Component(3)\n    comp3 = Component(4)\n    mediator.register(comp1)\n    mediator.register(comp2)\n    mediator.register(comp3)\n    comp1.send_message('update', 12)\n    comp2.send_message('change', 18)\n    return comp1.get_state() + comp2.get_state() + comp3.get_state()\n\n# What value is returned by test_mediator_pattern()?",\n    "answer": "26"
}
},
{
    "id": "MF-SP-S008-V006",
    "metadata": {
        "name": "MultiFunc-StatePropagation-ObjectPooling",
        "category": "MultiFunc-Level",
        "subcategory": "State Propagation",
        "type": "variant",
        "source": "Generated",
    }
}

```

```

    "language": "python",
    "difficulty": "hard",
    "intervention": 2,
    "variant_type": "object_pooling",
    "state_type": "object",
    "propagation_pattern": "circular",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Analyze object pooling system with reusable objects and state persistence across usage cycles.",
    "code": "class PooledObject:\n    def __init__(self, obj_id):\n        self.obj_id = obj_id\n        self.usage_count = 0\n        self.accumulated_work = 0\n        self.in_use = False\n\n    def activate(self):\n        self.in_use = True\n\n    def do_work(self, work_amount):\n        if self.in_use:\n            self.accumulated_work += work_amount * self.obj_id\n        return self.accumulated_work\n\n    def deactivate(self):\n        self.in_use = False\n        return self.usage_count\n\n    def get_total_value(self):\n        return self.accumulated_work + self.usage_count\n\nclass ObjectPool:\n    def __init__(self):\n        self.objects = [PooledObject(i) for i in range(1, 4)]\n        self.checkout_count = 0\n\n    def get_object(self):\n        for obj in self.objects:\n            if not obj.in_use:\n                obj.activate()\n\n        self.checkout_count += 1\n        return obj\n\n    def return_object(self, obj):\n        obj.deactivate()\n\n    def test_object_pooling():\n        pool = ObjectPool()\n        obj1 = pool.get_object()\n        obj1.do_work(5)\n        obj2 = pool.get_object()\n        obj2.do_work(3)\n\n        pool.return_object(obj1)\n        obj3 = pool.get_object()\n        obj3.do_work(2)\n\n        return obj1.get_total_value() +\n               obj2.get_total_value() + obj3.get_total_value()\n\n\n# What value is returned by test_object_pooling()?",\n    "answer": "19"
  }
},
{
  "id": "MF-SP-S008-V007",
  "metadata": {
    "name": "MultiFunc-StatePropagation-ObjectRegistry",
    "category": "MultiFunc-Level",
    "subcategory": "State Propagation",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "object_registry_system",
    "state_type": "object",
    "propagation_pattern": "complex",
    "persistence_level": "persistent"
  },
  "task": {
    "description": "Track object registration system with centralized object tracking and state aggregation."
  }
}

```

```

    "code": "class ObjectRegistry:\n        def __init__(self):\n            self.registered_objects = {}\n            self.registration_count = 0\n            def register(self, obj_id, obj):\n                self.registered_objects[obj_id] = obj\n                self.registration_count += 1\n                obj.registry_id = obj_id\n                return obj_id\n\n        def get_aggregate_value(self):\n            total = 0\n            for obj in self.registered_objects.values():\n                total += obj.get_value()\n            return total + self.registration_count\n\nclass RegisteredObject:\n    def __init__(self, initial_value):\n        self.value = initial_value\n        self.registry_id = None\n        self.operations = 0\n\n    def update(self, delta):\n        self.value += delta\n        self.operations += 1\n        return self.value\n\n    def get_value(self):\n        multiplier = self.registry_id if self.registry_id else 1\n        return self.value * multiplier + self.operations\n\n\ndef test_object_registry():\n    registry = ObjectRegistry()\n\n    obj1 = RegisteredObject(5)\n    obj2 = RegisteredObject(8)\n\n    registry.register(2, obj1)\n    registry.register(3, obj2)\n\n    obj1.update(3)\n    obj2.update(2)\n    obj1.update(1)\n\n    return registry.get_aggregate_value()\n\n#\n# What value is returned by test_aggregate_registry()?",\n\n    \"answer\": \"55\"\n}\n},\n{\n    \"id\": \"MF-SP-S008-v008\", \n    \"metadata\": {\n        \"name\": \"MultiFunc-StatePropagation-ObjectCommand\", \n        \"category\": \"MultiFunc-Level\", \n        \"subcategory\": \"State Propagation\", \n        \"type\": \"variant\", \n        \"source\": \"Generated\", \n        \"language\": \"python\", \n        \"difficulty\": \"expert\", \n        \"intervention\": 3, \n        \"variant_type\": \"object_command_pattern\", \n        \"state_type\": \"object\", \n        \"propagation_pattern\": \"complex\", \n        \"persistence_level\": \"persistent\"\n    }, \n    \"task\": {\n        \"description\": \"Analyze object command pattern with command execution tracking and state modification history.\"\n    }\n}
```

```

"code": "class Command:\n    def __init__(self, command_id):\n        self.command_id = command_id\n        self.executed = False\n        self.execution_count = 0\n    def execute(self, receiver):\n        self.executed = True\n        self.execution_count += 1\n        return receiver.perform_action(self.command_id)\n\nclass Receiver:\n    def __init__(self):\n        self.state = 0\n        self.actions_performed = 0\n        self.command_history = []\n    def perform_action(self, action_id):\n        self.actions_performed += 1\n        self.command_history.append(action_id)\n        self.state += action_id * len(self.command_history)\n        return self.state\n\nclass Invoker:\n    def __init__(self):\n        self.commands = []\n        self.invocations = 0\n    def add_command(self, command):\n        self.commands.append(command)\n    def execute_commands(self, receiver):\n        results = []\n        for command in self.commands:\n            self.invocations += 1\n            results.append(command.execute(receiver))\n        return sum(results)\n\ndef test_command_pattern():\n    receiver = Receiver()\n    invoker = Invoker()\n    cmd1 = Command(3)\n    cmd2 = Command(5)\n    cmd3 = Command(2)\n    invoker.add_command(cmd1)\n    invoker.add_command(cmd2)\n    invoker.add_command(cmd3)\n    invoker.execute_commands(receiver)\n    return receiver.get_total_state() + cmd1.get_stats() + cmd2.get_stats() + cmd3.get_stats()\n# What value is returned by test_command_pattern()?",\n    \"answer\": \"67\"\n}\n}\n]"

```

4D - 副作用推理 [Side Effect] (59)

副作用推理变式生成提示词

任务目标

基于给定的副作用推理种子任务，为每个种子生成8-12个多样化的变式任务，全面测试大模型对跨函数副作用机制的理解能力，包括内存修改、I/O操作、全局状态改变和异常处理等核心副作用推理技能。

副作用推理特征分析

副作用推理关注函数执行过程中除返回值外的所有外部可观察变化，重点测试模型对内存修改、外部状态变更、资源操作和隐式影响的深层理解，强调对函数副作用的识别、追踪和分析能力。

关键要求

- **描述语言**：所有task描述必须使用英文
- **变式数量**：每个种子任务严格生成8-12个变式，确保数量充足
- **答案唯一性**：保证答案准确且唯一，避免歧义和多解情况
- **副作用聚焦**：重点关注函数的副作用分析，强调隐式效果和外部影响

变式生成维度

1. 内存修改副作用分析变式

- **指针修改副作用变式**：通过指针修改内存内容的副作用分析
- **数组元素修改变式**：数组元素修改的副作用传播
- **结构体成员修改变式**：结构体成员修改的副作用影响
- **动态内存副作用变式**：动态内存分配释放的副作用
- **内存重叠修改变式**：内存重叠区域修改的副作用
- **缓冲区溢出副作用变式**：缓冲区操作的潜在副作用

2. 全局状态修改分析变式

- **全局变量修改变式**：函数对全局变量的副作用修改
- **静态变量副作用变式**：静态变量修改的持久性副作用
- **全局数据结构变式**：全局数据结构修改的副作用
- **单例状态修改变式**：单例对象状态修改的副作用
- **环境变量修改变式**：环境变量修改的系统级副作用
- **配置状态修改变式**：全局配置状态的修改副作用

3. I/O操作副作用分析变式

- **文件写入副作用变式**：文件写入操作的副作用分析
- **标准输出副作用变式**：标准输出的副作用影响
- **网络通信副作用变式**：网络操作的副作用分析
- **数据库操作副作用变式**：数据库操作的持久化副作用
- **日志记录副作用变式**：日志记录的副作用影响
- **缓存操作副作用变式**：缓存操作的副作用分析

4. 资源管理副作用分析变式

- **文件句柄副作用变式**：文件句柄操作的资源副作用
- **内存泄漏副作用变式**：内存泄漏的资源副作用
- **锁获取释放变式**：锁操作的并发副作用
- **信号量操作变式**：信号量操作的同步副作用
- **线程创建副作用变式**：线程创建的系统资源副作用
- **连接池副作用变式**：连接池操作的资源副作用

5. 异常和错误副作用分析变式

- **异常抛出副作用变式**：异常抛出的控制流副作用
- **错误状态设置变式**：错误状态设置的副作用
- **资源清理副作用变式**：异常处理中的资源清理副作用
- **堆栈展开副作用变式**：异常堆栈展开的副作用
- **错误传播副作用变式**：错误传播的连锁副作用
- **恢复机制副作用变式**：错误恢复机制的副作用

6. 时间和随机性副作用变式

- **时间戳副作用变式**：时间相关操作的副作用
- **随机数生成变式**：随机数生成器状态的副作用
- **计时器设置变式**：计时器操作的副作用
- **延迟操作副作用变式**：延迟操作的时序副作用
- **调度影响副作用变式**：任务调度的副作用影响
- **性能统计副作用变式**：性能统计收集的副作用

7. 并发和同步副作用分析变式

- **竞争条件副作用变式**：竞争条件产生的副作用
- **死锁风险副作用变式**：死锁风险的副作用分析
- **原子操作副作用变式**：原子操作的并发副作用
- **内存顺序副作用变式**：内存顺序约束的副作用
- **线程同步副作用变式**：线程同步机制的副作用
- **数据竞争副作用变式**：数据竞争的副作用影响

8. 系统调用副作用分析变式

- **系统调用副作用变式**：系统调用的系统级副作用
- **进程创建副作用变式**：进程创建的系统资源副作用
- **权限修改副作用变式**：权限修改的安全副作用

- **系统配置副作用变式**：系统配置修改的副作用
- **设备操作副作用变式**：设备操作的硬件副作用
- **中断处理副作用变式**：中断处理的副作用影响

9. 函数式副作用分析变式

- **闭包副作用变式**：闭包捕获变量的副作用
- **高阶函数副作用变式**：高阶函数的间接副作用
- **惰性求值副作用变式**：惰性求值的延迟副作用
- **记忆化副作用变式**：记忆化缓存的副作用
- **尾调用副作用变式**：尾调用优化的副作用
- **函数组合副作用变式**：函数组合的累积副作用

10. 高级副作用模式变式

- **观察者模式副作用变式**：观察者模式的通知副作用
- **代理模式副作用变式**：代理模式的间接副作用
- **装饰器副作用变式**：装饰器模式的额外副作用
- **中间件副作用变式**：中间件的处理副作用
- **事件发布副作用变式**：事件发布订阅的副作用
- **AOP副作用变式**：面向切面编程的副作用

复杂度层次设计

简单副作用分析 (Easy)

- 单一明确的副作用操作
- 直接的内存或状态修改
- 清晰可见的副作用结果
- 基础的副作用概念理解

中等副作用分析 (Medium)

- 2-3种副作用的组合
- 包含条件的副作用触发
- 中等复杂度的副作用传播
- 需要理解副作用的间接影响

复杂副作用分析 (Hard)

- 多种副作用的复杂交互
- 深层的副作用传播链
- 隐式和间接的副作用
- 涉及系统级的副作用分析

专家级副作用分析 (Expert)

- 极复杂的副作用网络
- 并发环境的副作用分析
- 需要深度系统知识的副作用理解
- 涉及性能和安全的副作用

生成策略

种子分析策略

1. **识别副作用类型**：分析种子任务的主要副作用类型
2. **提取影响范围**：识别副作用的影响范围和传播路径
3. **确定可观察性**：分析副作用的可观察性和检测方法
4. **评估复杂度层次**：评估副作用分析的复杂程度

变式设计原则

1. **副作用导向**：每个变式都应明确关注函数的副作用
2. **英文描述**：所有**task**描述必须使用标准英文
3. **答案唯一**：严格确保答案的准确性和唯一性
4. **数量保证**：严格确保每个种子生成8-12个变式

质量保证

1. **副作用语义验证**：验证副作用操作的语义正确性
2. **影响范围检查**：确保副作用影响范围分析的准确性
3. **答案唯一性验证**：严格检查答案的唯一性和确定性
4. **英文质量保证**：确保描述的英文表达准确清晰

输出格式要求

```
```json
[
 {
 "id": "MF-SE-S00X-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-VariantName",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "target_language",
 "difficulty": "easy/medium/hard/expert",
 "intervention": 0,
 "variant_type": "variant_type_label",
 "side_effect_type": "memory/io/state/resource/exception",
 "effect_scope": "local/global/system/external",
 "observability": "direct/indirect/delayed/hidden"
 },
 "task": {
 "description": "English description of the side effect analysis task",
 "code": "Code containing side effect operations to analyze",
 "answer": "Unique and accurate side effect analysis result"
 }
 },
 {下一个变式...}
]

```

### 特殊字段说明

**side\_effect\_type**: 标识主要的副作用类型

**effect\_scope**: 标识副作用的影响范围

**observability**: 标识副作用的可观察性级别

### 生成目标

为每个提供的种子任务严格生成8-12个副作用推理变式，确保：

每种难度等级至少包含2-3个变式

涵盖至少6-8种不同的副作用分析类型

包含不同的副作用范围和复杂度

所有**task**描述使用标准英文表达

严格保证答案的准确性和唯一性

## 重点测试场景

- 副作用识别：准确识别函数执行的所有副作用
- 影响范围分析：分析副作用的影响范围和传播路径
- 副作用顺序：理解多个副作用的执行顺序和依赖
- 隐式副作用：识别隐式和间接的副作用
- 副作用组合：分析多个副作用的组合效应
- 副作用持久性：理解副作用的持久性和生存期

## 英文描述模板示例

```
"Analyze the side effects of function execution and determine..."
"Identify all observable side effects after the function calls..."
"Determine the external state changes caused by the function..."
"Calculate the cumulative side effects of the function sequence..."
"Trace the side effect propagation through the function chain..."
"Evaluate the persistent changes caused by function side effects..."
```

## 答案唯一性保证策略

明确副作用定义：清晰定义什么构成可观察的副作用

具体状态值：使用具体的状态值，避免抽象描述

确定性操作：确保副作用操作是确定性的

避免竞争条件：避免产生竞争条件的副作用

清晰时序：确保副作用的时序和顺序明确

## 常见副作用模式

写入操作：对外部存储或状态的写入修改

输出操作：产生可观察输出的操作

资源分配：系统资源的分配和释放

状态修改：全局或共享状态的修改

事件触发：触发外部事件或通知

缓存更新：缓存或临时存储的更新

请基于此提示词，为给定的副作用推理种子任务生成全面的变式集合，严格确保每个种子8-12个变式，所有描述使用英文，保证答案准确且唯一，输出格式为包含所有变式的JSON数组。

[

```
{
 "id": "MF-SE-S001-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-BufferSizeTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "buffer_size_modification",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
},
 "task": {
 "description": "Analyze buffer size modification side effects. What is the final
value of buffer_size?",
 }

```

```
 "code": "int buffer_size = 1024;\n\nint expand_buffer(int additional_size) {\n buffer_size += additional_size;\n printf(\"Buffer expanded by %d bytes\\n\", additional_size);\n return buffer_size;\n}\n\nint process_expansion() {\n expand_buffer(256);\n expand_buffer(512);\n expand_buffer(128);\n return buffer_size;\n}\n\n// What value is returned by process_expansion()?",\n "answer": "1920"
 },
},
{
 "id": "MF-SE-S001-V002",
 "metadata": {
 "name": "MultiFunc-SideEffect-OutputCountTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "output_counting",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track output operation side effects. What is the final value of output_count?",\n "code": "int output_count = 0;\n\nint print_message(const char* message) {\n output_count++;\n printf(\"%s\\n\", message);\n return strlen(message);\n}\n\nint print_numbered_list(const char* items[], int count) {\n for (int i = 0; i < count; i++)\n print_message(items[i]);\n return output_count;\n}\n\nexecute_printing() {\n const char* messages[] = {\"Hello\", \"World\", \"Test\", \"Done\"};\n print_numbered_list(messages, 4);\n print_message(\"Final\");\n return output_count;\n}\n\n// What value is returned by execute_printing()?",\n "answer": "5"
 },
},
{
 "id": "MF-SE-S001-V003",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalWriting",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_io",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
}
```

```

"task": {
 "description": "Analyze conditional I/O side effects. What is the final value of chars_processed?",
 "code": "int chars_processed = 0;\n\nint conditional_write(const char* data, int threshold) {\n int len = strlen(data);\n if (len > threshold) {\n chars_processed += len;\n printf(\"writing: %s\\n\", data);\n return len;\n }\n return 0;\n}\n\nint process_conditional() {\n conditional_write(\"Short\", 10);\n // 5 chars, threshold 10\n conditional_write(\"Medium length\", 8); // 13 chars, threshold 8\n conditional_write(\"Hi\", 5); // 2 chars, threshold 5\n conditional_write(\"Very long string here\", 15); // 20 chars, threshold 15\n return chars_processed;\n}\n\n// what value is returned by process_conditional()?",
 "answer": "33"
},
{
 "id": "MF-SE-S001-V004",
 "metadata": {
 "name": "MultiFunc-SideEffect-NestedIOOperations",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_io",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track nested I/O operation side effects. What is the final value of total_operations?",
 "code": "int total_operations = 0;\n\nint log_write(const char* data) {\n total_operations++;\n printf(\"LOG: %s\\n\", data);\n return total_operations;\n}\n\nint debug_write(const char* data) {\n total_operations++;\n printf(\"DEBUG: %s\\n\", data);\n log_write(\"Debug operation logged\");\n return total_operations;\n}\n\nint error_write(const char* data) {\n total_operations++;\n printf(\"ERROR: %s\\n\", data);\n debug_write(\"Error debugging initiated\");\n log_write(\"Error operation completed\");\n return total_operations;\n}\n\nint execute_nested_ops() {\n error_write(\"Critical failure\");\n return total_operations;\n}\n\n// what value is returned by execute_nested_ops()?",
 "answer": "5"
 }
},
{
 "id": "MF-SE-S001-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-BatchIOProcessing",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "batch_io",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 }
}

```

```

 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "batch_processing",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze batch I/O processing side effects. What is the final value of batch_size?",
 "code": "int batch_size = 0;\nint batches_processed = 0;\n\nint add_to_batch(int size) {\n batch_size += size;\n if (batch_size >= 100) {\n batches_processed++;\n printf(\"Batch %d processed, size: %d\\n\", batches_processed, batch_size);\n batch_size = 0;\n }\n return batch_size;\n}\n\nint process_items() {\n add_to_batch(30);\n add_to_batch(40);\n add_to_batch(50); // Triggers batch\n add_to_batch(25);\n add_to_batch(60);\n add_to_batch(20); // Triggers batch\n add_to_batch(15);\n return batch_size;\n}\n\n// What value is returned by process_items()?",
 "answer": "15"
 }
},
{
 "id": "MF-SE-S001-v006",
 "metadata": {
 "name": "MultiFunc-SideEffect-IOStateTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "state_tracking",
 "side_effect_type": "state",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track I/O state modification side effects. What is the final value of io_state?",
 "code": "int io_state = 0; // 0=idle, 1=reading, 2=writing, 3=error\n\nint start_read() {\n io_state = 1;\n printf(\"Started reading\\n\");\n return io_state;\n}\n\nint start_write() {\n io_state = 2;\n printf(\"Started writing\\n\");\n return io_state;\n}\n\nint finish_operation() {\n io_state = 0;\n printf(\"Operation finished\\n\");\n return io_state;\n}\n\nint perform_io_sequence() {\n start_read();\n finish_operation();\n start_write();\n return io_state;\n}\n\n// What value is returned by perform_io_sequence()?",
 "answer": "2"
 }
},
{
 "id": "MF-SE-S001-v007",

```

```
"metadata": {
 "name": "MultiFunc-SideEffect-AccumulativeIO",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "accumulative_effects",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
},
"task": {
 "description": "Analyze accumulative I/O side effects. What is the final value of total_bytes?",
 "code": "int total_bytes = 0;\nint write_count = 0;\n\nint\nwrite_with_header(const char* data) {\n int header_size = 10;\n int data_size =\n strlen(data);\n total_bytes += header_size + data_size;\n write_count++;\n printf(\"write %d: %d bytes\\n\", write_count, header_size + data_size);\n return\n total_bytes;\n}\n\nint\nwrite_formatted_data() {\n write_with_header(\"ABC\"); // 10\n + 3 = 13\n write_with_header(\"DEFGH\"); // 10 + 5 = 15\n write_with_header(\"IJK\"); // 10 + 3 = 13\n return total_bytes;\n}\n\n// What value is returned by write_formatted_data()?",
 "answer": "41"
},
{
 "id": "MF-SE-S001-v008",
 "metadata": {
 "name": "MultiFunc-SideEffect-IOErrorHandler",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "error_handling",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
},
 "task": {
 "description": "Track I/O error handling side effects. What is the final value of error_bytes?",
 }
}
```

```

 "code": "int error_bytes = 0;\nint successful_writes = 0;\n\nint\nsafe_write(const char* data, int simulate_error) {\n int len = strlen(data);\n if\n(simulate_error) {\n error_bytes += len;\n printf(\"Write failed: %d\nbytes\\n\", len);\n return 0;\n } else {\n successful_writes++;\n printf(\"Write success: %d bytes\\n\", len);\n return len;\n }\n}\n\nint\nperform_writes() {\n safe_write(\"Data1\", 0); // Success: 5 bytes\n safe_write(\"Data22\", 1); // Error: 6 bytes\n safe_write(\"Data333\", 0); // Success: 7\n bytes\n safe_write(\"Data4444\", 1); // Error: 8 bytes\n return error_bytes;\n}\n\n//\n// what value is returned by perform_writes()?",\n "answer": "14"
 },
},
{
 "id": "MF-SE-S001-V009",
 "metadata": {
 "name": "MultiFunc-SideEffect-MultistreamIO",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multi_stream",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Analyze multi-stream I/O side effects. What is the final value of stream_a_bytes?",
 "code": "int stream_a_bytes = 0;\nint stream_b_bytes = 0;\nint current_stream = 0; // 0=A, 1=B\n\nint switch_stream() {\n current_stream = 1 - current_stream;\n printf(\"Switched to stream %c\\n\", 'A' + current_stream);\n return current_stream;\n}\n\nint write_to_current_stream(const char* data) {\n int len = strlen(data);\n if (current_stream == 0) {\n stream_a_bytes += len;\n } else {\n stream_b_bytes += len;\n }\n printf(\"Stream %c: %d bytes\\n\", 'A' + current_stream, len);\n return len;\n}\n\nint multi_stream_operations() {\n write_to_current_stream(\"Start\"); // Stream A: 5\n switch_stream();\n write_to_current_stream(\"Middle\"); // Stream B: 6\n switch_stream();\n write_to_current_stream(\"End\"); // Stream A: 3\n return stream_a_bytes;\n}\n\n//\n// what value is returned by multi_stream_operations()?",\n "answer": "8"
 },
},
{
 "id": "MF-SE-S001-V010",
 "metadata": {
 "name": "MultiFunc-SideEffect-IOBufferManagement",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3
 }
}

```

```

 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "buffer_management",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Track I/O buffer management side effects. What is the final value of flush_count?",
 "code": "int buffer_used = 0;\nint flush_count = 0;\nconst int BUFFER_SIZE = 50;\n\nint add_to_buffer(const char* data) {\n int len = strlen(data);\n buffer_used += len;\n if (buffer_used >= BUFFER_SIZE) {\n flush_count++;\n printf(\"Buffer flushed (flush %d)\\n\", flush_count);\n buffer_used = 0;\n }\n return buffer_used;\n}\n\nnint buffered_operations() {\n add_to_buffer(\"First data chunk here\"); // 20 chars\n add_to_buffer(\"Second chunk\"); // 12 chars, total 32\n add_to_buffer(\"Third chunk of data\"); // 18 chars, total 50 -> flush\n add_to_buffer(\"Fourth\"); // 6 chars\n add_to_buffer(\"Fifth chunk here now\"); // 19 chars, total 25\n add_to_buffer(\"Sixth and final chunk\"); // 20 chars, total 45\n add_to_buffer(\"Last\"); // 4 chars, total 49\n add_to_buffer(\"X\"); // 1 char, total 50 -> flush\n return flush_count;\n}\n\n// What value is returned by buffered_operations()?",
 "answer": "2"
 }
},
{
 "id": "MF-SE-S002-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-DocumentLineTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "line_tracking",
 "side_effect_type": "state",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track document line counting side effects. What is the final line count?",
 "code": "line_count = 0\n\ndef add_line(content):\n global line_count\n line_count += 1\n print(f\"Line {line_count}: {content}\")\n\ndef add_paragraph(sentences):\n for sentence in sentences:\n add_line(sentence)\n\nadd_line(\"\") # Empty line after paragraph\n\nreturn line_count\n\n\ndef create_document():\n add_paragraph([\\"First sentence.\", \\"Second sentence.\"])\n add_paragraph([\\"Third sentence.\"])\n return line_count\n\n# What value is returned by create_document()?",
 "answer": "5"
 }
}

```

```
 }
 },
{
 "id": "MF-SE-S002-V002",
 "metadata": {
 "name": "MultiFunc-SideEffect-Conditionalwriting",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_writing",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional file writing side effects. What is the final line count?",
 "code": "line_count = 0\n\ndef write_if_valid(content, min_length):\n global line_count\n if len(content) >= min_length:\n line_count += 1\n\nprint(f\"Line {line_count}: {content}\")\n\n return line_count\n\ndef process_content():\n items = [\"Hi\", \"Hello world\", \"OK\", \"This is a longer sentence\", \"No\"]\n for item in items:\n write_if_valid(item, 5) # Minimum 5 characters\n return line_count\n\n# what value is returned by process_content()?",
 "answer": "2"
 }
},
{
 "id": "MF-SE-S002-V003",
 "metadata": {
 "name": "MultiFunc-SideEffect-NestedSections",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_sections",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track nested section writing side effects. What is the final line count?",
 }
}
```

```

 "code": "line_count = 0\n\ndef write_line(content):\n global line_count\n line_count += 1\n print(f\"Line {line_count}: {content}\")\n return line_count\n\ndef write_section(title, content_lines):\n write_line(f\"--- {title} ---\")\n for content in content_lines:\n write_line(content)\n write_line(\"\") # Empty line after section\n return line_count\n\ndef write_subsection(title, content_lines):\n write_line(f\"--- {title} ---\")\n for content in content_lines:\n write_line(content)\n return line_count\n\ndef create_nested_document():\n write_section(\"Main\", [\"Introduction\"])\n write_subsection(\"Details\", [\"Point 1\", \"Point 2\"])\n write_line(\"Conclusion\")\n return line_count\n\n# What value is returned by create_nested_document()?",\n "answer": "8"
 },
 {
 "id": "MF-SE-S002-V004",
 "metadata": {
 "name": "MultiFunc-SideEffect-NumberedListGeneration",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "numbered_lists",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze numbered list generation side effects. What is the final line count?",\n "code": "line_count = 0\nitem_number = 0\n\ndef write_numbered_item(content):\n global line_count, item_number\n item_number += 1\n line_count += 1\n print(f\"Line {line_count}: {item_number}. {content}\")\n\ndef write_numbered_list(items):\n for item in items:\n write_numbered_item(item)\n\nreturn line_count\n\ndef generate_lists():\n write_numbered_list([\"First\", \"Second\"])\n write_numbered_list([\"Third\", \"Fourth\", \"Fifth\"])\n return line_count\n\n# What value is returned by generate_lists()?",\n "answer": "5"
 },
 {
 "id": "MF-SE-S002-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-FormattedOutput",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 }
 }
 }
]
```

```

 "variant_type": "formatted_output",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track formatted output side effects. What is the final line count?",
 "code": "line_count = 0\n\ndef write_formatted_line(template, *args):\n global line_count\n line_count += 1\n content = template.format(*args)\n print(f\"Line {line_count}: {content}\")\n return len(content)\n\ndef write_table_row(columns):\n template = \"| {} | {} | {} | {} |\n return write_formatted_line(template, *columns)\n\ndef create_table():\n write_formatted_line(\"| {} | {} | {} | {} |\", \"Name\", \"Age\", \"City\")\n write_formatted_line(\"| {} | {} | {} | {} |\", \"---\", \"---\", \"---\")\n write_table_row(['Alice', '25', 'NYC'])\n write_table_row(['Bob', '30', 'LA'])\n return line_count\n\n# What value is returned by create_table()?",
 "answer": "4"
 }
},
{
 "id": "MF-SE-S002-V006",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalFormatting",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_formatting",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional formatting side effects. What is the final line count?",
 "code": "line_count = 0\n\ndef write_formatted_item(item, use_bullets):\n global line_count\n line_count += 1\n if use_bullets:\n content = f\"• {item}\n else:\n content = f\"{line_count}. {item}\n print(f\"Line {line_count}: {content}\")\n return line_count\n\ndef write_mixed_list():\n items = ['Apple', 'Banana', 'Cherry']\n for i, item in enumerate(items):\n use_bullets = (i % 2 == 0) # Even indices use bullets\n write_formatted_item(item, use_bullets)\n return line_count\n\n# What value is returned by write_mixed_list()?",
 "answer": "3"
 }
},
{
 "id": "MF-SE-S002-V007",
 "metadata": {
 "name": "MultiFunc-SideEffect-MultiDocumentGeneration",

```

```
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "multi_document",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track multi-document generation side effects. What is the final total_lines?",
 "code": "total_lines = 0\ncurrent_doc_lines = 0\n\nstart_new_document(title):\n global total_lines, current_doc_lines\n current_doc_lines = 0\n total_lines += 1\n current_doc_lines += 1\n print(f'== Document: {title} ==')\n\n return total_lines\n\nndef add_document_line(content):\n global total_lines,\n current_doc_lines\n total_lines += 1\n current_doc_lines += 1\n print(f'Doc line {current_doc_lines}: {content}')\n\n return total_lines\n\nndef generate_documents():\n start_new_document(\"Report A\")\n add_document_line(\"Content A1\")\n\n add_document_line(\"Content A2\")\n\n start_new_document(\"Report B\")\n\n add_document_line(\"Content B1\")\n\n return total_lines\n\n# What value is returned by generate_documents()?",
 "answer": "5"
 }
},
{
 "id": "MF-SE-S002-V008",
 "metadata": {
 "name": "MultiFunc-SideEffect-TemplateProcessing",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "template_processing",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze template processing side effects. What is the final line count?",
 }
}
```



```

 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "hierarchical_structure",
 "side_effect_type": "io",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track hierarchical structure generation side effects. What is the final line count?",
 "code": "line_count = 0\nindent_level = 0\n\ndef increase_indent():\n global indent_level\n indent_level += 1\n\ndef decrease_indent():\n global indent_level\nif indent_level > 0:\n indent_level -= 1\n\ndef writeIndentedLine(content):\n global line_count\n line_count += 1\n indent = ' ' * indent_level\n print(f'Line {line_count}: {indent}{content}\n')\n return line_count\n\ndef writeHierarchy():\n writeIndentedLine(\"Root\")\n increase_indent()\n writeIndentedLine(\"Child 1\")\n increase_indent()\n writeIndentedLine(\"Grandchild 1\")\n decrease_indent()\n writeIndentedLine(\"Child 2\")\n decrease_indent()\n writeIndentedLine(\"Root 2\")\n\nreturn line_count\n\n# What value is returned by write_hierarchy()?",

 "answer": "5"
 }
},
{
 "id": "MF-SE-S003-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-BasicExceptionCounting",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "exception_counting",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track basic exception counting side effects. What is the final exception_count?",
 "code": "exception_count = 0\n\ndef safe_divide(a, b):\n global exception_count\n try:\n return a / b\n except ZeroDivisionError:\n exception_count += 1\n return 0\n\ndef performDivisions():\n safe_divide(10, 2)\n\n# No exception\n safe_divide(8, 0) # Exception\n safe_divide(15, 3) # No exception\n safe_divide(7, 0) # Exception\n\n return exception_count\n\n# What value is returned by performDivisions()?",

 "answer": "2"
 }
},
{
 "id": "MF-SE-S003-V002",

```

```
"metadata": {
 "name": "MultiFunc-SideEffect-TypeErrorHandlering",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "type_error_handling",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
},
"task": {
 "description": "Analyze type error handling side effects. What is the final type_errors?",
 "code": "type_errors = 0\nvalue_errors = 0\n\ndef safe_convert_to_int(value):\n global type_errors, value_errors\n try:\n result = int(value)\n if result < 0:\n raise ValueError(\"Negative not allowed\")\n return result\n except TypeError:\n type_errors += 1\n return 0\n except ValueError:\n value_errors += 1\n return 0\n\n def process_conversions():\n safe_convert_to_int(\"10\")\n # Success\n safe_convert_to_int(None)\n # TypeError\n safe_convert_to_int(\"-5\")\n # ValueError\n safe_convert_to_int([1, 2])\n # TypeError\n safe_convert_to_int(\"5\")\n # Success\n return type_errors\n\n # What value is returned by process_conversions()?",
 "answer": "2"
},
{
 "id": "MF-SE-S003-V003",
 "metadata": {
 "name": "MultiFunc-SideEffect-NestedExceptionHandling",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "nested_exceptions",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "indirect"
},
"task": {
 "description": "Track nested exception handling side effects. What is the final inner_exceptions?"
},
```

```

 "code": "inner_exceptions = 0\nouter_exceptions = 0\n\ndef
inner_function(value):\n global inner_exceptions\n try:\n if value == 0:\n raise ValueError(\"Zero not allowed\")\n return 100 / value\n except
ValueError:\n inner_exceptions += 1\n raise # Re-raise the exception\n\ndef
outer_function(value):\n global outer_exceptions\n try:\n return
inner_function(value)\n except ValueError:\n outer_exceptions += 1\n return
-1\n\ndef process_nested_calls():\n outer_function(10) # Success\n outer_function(0)
Inner and outer exception\n outer_function(5) # Success\n outer_function(0)
Inner and outer exception\n return inner_exceptions\n\n# What value is returned by
process_nested_calls()?",

 "answer": "2"
 }
},
{
 "id": "MF-SE-S003-V004",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionRecovery",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "exception_recovery",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze exception recovery side effects. What is the final
recovery_count?",

 "code": "recovery_count = 0\nfailed_operations = 0\n\ndef
risky_operation(data):\n global recovery_count, failed_operations\n try:\n if
len(data) == 0:\n raise ValueError(\"Empty data\")\n return len(data) *
2\n except ValueError:\n failed_operations += 1\n recovery_count += 1\n
 return 1 # Default recovery value\n\ndef process_with_recovery():\n data_sets =
[\"abc\", \"\", \"de\", \"\", \"fgh\"]\n results = []\n for data in data_sets:\n
 result = risky_operation(data)\n results.append(result)\n return
recovery_count\n\n# What value is returned by process_with_recovery()?",

 "answer": "2"
 }
},
{
 "id": "MF-SE-S003-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionChaining",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 }
}

```

```

 "intervention": 2,
 "variant_type": "exception_chaining",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track exception chaining side effects. What is the final chain_length?",
 "code": "chain_length = 0\n\ndef level_one(value):\n global chain_length\n\ntry:\n return level_two(value)\nexcept ValueError:\n chain_length += 1\n\n raise RuntimeError(\"Level one failed\") from None\n\ndef level_two(value):\n global chain_length\n\ntry:\n return level_three(value)\nexcept TypeError:\n chain_length += 1\n\n raise ValueError(\"Level two failed\") from None\n\ndef level_three(value):\n global chain_length\n\n if isinstance(value, str):\n chain_length += 1\n\n raise TypeError(\"String not allowed\")\n\n return value *\n\n2\n\ndef test_exception_chain():\n try:\n level_one(\"test\")\n except RuntimeError:\n pass # Catch final exception\n\n return chain_length\n\n# What value is returned by test_exception_chain()?",
 "answer": "3"
 }
},
{
 "id": "MF-SE-S003-V006",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalExceptions",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_exceptions",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional exception side effects. What is the final triggered_exceptions?",
 "code": "triggered_exceptions = 0\n\nprocessed_items = 0\n\n\ndef conditional_process(item, strict_mode):\n global triggered_exceptions, processed_items\n\n try:\n if strict_mode and item < 10:\n raise ValueError(\"Item too small in strict mode\")\n if item < 0:\n raise ValueError(\"Negative item\")\n processed_items += 1\n return item * 2\n except ValueError:\n triggered_exceptions += 1\n\n return 0\n\n\ndef run_processing():\n items = [15, 5, -3, 12, 8]\n\n # First pass: strict_mode = True\n for item in items[:3]:\n conditional_process(item, True)\n\n # Second pass: strict_mode = False\n for item in items[3:]:\n conditional_process(item, False)\n\n return triggered_exceptions\n\n# What value is returned by run_processing()?",
 "answer": "2"
 }
}

```

```
 "id": "MF-SE-S003-V007",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionStatistics",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "exception_statistics",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track exception statistics side effects. What is the final\nal_exceptions?",
 "code": "type_error_count = 0\nvalue_error_count = 0\nindex_error_count =\n\n\n\ndef risky_list_operation(data, index):\n global type_error_count, value_error_count,\n index_error_count\n try:\n if not isinstance(data, list):\n raise\n eError(\"Not a list\")\n if index < 0:\n raise ValueError(\"Negative\n index\")\n return data[index]\n except TypeError:\n type_error_count += 1\n return None\n except ValueError:\n value_error_count += 1\n return\n except IndexError:\n index_error_count += 1\n return None\n\n\ndef\nt_operations():\n risky_list_operation([1, 2, 3], 1) # Success\n risky_list_operation(\"not_list\", 0) # TypeError\n risky_list_operation([1, 2], -1) # ValueError\n risky_list_operation([1], 5) # IndexError\n risky_list_operation({}, 0) # TypeError\n\n return type_error_count +\n value_error_count + index_error_count\n\n\n# What value is returned by test_operations()?",
 "answer": "4"
 }
 },
 {
 "id": "MF-SE-S003-V008",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionPropagation",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "exception_propagation",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track exception propagation side effects. What is the final\nal_exceptions?",
 "code": "type_error_count = 0\nvalue_error_count = 0\nindex_error_count =\n\n\n\ndef risky_list_operation(data, index):\n global type_error_count, value_error_count,\n index_error_count\n try:\n if not isinstance(data, list):\n raise\n eError(\"Not a list\")\n if index < 0:\n raise ValueError(\"Negative\n index\")\n return data[index]\n except TypeError:\n type_error_count += 1\n return None\n except ValueError:\n value_error_count += 1\n return\n except IndexError:\n index_error_count += 1\n return None\n\n\ndef\nt_operations():\n risky_list_operation([1, 2, 3], 1) # Success\n risky_list_operation(\"not_list\", 0) # TypeError\n risky_list_operation([1, 2], -1) # ValueError\n risky_list_operation([1], 5) # IndexError\n risky_list_operation({}, 0) # TypeError\n\n return type_error_count +\n value_error_count + index_error_count\n\n\n# What value is returned by test_operations()?",
 "answer": "4"
 }
 }
]
```

```

 "description": "Analyze exception propagation side effects. What is the final propagation_depth?",

 "code": "propagation_depth = 0\nexception_caught = False\n\ndef deep_function_4():\n global propagation_depth\n propagation_depth += 1\n raise ValueError(\"Deep error\")\n\ndef deep_function_3():\n global propagation_depth\n propagation_depth += 1\n return deep_function_4()\n\ndef deep_function_2():\n global propagation_depth\n propagation_depth += 1\n return deep_function_3()\n\ndef deep_function_1():\n global propagation_depth, exception_caught\n propagation_depth += 1\n try:\n return deep_function_2()\n except ValueError:\n exception_caught = True\n return 0\n\ndef test_propagation():\n deep_function_1()\n return propagation_depth\n\n# What value is returned by test_propagation()?",

 "answer": "4"
 },
 {
 "id": "MF-SE-S003-V009",
 "metadata": {
 "name": "MultiFunc-SideEffect-ResourceCleanupExceptions",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "resource_cleanup",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Track resource cleanup exception side effects. What is the final cleanup_calls?",

 "code": "cleanup_calls = 0\nresource_errors = 0\n\ndef acquire_resource(resource_id):\n if resource_id == 99:\n raise RuntimeError(\"Resource unavailable\")\n return f\"resource_{resource_id}\"\n\ndef cleanup_resource():\n global cleanup_calls\n cleanup_calls += 1\n\ndef use_resource_safely(resource_id):\n global resource_errors\n try:\n resource = acquire_resource(resource_id)\n return f\"Used {resource}\"\n except RuntimeError:\n resource_errors += 1\n return \"Failed\"\n\nfinally:\n cleanup_resource()\n\ndef test_resource_usage():\n use_resource_safely(1) # Success,\n cleanup\n use_resource_safely(99) # Error, cleanup\n use_resource_safely(2) # Success, cleanup\n use_resource_safely(99) # Error, cleanup\n return cleanup_calls\n\n# What value is returned by test_resource_usage()?",

 "answer": "4"
 },
 {
 "id": "MF-SE-S003-V010",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionStateRecovery",
 "category": "MultiFunc-Level",
 }
 }
 }
]
```

```
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "state_recovery",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Analyze exception state recovery side effects. What is the final recovery_operations?",
 "code": "system_state = 100\nrecovery_operations = 0\nfailed_operations = 0\n\ndef risky_state_change(delta):\n global system_state, failed_operations,\n recovery_operations\n original_state = system_state\n try:\n system_state += delta\n if system_state < 0:\n raise ValueError(\"Invalid state\")\n if system_state > 200:\n raise OverflowError(\"State overflow\")\n return system_state\n except (ValueError, OverflowError):\n failed_operations += 1\n system_state = original_state # Rollback\n recovery_operations += 1\n\nreturn system_state\n\ndef test_state_changes():\n risky_state_change(50) # 150, success\n risky_state_change(-200) # would be -50, error, rollback to 150\n risky_state_change(60) # would be 210, error, rollback to 150\n risky_state_change(10) # 160, success\n return recovery_operations\n\n# What value is returned by test_state_changes()?",\n "answer": "2"
 }
},
{
 "id": "MF-SE-S004-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-FinallyBlockExecution",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "finally_block",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track finally block execution side effects. What is the final finally_count?",
```

```

 "code": "finally_count = 0\nsuccess_count = 0\n\noperation_with_finally(should_fail):\n global finally_count, success_count\n try:\n if should_fail:\n raise ValueError(\"Forced failure\")\n success_count += 1\n return \"success\"\n except ValueError:\n return \"failed\"\n finally:\n finally_count += 1\n undef test_finally_blocks()\noperation_with_finally(False) # success, finally runs\noperation_with_finally(True) # Failure, finally runs\noperation_with_finally(False) # success, finally runs\nreturn finally_count\n\n# what value is returned by test_finally_blocks()?",\n "answer": "3\"\n }\n},\n{\n "id": "MF-SE-S004-V002",\n "metadata": {\n "name": "MultiFunc-SideEffect-NestedFinallyBlocks",\n "category": "MultiFunc-Level",\n "subcategory": "Side Effect",\n "type": "variant",\n "source": "Generated",\n "language": "python",\n "difficulty": "medium",\n "intervention": 1,\n "variant_type": "nested_finally",\n "side_effect_type": "exception",\n "effect_scope": "global",\n "observability": "indirect"\n },\n "task": {\n "description": "Analyze nested finally block side effects. What is the final outer_finally_count?",\n "code": "outer_finally_count = 0\ninner_finally_count = 0\n\ninner_operation(value):\n global inner_finally_count\n try:\n if value < 0:\n raise ValueError(\"Negative value\")\n return value * 2\n finally:\n inner_finally_count += 1\n undef outer_operation(value)\n global outer_finally_count\n try:\n return inner_operation(value)\n except ValueError:\n return 0\nfinally:\n outer_finally_count += 1\n undef test_nested_finally()\n\nouter_operation(5) # Success, both finally blocks run\nouter_operation(-3) # Error, both finally blocks run\nouter_operation(10) # Success, both finally blocks run\nreturn outer_finally_count\n\n# what value is returned by test_nested_finally()?",\n "answer": "3\"\n }\n},\n{\n "id": "MF-SE-S004-V003",\n "metadata": {\n "name": "MultiFunc-SideEffect-MultipleResourceCleanup",\n "category": "MultiFunc-Level",\n "subcategory": "Side Effect",\n "type": "variant",\n "source": "Generated",\n "language": "python",\n "difficulty": "hard",\n "intervention": 2,\n }\n}
```

```
 "variant_type": "multiple_resources",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track multiple resource cleanup side effects. What is the final total_cleanups?",
 "code": "resource_a_cleanups = 0\nresource_b_cleanups = 0\nresource_c_cleanups = 0\n\nndef cleanup_resource_a():\n global resource_a_cleanups\n resource_a_cleanups += 1\n\nndef cleanup_resource_b():\n global resource_b_cleanups\n resource_b_cleanups += 1\n\nndef cleanup_resource_c():\n global resource_c_cleanups\n resource_c_cleanups += 1\n\nndef complex_operation(fail_at_step):\n try:\n # Step 1\n if fail_at_step == 1:\n raise ValueError(\"Step 1 failed\")\n # Step 2\n if fail_at_step == 2:\n raise ValueError(\"Step 2 failed\")\n # Step 3\n if fail_at_step == 3:\n raise ValueError(\"Step 3 failed\")\n\n return \"success\"\n except ValueError:\n return \"failed\"\n finally:\n cleanup_resource_a()\n cleanup_resource_b()\n cleanup_resource_c()\n\nndef test_multiple_resources():\n complex_operation(0) # success, all cleanups\n\ncomplex_operation(2) # Fail at step 2, all cleanups\n\n return resource_a_cleanups +\nresource_b_cleanups + resource_c_cleanups\n\n# what value is returned by\ntest_multiple_resources()?",
 "answer": "6"
 }
},
{
 "id": "MF-SE-S004-v004",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalCleanup",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_cleanup",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional cleanup side effects. What is the final cleanup_operations?",
 }
}
```

```

 "code": "cleanup_operations = 0\nresources_acquired = 0\n\ndef
acquire_resource(resource_type):\n global resources_acquired\n if resource_type ==
\"invalid\":\n return None\n resources_acquired += 1\n return
f\"resource_{resources_acquired}\"\\n\\n\\ndef cleanup_if_needed(resource):\n global
cleanup_operations\\n if resource is not None:\\n cleanup_operations += 1\\n\\n\\ndef
safe_resource_operation(resource_type):\n resource = None\\n try:\\n resource =
acquire_resource(resource_type)\\n if resource is None:\\n raise
ValueError(\"Failed to acquire resource\")\\n return f\"Used {resource}\"\\n except
ValueError:\\n return \"Operation failed\"\\n finally:\\n cleanup_if_needed(resource)\\n\\n\\ndef test_conditional_cleanup():
safe_resource_operation(\"valid\") # Acquire + cleanup\\n
safe_resource_operation(\"invalid\") # No acquire, no cleanup\\n
safe_resource_operation(\"valid\") # Acquire + cleanup\\n return
cleanup_operations\\n\\n# what value is returned by test_conditional_cleanup()?",

 "answer": "2"
 }
},
{
 "id": "MF-SE-S004-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-ExceptionInFinally",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "exception_in_finally",
 "side_effect_type": "exception",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track exception in finally block side effects. What is the final
finally_exceptions?",

 "code": "finally_exceptions = 0\\nmain_exceptions = 0\\nfinally_calls = 0\\n\\n\\ndef
risky_cleanup(should_fail):\n global finally_exceptions, finally_calls\\n finally_calls
+= 1\\n if should_fail:\\n finally_exceptions += 1\\n raise
RuntimeError(\"Cleanup failed\")\\n\\n\\ndef operation_with_risky_cleanup(main_fails,
cleanup_fails):\n global main_exceptions\\n try:\\n if main_fails:\\n raise
ValueError(\"Main operation failed\")\\n return \"success\"\\n except
ValueError:\\n main_exceptions += 1\\n return \"main_failed\"\\n finally:\\n
 try:\\n risky_cleanup(cleanup_fails)\\n except RuntimeError:\\n
 pass # Suppress cleanup exception\\n\\n\\ndef test_risky_cleanup():
operation_with_risky_cleanup(False, False) # Success, cleanup ok\\n
operation_with_risky_cleanup(True, False) # Main fails, cleanup ok\\n
operation_with_risky_cleanup(False, True) # Success, cleanup fails\\n return
finally_exceptions\\n\\n# what value is returned by test_risky_cleanup()?",

 "answer": "1"
 }
}

```

```

"id": "MF-SE-S004-V006",
"metadata": {
 "name": "MultiFunc-SideEffect-ResourceStateTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "resource_state",
 "side_effect_type": "state",
 "effect_scope": "global",
 "observability": "direct"
},
"task": {
 "description": "Track resource state changes side effects. What is the final active_resources?",
 "code": "active_resources = 0\nmax_resources_used = 0\n\ndef allocate_resource():\n global active_resources, max_resources_used\n active_resources += 1\n max_resources_used = max(max_resources_used, active_resources)\n return f\"resource_{active_resources}\".\n\ndef deallocate_resource():\n global active_resources\n if active_resources > 0:\n active_resources -= 1\n\ndef resource_operation(should_fail):\n resource = None\n try:\n resource = allocate_resource()\n if should_fail:\n raise RuntimeError(\"Operation failed\")\n return \"success\"\n except RuntimeError:\n return \"failed\"\n finally:\n if resource:\n deallocate_resource()\n\n\ndef test_resource_tracking():\n resource_operation(False) # success\n resource_operation(True) # Failure\n resource_operation(False) # success\n return active_resources\n\n# What value is returned by test_resource_tracking()?",
 "answer": "0"
},
{
 "id": "MF-SE-S004-V007",
 "metadata": {
 "name": "MultiFunc-SideEffect-BatchCleanupOperations",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "batch_cleanup",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "delayed"
},
 "task": {
 "description": "Analyze batch cleanup operation side effects. What is the final batch_cleanups?",
 }
}

```

```

 "code": "pending_cleanups = []\nbatch_cleanups = 0\n\ndef
add_cleanup_task(task_id):\n global pending_cleanups\n pending_cleanups.append(task_id)\n\n global batch_cleanups,
pending_cleanups\n if pending_cleanups:\n batch_cleanups += 1\n\nbatch_cleanups = []\n\ndef batch_operation(operations):\n try:\n for i, op in
enumerate(operations):\n add_cleanup_task(f\"task_{i}\")\n\n if op ==
\"fail\":\n raise ValueError(f\"Operation {i} failed\")\n\n return
\"success\"\n except ValueError:\n return \"failed\"\n finally:\n execute_batch_cleanup()\n\n global batch_operation([\"ok\",
\"ok\"])\n # Success, cleanup\n batch_operation([\"ok\", \"fail\", \"ok\"])\n # Failure, cleanup\n return batch_cleanups\n\n# What value is returned by
test_batch_operations()?",\n "answer": "2"
 }
},
{
 "id": "MF-SE-S004-V008",
 "metadata": {
 "name": "MultiFunc-SideEffect-HierarchicalCleanup",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "hierarchical_cleanup",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track hierarchical cleanup side effects. What is the final
level_cleanups?",\n "code": "level_1_cleanups = 0\nlevel_2_cleanups = 0\nlevel_3_cleanups = 0\n\ndef
cleanup_level_3():\n global level_3_cleanups\n level_3_cleanups += 1\n\ndef
cleanup_level_2():\n global level_2_cleanups\n level_2_cleanups += 1\n\ndef
cleanup_level_1():\n global level_1_cleanups\n level_1_cleanups += 1\n\n cleanup_level_2()\n\n global hierarchical_operation(fail_at_level):\n try:\n if fail_at_level == 1:\n raise ValueError(\"Level 1 failure\")\n\n if fail_at_level == 2:\n raise ValueError(\"Level 2 failure\")\n\n if fail_at_level == 3:\n raise ValueError(\"Level 3 failure\")\n\n return\n\n \"success\"\n except ValueError:\n return \"failed\"\n finally:\n cleanup_level_1()\n\n global hierarchical_operation(0)\n # Success, all cleanups\n hierarchical_operation(2)\n # Failure, all cleanups\n return
level_1_cleanups + level_2_cleanups + level_3_cleanups\n\n# What value is returned by
test_hierarchical_cleanup()?",\n "answer": "6"
 }
},
{
 "id": "MF-SE-S004-V009",
 "metadata": {

```

```
 "name": "MultiFunc-SideEffect-TimedCleanupOperations",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "timed_cleanup",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Analyze timed cleanup operation side effects. What is the final cleanup_time?",
 "code": "cleanup_time = 0\noperations_completed = 0\n\ndef simulate_cleanup_time(duration):\n global cleanup_time\n cleanup_time +=\nduration\n\ndef timed_operation(duration, should_fail):\n global operations_completed\n start_time = cleanup_time\n try:\n # Simulate operation time\n simulate_cleanup_time(duration)\n if should_fail:\n raise\nRuntimeError(\"Operation failed\")\n operations_completed += 1\n return\n\"success\"\n except RuntimeError:\n return \"failed\"\n finally:\n # Cleanup always takes 5 time units\n simulate_cleanup_time(5)\n\ndef test_timed_operations():\n timed_operation(10, False) # 10 + 5 = 15\n timed_operation(8, True) # 8 + 5 = 13\n timed_operation(12, False) # 12 + 5 = 17\n\nreturn cleanup_time\n\n# What value is returned by test_timed_operations()?",
 "answer": "45"
 }
},
{
 "id": "MF-SE-S004-V010",
 "metadata": {
 "name": "MultiFunc-SideEffect-CleanupSequenceTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "python",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "cleanup_sequence",
 "side_effect_type": "resource",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track cleanup sequence execution side effects. What is the final sequence_number?",
 }
}
```

```

 "code": "sequence_number = 0\ncleanup_sequence = []\n\nndef
record_cleanup(cleanup_type):\n global sequence_number, cleanup_sequence\nsequence_number += 1\n cleanup_sequence.append((sequence_number, cleanup_type))\n\nndef
multi_resource_operation(resources, fail_after):\n acquired_resources = []\n try:\n for i, resource in enumerate(resources):\n if i == fail_after:\n raise ValueError(f\"Failed at resource {i}\")\n acquired_resources.append(resource)\n return \"success\"\n except ValueError:\n return \"failed\"\n finally:\n # Cleanup in reverse order\n for\nresource in reversed(acquired_resources):\nrecord_cleanup(f\"cleanup_{resource}\")\n if acquired_resources:\nrecord_cleanup(\"final_cleanup\")\n\nndef test_cleanup_sequence():\nmulti_resource_operation([\"A\", \"B\", \"C\"], 999) # Success, cleanup A,B,C + final\nmulti_resource_operation([\"D\", \"E\"], 1) # Fail at E, cleanup D + final\nreturn sequence_number\n\n# what value is returned by test_cleanup_sequence()?",\n \"answer\": \"6\"\n }\n},\n{\n \"id\": \"MF-SE-S005-V001\",\n \"metadata\": {\n \"name\": \"MultiFunc-SideEffect-SimpleMemoryTracking\",\n \"category\": \"MultiFunc-Level\",\n \"subcategory\": \"Side Effect\",\n \"type\": \"variant\",\n \"source\": \"Generated\",\n \"language\": \"c\",\n \"difficulty\": \"easy\",\n \"intervention\": 0,\n \"variant_type\": \"simple_tracking\",\n \"side_effect_type\": \"memory\",\n \"effect_scope\": \"global\",\n \"observability\": \"direct\"\n },\n \"task\": {\n \"description\": \"Track simple memory allocation side effects. What is the final allocation_count?\",\n \"code\": \"int allocation_count = 0;\n\nvoid* simple_alloc(size_t size) {\n allocation_count++;\n return malloc(size);\n}\n\nvoid simple_free(void* ptr) {\n if (ptr != NULL) {\n allocation_count--;\n free(ptr);\n }\n}\n\nint simple_memory_test() {\n void* ptr1 = simple_alloc(50);\n void* ptr2 = simple_alloc(100);\n simple_free(ptr1);\n void* ptr3 = simple_alloc(75);\n return allocation_count;\n}\n\n// what value is returned by simple_memory_test()?\",\n \"answer\": \"2\"\n }\n},\n{\n \"id\": \"MF-SE-S005-V002\",\n \"metadata\": {\n \"name\": \"MultiFunc-SideEffect-MemoryPoolManagement\",\n \"category\": \"MultiFunc-Level\",\n \"subcategory\": \"Side Effect\",\n \"type\": \"variant\",\n \"source\": \"Generated\",
```

```
"language": "c",
"difficulty": "medium",
"intervention": 1,
"variant_type": "pool_management",
"side_effect_type": "memory",
"effect_scope": "global",
"observability": "direct"
},
"task": {
 "description": "Analyze memory pool management side effects. What is the final pool_size?",
 "code": "int pool_size = 1000;\nint allocated_from_pool = 0;\n\nint\nallocate_from_pool(int size) {\n if (pool_size >= size) {\n pool_size -= size;\n allocated_from_pool += size;\n return 1; // Success\n }\n return 0; //\nFailure\n}\n\nint deallocate_to_pool(int size) {\n pool_size += size;\n allocated_from_pool -= size;\n return pool_size;\n}\n\nint test_pool_operations() {\n allocate_from_pool(300); // pool_size = 700\n allocate_from_pool(200); // pool_size = 500\n deallocate_to_pool(100); // pool_size = 600\n allocate_from_pool(250); //\n pool_size = 350\n return pool_size;\n}\n\n// What value is returned by test_pool_operations()?",
 "answer": "350"
},
{
 "id": "MF-SE-S005-V003",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryLeakDetection",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "leak_detection",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track memory leak detection side effects. What is the final leaked_blocks?",
 "code": "int total_allocations = 0;\nint total_deallocations = 0;\nint\nleaked_blocks = 0;\n\nvoid* tracked_malloc(size_t size) {\n total_allocations++;\n return malloc(size);\n}\n\nvoid tracked_free(void* ptr) {\n if (ptr != NULL) {\n total_deallocations++;\n free(ptr);\n }\n}\n\nint check_leaks() {\n leaked_blocks = total_allocations - total_deallocations;\n return\n leaked_blocks;\n}\n\nint memory_leak_test() {\n void* ptr1 = tracked_malloc(100);\n void* ptr2 = tracked_malloc(200);\n void* ptr3 = tracked_malloc(150);\n tracked_free(ptr1);\n tracked_free(ptr3); // ptr2 is leaked\n void* ptr4 =\n tracked_malloc(300);\n // ptr4 is also leaked\n return check_leaks();\n}\n\n// What value is returned by memory_leak_test()?",
 "answer": "2"
 }
}
```

```

 }
 },
 {
 "id": "MF-SE-S005-V004",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalAllocation",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_allocation",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional allocation side effects. What is the final successful_allocations?",

 "code": "int successful_allocations = 0;\nint failed_allocations = 0;\nint

memory_limit = 500;\nint memory_used = 0;\n\nvoid* conditional_alloc(size_t size) {\n if

(memory_used + size <= memory_limit) {\n memory_used += size;\n successful_allocations++;\n return malloc(size);\n } else {\n failed_allocations++;\n return NULL;\n }\n}\n\nvoid conditional_free(void* ptr,

size_t size) {\n if (ptr != NULL) {\n memory_used -= size;\n free(ptr);\n }\n}\n\nint test_conditional_allocation() {\n void* ptr1 = conditional_alloc(200); // Success\n void* ptr2 = conditional_alloc(150); // Success\n void* ptr3 = conditional_alloc(200); // Fail (200+150+200 > 500)\n conditional_free(ptr1, 200);\n void* ptr4 = conditional_alloc(180); // Success\n \n return successful_allocations;\n}\n\n// What value is returned by test_conditional_allocation()?",

 "answer": "3"
 }
 },
 {
 "id": "MF-SE-S005-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryFragmentation",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "fragmentation",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {

```

```

 "description": "Track memory fragmentation side effects. What is the final fragmentation_score?",
 "code": "int fragmentation_score = 0;\nint allocation_sequence = 0;\n\nvoid* fragment_alloc(size_t size) {\n allocation_sequence++; // Simulate fragmentation: odd allocations increase fragmentation\n if (allocation_sequence % 2 == 1) {\n fragmentation_score += size / 10;\n }\n return malloc(size);\n}\n\nvoid fragment_free(void* ptr, size_t size) {\n if (ptr != NULL) {\n // Deallocation reduces fragmentation slightly\n fragmentation_score -= size / 20;\n }\n}\n\nint test_fragmentation() {\n void* ptr1 = fragment_alloc(100); // +10 fragmentation\n void* ptr2 = fragment_alloc(80); // +0 fragmentation (even)\n void* ptr3 = fragment_alloc(120); // +12 fragmentation\n fragment_free(ptr1, 100); // -5 fragmentation\n void* ptr4 = fragment_alloc(60); // +0 fragmentation (even)\n return fragmentation_score;\n}\n\n// What value is returned by test_fragmentation()?",

 "answer": "17"
 },
 {
 "id": "MF-SE-S005-V006",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryAlignmentTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "alignment_tracking",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze memory alignment tracking side effects. What is the final aligned_allocations?",
 "code": "int aligned_allocations = 0;\nint unaligned_allocations = 0;\n\nvoid* aligned_alloc(size_t size, size_t alignment) {\n if (size % alignment == 0) {\n aligned_allocations++;\n } else {\n unaligned_allocations++;\n }\n return malloc(size);\n}\n\nint test_alignment() {\n aligned_alloc(64, 8); // 64 % 8 == 0, aligned\n aligned_alloc(100, 8); // 100 % 8 == 4, unaligned\n aligned_alloc(96, 16); // 96 % 16 == 0, aligned\n aligned_alloc(50, 4); // 50 % 4 == 2, unaligned\n aligned_alloc(128, 32); // 128 % 32 == 0, aligned\n return aligned_allocations;\n}\n\n// What value is returned by test_alignment()?",

 "answer": "3"
 },
 },
 {
 "id": "MF-SE-S005-V007",
 "metadata": {
 "name": "MultiFunc-SideEffect-BatchMemoryOperations",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 }
 }
]
```

```
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "batch_operations",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Track batch memory operation side effects. What is the final batch_operations?",
 "code": "int batch_operations = 0;\nint pending_allocations = 0;\nconst int BATCH_SIZE = 3;\n\nvoid add_to_allocation_batch(size_t size) {\n pending_allocations++;\n\n if (pending_allocations >= BATCH_SIZE) {\n process_allocation_batch();\n }\n}\n\nvoid process_allocation_batch() {\n batch_operations++;\n pending_allocations = 0;\n}\n\nint test_batch_allocations() {\n add_to_allocation_batch(100); // pending = 1\n add_to_allocation_batch(200); // pending = 2\n add_to_allocation_batch(150); // pending = 3, triggers batch\n add_to_allocation_batch(300); // pending = 1\n add_to_allocation_batch(250); // pending = 2\n add_to_allocation_batch(180); // pending = 3, triggers batch\n add_to_allocation_batch(120); // pending = 1\n\n return batch_operations;\n}\n\n// what value is returned by test_batch_allocations()?",
 "answer": "2"
 }
},
{
 "id": "MF-SE-S005-V008",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryPressureTracking",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "pressure_tracking",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Analyze memory pressure tracking side effects. What is the final pressure_level?"
 }
}
```

```

 "code": "int pressure_level = 0;\nint total_memory = 1000;\nint used_memory = 0;\n\nvoid update_pressure_level() {\n int usage_percent = (used_memory * 100) / total_memory;\n if (usage_percent >= 80) {\n pressure_level = 3; // High\n } else if (usage_percent >= 50) {\n pressure_level = 2; // Medium\n } else if (usage_percent >= 20) {\n pressure_level = 1; // Low\n } else {\n pressure_level = 0; // None\n }\n}\n\nvoid* pressure_aware_alloc(size_t size) {\n used_memory += size;\n update_pressure_level();\n return malloc(size);\n}\n\nvoid pressure_aware_free(void* ptr, size_t size) {\n if (ptr != NULL) {\n used_memory -= size;\n update_pressure_level();\n free(ptr);\n }\n}\n\nint test_memory_pressure() {\n void* ptr1 = pressure_aware_alloc(300); // 30%, level = 1\n void* ptr2 = pressure_aware_alloc(400); // 70%, level = 2\n void* ptr3 = pressure_aware_alloc(200); // 90%, level = 3\n pressure_aware_free(ptr1, 300);\n // 60%, level = 2\n return pressure_level;\n}\n\n// What value is returned by test_memory_pressure()?",\n "answer": "2\"\n }\n},\n{\n "id": "MF-SE-S005-V009",\n "metadata": {\n "name": "MultiFunc-SideEffect-MemoryStatisticsTracking",\n "category": "MultiFunc-Level",\n "subcategory": "Side Effect",\n "type": "variant",\n "source": "Generated",\n "language": "c",\n "difficulty": "expert",\n "intervention": 3,\n "variant_type": "statistics_tracking",\n "side_effect_type": "memory",\n "effect_scope": "global",\n "observability": "delayed"\n },\n "task": {\n "description": "Track memory statistics side effects. What is the final peak_memory_used?",\n "code": "int current_memory_used = 0;\nint peak_memory_used = 0;\nint total_allocations = 0;\nint average_allocation_size = 0;\n\nvoid update_memory_stats(size_t size, int is_allocation) {\n if (is_allocation) {\n current_memory_used += size;\n total_allocations++;\n if (current_memory_used > peak_memory_used) {\n peak_memory_used = current_memory_used;\n }\n average_allocation_size = peak_memory_used / total_allocations;\n } else {\n current_memory_used -= size;\n }\n}\n\nvoid* stats_alloc(size_t size) {\n update_memory_stats(size, 1);\n return malloc(size);\n}\n\nvoid stats_free(void* ptr, size_t size) {\n if (ptr != NULL) {\n update_memory_stats(size, 0);\n free(ptr);\n }\n}\n\nint test_memory_stats() {\n void* ptr1 = stats_alloc(150); // current=150, peak=150\n void* ptr2 = stats_alloc(200); // current=350, peak=350\n void* ptr3 = stats_alloc(100); // current=450, peak=450\n stats_free(ptr2, 200); // current=250, peak=450\n void* ptr4 = stats_alloc(300); // current=550, peak=550\n return peak_memory_used;\n}\n\n// What value is returned by test_memory_stats()?",\n "answer": "550\"\n }\n},\n{

```

```
{
 "id": "MF-SE-S005-V010",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryGarbageCollection",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "garbage_collection",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Analyze garbage collection side effects. What is the final gc_cycles?",
 "code": "int gc_cycles = 0;\nint allocated_objects = 0;\nint gc_threshold = 5;\nint objects_collected = 0;\n\nvoid trigger_gc_if_needed() {\n if (allocated_objects >= gc_threshold) {\n gc_cycles++;\n objects_collected += allocated_objects / 2;\n }\n}\n\nvoid* gc_alloc(size_t size) {\n allocated_objects++;\n trigger_gc_if_needed();\n return malloc(size);\n}\n\nvoid gc_free(void* ptr) {\n if (ptr != NULL) {\n allocated_objects--;\n free(ptr);\n }\n}\n\nint test_garbage_collection() {\n gc_alloc(100); // objects=1\n gc_alloc(200); // objects=2\n gc_alloc(150); // objects=3\n gc_alloc(300); // objects=4\n gc_alloc(250); // objects=5, triggers GC, objects=2, gc_cycles=1\n gc_alloc(180); // objects=3\n gc_alloc(220); // objects=4\n gc_alloc(160); // objects=5, triggers GC, objects=2, gc_cycles=2\n \n return gc_cycles;\n}\n\n// What value is returned by test_garbage_collection()?",
 "answer": "2"
 }
},
{
 "id": "MF-SE-S006-V001",
 "metadata": {
 "name": "MultiFunc-SideEffect-SimpleArrayModification",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "easy",
 "intervention": 0,
 "variant_type": "simple_array",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze simple array modification side effects. What is the final value at data[1]?",
 "code": "int data[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};\n\nvoid modify_array() {\n data[1] = 10;\n}\n\nint main() {\n modify_array();\n \n return data[1];\n}"
 }
}
```

```

 "code": "int data[4] = {10, 20, 30, 40};\n\nvoid increment_element(int index)\n{\n if (index >= 0 && index < 4) {\n data[index] += 10;\n }\n}\n\nint simple_array_test()\n{\n increment_element(1); // data[1] = 30\n increment_element(1);\n // data[1] = 40\n increment_element(2); // data[2] = 40\n return data[1];\n}\n\n// what value is returned by simple_array_test()?",\n
 "answer": "40"
 },
 {
 "id": "MF-SE-S006-V002",
 "metadata": {
 "name": "MultiFunc-SideEffect-ConditionalModification",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "conditional_modification",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze conditional memory modification side effects. What is the final value at buffer[3]?",\n "code": "int buffer[6] = {5, 10, 15, 20, 25, 30};\n\nvoid conditional_update(int index, int threshold) {\n if (index >= 0 && index < 6 && buffer[index] >= threshold) {\n buffer[index] *= 2;\n }\n}\n\nint test_conditional_updates()\n{\n conditional_update(3, 15); // 20 >= 15, buffer[3] = 40\n conditional_update(1, 15); // 10 < 15, no change\n conditional_update(3, 30); // 40 >= 30, buffer[3] = 80\n conditional_update(2, 10); // 15 >= 10, buffer[2] = 30\n return buffer[3];\n}\n\n// what value is returned by test_conditional_updates()?",\n "answer": "80"
 }
 },
 {
 "id": "MF-SE-S006-V003",
 "metadata": {
 "name": "MultiFunc-SideEffect-PointerBasedModification",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "pointer_modification",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Analyze pointer-based memory modification side effects. What is the final value at buffer[3]?",\n "code": "int* data = {10, 20, 30, 40};\n\nvoid increment_element(int index)\n{\n if (index >= 0 && index < 4) {\n data[index] += 10;\n }\n}\n\nint simple_array_test()\n{\n increment_element(1); // data[1] = 30\n increment_element(1);\n // data[1] = 40\n increment_element(2); // data[2] = 40\n return data[1];\n}\n\n// what value is returned by simple_array_test()?",\n "answer": "40"
 }
 }
]
```

```

"task": {
 "description": "Track pointer-based modification side effects. What is the final value at memory[4]?",
 "code": "int memory[8] = {1, 2, 3, 4, 5, 6, 7, 8};\n\nvoid pointer_modify(int* ptr, int offset, int value) {\n if (ptr != NULL) {\n *(ptr + offset) = value;\n }\n}\n\nvoid indirect_modify(int base_index, int offset, int value) {\n if (base_index >= 0 && base_index < 8) {\n pointer_modify(&memory[base_index], offset, value);\n }\n}\n\nint test_pointer_modifications() {\n indirect_modify(2, 2, 99); // memory[2+2] = memory[4] = 99\n indirect_modify(3, 1, 77); // memory[3+1] = memory[4] = 77\n indirect_modify(4, 0, 55); // memory[4+0] = memory[4] = 55\n return memory[4];\n}\n\n// what value is returned by test_pointer_modifications()?",
 "answer": "55"
},
{
 "id": "MF-SE-S006-V004",
 "metadata": {
 "name": "MultiFunc-SideEffect-BatchMemoryUpdates",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "batch_updates",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze batch memory update side effects. What is the final value at values[2]?",
 "code": "int values[5] = {100, 200, 300, 400, 500};\n\nvoid batch_add(int start_index, int count, int addend) {\n for (int i = 0; i < count; i++) {\n int index = start_index + i;\n if (index >= 0 && index < 5) {\n values[index] += addend;\n }\n }\n}\n\nint test_batch_operations() {\n batch_add(1, 3, 50);\n // values[1,2,3] += 50\n batch_add(2, 2, 25); // values[2,3] += 25\n batch_add(0, 4, 10); // values[0,1,2,3] += 10\n return values[2];\n}\n\n// what value is returned by test_batch_operations()?",
 "answer": "385"
 }
},
{
 "id": "MF-SE-S006-V005",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemorySwapping",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 1
 }
}

```

```

 "intervention": 2,
 "variant_type": "memory_swapping",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Track memory swapping side effects. What is the final value at array[1]?",

 "code": "int array[6] = {11, 22, 33, 44, 55, 66};\nint swap_count = 0;\n\nvoid swap_elements(int index1, int index2) {\n if (index1 >= 0 && index1 < 6 && index2 >= 0 && index2 < 6) {\n int temp = array[index1];\n array[index1] = array[index2];\n array[index2] = temp;\n swap_count++;\n }\n}\n\nvoid rotate_three(int a, int b, int c) {\n // Rotate: a -> b -> c -> a\n int temp = array[a];\n array[a] = array[c];\n array[c] = array[b];\n array[b] = temp;\n swap_count += 2; // Count as 2 swap operations\n}\n\nint test_swapping() {\n swap_elements(1, 3); // array[1]=44, array[3]=22\n rotate_three(0, 1, 2); // 0->1->2->0: array[0]=33, array[1]=11, array[2]=44\n swap_elements(1, 4); // array[1]=55, array[4]=11\n return array[1];\n}\n\n// What value is returned by test_swapping()?",

 "answer": "55"
 }
},
{
 "id": "MF-SE-S006-v006",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryChecksum",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "medium",
 "intervention": 1,
 "variant_type": "checksum_tracking",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Analyze memory checksum tracking side effects. What is the final checksum?",

 "code": "int data_array[4] = {10, 20, 30, 40};\nint checksum = 0;\n\nvoid update_checksum() {\n checksum = 0;\n for (int i = 0; i < 4; i++) {\n checksum += data_array[i];\n }\n}\n\nvoid modify_and_update(int index, int new_value) {\n if (index >= 0 && index < 4) {\n data_array[index] = new_value;\n update_checksum();\n }\n}\n\nint test_checksum_tracking() {\n update_checksum();\n // Initial: 10+20+30+40 = 100\n modify_and_update(1, 25); // 10+25+30+40 = 105\n modify_and_update(3, 50); // 10+25+30+50 = 115\n modify_and_update(0, 15); // 15+25+30+50 = 120\n return checksum;\n}\n\n// What value is returned by test_checksum_tracking()?",

 "answer": "120"
 }
},

```

```
{
 "id": "MF-SE-S006-V007",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryPatternFilling",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "hard",
 "intervention": 2,
 "variant_type": "pattern_filling",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "direct"
 },
 "task": {
 "description": "Track memory pattern filling side effects. What is the final value at grid[1][2]?",
 "code": "int grid[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};\n\nvoid fill_pattern(int start_row, int start_col, int pattern) {\n for (int r = 0; r < 2; r++)\n for (int c = 0; c < 2; c++) {\n int row = start_row + r;\n int col = start_col + c;\n if (row >= 0 && row < 3 && col >= 0 && col < 3) {\n grid[row][col] = pattern + (r * 2 + c);\n }\n }\n}\n\nint test_pattern_filling() {\n fill_pattern(0, 1, 100); // Fill 2x2 area starting at (0,1) with pattern 100+offset\n // grid[0][1]=100, grid[0][2]=101, grid[1][1]=102, grid[1][2]=103\n fill_pattern(1, 0, 200); // Fill 2x2 area starting at (1,0) with pattern 200+offset\n // grid[1][0]=200, grid[1][1]=201, grid[2][0]=202, grid[2][1]=203\n return grid[1][2];\n}\n\n// What value is returned by test_pattern_filling()?",
 "answer": "103"
 }
},
{
 "id": "MF-SE-S006-V008",
 "metadata": {
 "name": "MultiFunc-SideEffect-CircularBufferOperations",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "circular_buffer",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
 },
 "task": {
 "description": "Analyze circular buffer operation side effects. What is the final value at position 1?",
 }
}
```

```

 "code": "int circular_buffer[5] = {0, 0, 0, 0, 0};\nint write_pos = 0;\nint
read_pos = 0;\nint buffer_count = 0;\n\nvoid circular_write(int value) {\n circular_buffer[write_pos] = value;\n write_pos = (write_pos + 1) % 5;\n if (buffer_count < 5) {\n buffer_count++;\n } else {\n read_pos = (read_pos + 1) % 5; // overwrite oldest\n }\n}\n\nint circular_read() {\n if (buffer_count > 0) {\n int value = circular_buffer[read_pos];\n read_pos = (read_pos + 1) % 5;\n buffer_count--;\n return value;\n }\n return -1;\n}\n\nint test_circular_buffer() {\n circular_write(10); // pos 0\n circular_write(20); // pos 1\n circular_write(30); // pos 2\n circular_read(); // Read 10 from pos 0\n circular_write(40); // pos 3\n circular_write(50); // pos 4\n circular_write(60); // pos 0 (overwrites)\n circular_write(70); // pos 1 (overwrites 20)\n return circular_buffer[1];\n}\n\n// what value is returned by test_circular_buffer()?",\n "answer": "70"
 }
},
{
 "id": "MF-SE-S006-V009",
 "metadata": {
 "name": "MultiFunc-SideEffect-MemoryCompression",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "memory_compression",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "delayed"
 },
 "task": {
 "description": "Track memory compression side effects. What is the final compressed_size?",\n "code": "int original_data[8] = {1, 1, 2, 2, 2, 3, 3, 4};\nint compressed_data[8] = {0};\nint compressed_size = 0;\n\nvoid compress_run_length() {\n compressed_size = 0;\n int current_value = original_data[0];\n int count = 1;\n\n for (int i = 1; i < 8; i++) {\n if (original_data[i] == current_value) {\n count++;\n } else {\n compressed_data[compressed_size++] = current_value;\n compressed_data[compressed_size++] = count;\n\n current_value = original_data[i];\n count = 1;\n }\n }\n}\n\n// Handle last run\ncompressed_data[compressed_size++] = current_value;\ncompressed_data[compressed_size++] = count;\n\nvoid modify_and_recompress(int index, int new_value) {\n if (index >= 0 && index < 8) {\n original_data[index] = new_value;\n compress_run_length();\n }\n}\n\nint test_compression() {\n compress_run_length(); // Initial compression\n modify_and_recompress(2, 1); // Change pattern\n modify_and_recompress(7, 3); // Change pattern\n return compressed_size;\n}\n\n// what value is returned by test_compression()?",\n "answer": "8"
 }
},
{
 "id": "MF-SE-S006-V010",

```

```

"metadata": {
 "name": "MultiFunc-SideEffect-MemoryEncryption",
 "category": "MultiFunc-Level",
 "subcategory": "Side Effect",
 "type": "variant",
 "source": "Generated",
 "language": "c",
 "difficulty": "expert",
 "intervention": 3,
 "variant_type": "memory_encryption",
 "side_effect_type": "memory",
 "effect_scope": "global",
 "observability": "indirect"
},
"task": {
 "description": "Analyze memory encryption side effects. What is the final encrypted_checksum?",
 "code": "int secure_data[6] = {100, 200, 300, 400, 500, 600};\nint encryption_key = 42;\nint encrypted_checksum = 0;\nvoid encrypt_memory() {\n for (int i = 0; i < 6; i++) {\n secure_data[i] ^= encryption_key; // XOR encryption\n }\n}\nvoid decrypt_memory() {\n for (int i = 0; i < 6; i++) {\n secure_data[i] ^= encryption_key; // XOR decryption\n }\n}\nvoid calculate_encrypted_checksum() {\n encrypted_checksum = 0;\n for (int i = 0; i < 6; i++) {\n encrypted_checksum += secure_data[i];\n }\n}\nvoid secure_modify(int index, int value) {\n if (index >= 0 && index < 6) {\n decrypt_memory();\n secure_data[index] = value;\n encrypt_memory();\n calculate_encrypted_checksum();\n }\n}\nint test_secure_memory() {\n encrypt_memory();\n calculate_encrypted_checksum();\n secure_modify(1, 250); // Decrypt, modify, encrypt\n secure_modify(3, 450); // Decrypt, modify, encrypt\n return encrypted_checksum;\n}\n// What value is returned by test_secure_memory()?",
 "answer": "1850"
}
}
]

```

## 4E - 大混合 (41)

# 跨函数推理大混合变式生成提示词

## 任务目标

基于提供的函数调用链推理、参数传递推理、状态传播推理、副作用推理种子任务和大混合示例，生成60个左右融合多种跨函数推理类型的综合变式任务，全面测试大模型对函数调用链、参数传递、状态传播、副作用等多重机制的综合理解能力和交叉应用能力。

## 跨函数推理大混合特征分析

跨函数推理大混合关注程序中多个函数间复杂交互的综合效应，重点测试模型对调用链分析与参数传递、状态传播与副作用管理、内存操作与异常处理等复杂组合的深层理解，强调对多维度跨函数机制的综合分析能力。

## 关键要求

- \*\*描述语言\*\*：所有task描述必须使用英文
- \*\*变式数量\*\*：总计生成约60个混合变式任务
- \*\*答案唯一性\*\*：保证答案准确且唯一，避免歧义和多解情况
- \*\*混合复杂度\*\*：每个变式都应融合至少2-3种跨函数推理类型

## 混合变式生成维度

#### ### 1. 调用链+参数传递混合变式

- \*\*链式参数变换变式\*\*：参数在调用链中的逐步变换分析
- \*\*递归参数传递变式\*\*：递归调用中参数的传递和累积
- \*\*回调参数处理变式\*\*：回调函数的参数传递和处理
- \*\*指针链式传递变式\*\*：指针参数在调用链中的传递和修改
- \*\*结构体链式变式\*\*：复杂结构体在调用链中的传递
- \*\*引用链式修改变式\*\*：引用参数在调用链中的修改效果

#### ### 2. 调用链+状态传播混合变式

- \*\*全局状态调用链变式\*\*：全局状态在调用链中的传播和修改
- \*\*静态变量调用链变式\*\*：静态变量在复杂调用链中的状态管理
- \*\*闭包状态调用变式\*\*：闭包状态在嵌套调用中的传播
- \*\*对象状态调用链变式\*\*：对象状态在方法调用链中的演化
- \*\*共享状态竞争变式\*\*：多调用路径对共享状态的竞争访问
- \*\*状态缓存调用变式\*\*：状态缓存在调用链中的使用和更新

#### ### 3. 调用链+副作用混合变式

- \*\*I/O调用链副作用变式\*\*：I/O操作在调用链中的副作用累积
- \*\*内存操作调用链变式\*\*：内存操作在调用链中的副作用传播
- \*\*异常调用链传播变式\*\*：异常在调用链中的传播和处理
- \*\*资源管理调用链变式\*\*：资源管理在调用链中的副作用
- \*\*日志调用链副作用变式\*\*：日志记录在调用链中的副作用
- \*\*时间依赖调用链变式\*\*：时间相关副作用在调用链中的影响

#### ### 4. 参数传递+状态传播混合变式

- \*\*参数状态同步变式\*\*：参数传递与全局状态的同步分析
- \*\*指针状态修改变式\*\*：指针参数对全局状态的修改效果
- \*\*引用状态传播变式\*\*：引用参数的状态传播和持久化
- \*\*结构体状态传递变式\*\*：结构体参数的状态传递和修改
- \*\*数组状态传播变式\*\*：数组参数的状态传播分析
- \*\*闭包参数捕获变式\*\*：闭包参数捕获的状态传播

#### ### 5. 参数传递+副作用混合变式

- \*\*参数副作用传播变式\*\*：参数传递过程中的副作用分析
- \*\*指针副作用变式\*\*：指针参数的内存副作用
- \*\*数组修改副作用变式\*\*：数组参数修改的副作用传播
- \*\*结构体副作用变式\*\*：结构体参数的副作用影响
- \*\*引用副作用传播变式\*\*：引用参数的副作用传播
- \*\*对象参数副作用变式\*\*：对象参数的副作用分析

#### ### 6. 状态传播+副作用混合变式

- \*\*全局状态副作用变式\*\*：全局状态修改的副作用影响
- \*\*静态状态副作用变式\*\*：静态变量状态的副作用传播
- \*\*闭包状态副作用变式\*\*：闭包状态修改的副作用
- \*\*对象状态副作用变式\*\*：对象状态变化的副作用
- \*\*共享状态副作用变式\*\*：共享状态的副作用竞争
- \*\*状态持久化副作用变式\*\*：状态持久化的副作用影响

#### ### 7. 三重混合变式 (调用链+参数+状态)

- \*\*复杂调用状态管理变式\*\*：复杂调用链中的参数和状态管理

- **\*\*递归状态参数变式\*\***: 递归调用中的状态和参数综合分析
- **\*\*回调状态传递变式\*\***: 回调函数的状态和参数传递
- **\*\*链式状态变换变式\*\***: 链式调用中的状态和参数变换
- **\*\*嵌套调用状态变式\*\***: 嵌套调用的复杂状态和参数管理
- **\*\*多路径状态参数变式\*\***: 多路径调用的状态和参数分析

### ### 8. 三重混合变式 (调用链+参数+副作用)

- **\*\*调用链副作用参数变式\*\***: 调用链中参数传递的副作用分析
- **\*\*递归副作用参数变式\*\***: 递归调用的副作用和参数综合
- **\*\*回调副作用处理变式\*\***: 回调函数的副作用和参数处理
- **\*\*链式副作用传播变式\*\***: 链式调用的副作用传播分析
- **\*\*参数修改副作用链变式\*\***: 参数修改在调用链中的副作用
- **\*\*复杂副作用调用变式\*\***: 复杂副作用的调用链分析

### ### 9. 三重混合变式 (调用链+状态+副作用)

- **\*\*调用链状态副作用变式\*\***: 调用链中状态变化的副作用
- **\*\*递归状态副作用变式\*\***: 递归调用的状态和副作用综合
- **\*\*回调状态副作用变式\*\***: 回调函数的状态和副作用管理
- **\*\*链式状态副作用变式\*\***: 链式调用的状态和副作用传播
- **\*\*全局状态调用副作用变式\*\***: 全局状态在调用链副作用中的角色
- **\*\*复杂状态副作用链变式\*\***: 复杂状态和副作用的调用链

### ### 10. 四重混合变式 (调用链+参数+状态+副作用)

- **\*\*全维度跨函数变式\*\***: 四种机制的完全融合分析
- **\*\*复杂系统模拟变式\*\***: 复杂系统中的多机制交互
- **\*\*实时系统分析变式\*\***: 实时系统的多机制综合分析
- **\*\*数据库事务变式\*\***: 数据库事务的多机制分析
- **\*\*网络协议栈变式\*\***: 网络协议栈的跨函数综合分析
- **\*\*编译器优化变式\*\***: 编译器优化的多机制影响

## ## 复杂度层次设计

### ### 简单混合 (Easy) - 12个变式

- 2种机制的基础组合
- 清晰的交互路径和影响
- 直观的分析过程
- 基础的跨函数理解

### ### 中等混合 (Medium) - 18个变式

- 2-3种机制的中度组合
- 包含一定的复杂交互
- 中等复杂度的分析过程
- 需要深入的跨函数理解

### ### 复杂混合 (Hard) - 18个变式

- 3种机制的深度融合
- 复杂的交互和传播路径
- 多层次的综合分析
- 涉及高级概念的理解

### ### 专家级混合 (Expert) - 8个变式

- 3-4种机制的极度复杂组合
- 深度嵌套和复杂交互

- 需要专业级别的综合分析
- 涉及系统级和架构级的理解

#### ### 大师级混合 (Master) - 3个变式

- 四种机制的完全融合
- 极度复杂的综合分析
- 需要最高水平的理解能力
- 挑战性的实际系统模拟

#### ### 宗师级混合 (Grandmaster) - 1个变式

- 超级复杂的实际系统场景
- 极限挑战的综合分析
- 需要专家级的系统理解
- 涉及多个领域的知识综合

### ## 生成策略

#### ### 种子分析策略

1. \*\*识别机制特征\*\*: 分析每类种子的核心机制特征
2. \*\*提取交互模式\*\*: 识别不同机制间的潜在交互模式
3. \*\*确定融合点\*\*: 找出机制融合的自然切入点
4. \*\*保持分析深度\*\*: 在混合中保持每种机制的分析深度

#### ### 变式设计原则

1. \*\*多机制融合\*\*: 每个变式都应体现多种机制的有机结合
2. \*\*英文描述\*\*: 所有task描述必须使用标准英文
3. \*\*答案唯一\*\*: 严格确保答案的准确性和唯一性
4. \*\*渐进复杂\*\*: 从简单组合逐步增加到复杂综合

#### ### 质量保证

1. \*\*机制一致性\*\*: 确保各种机制的语义一致性
2. \*\*交互正确性\*\*: 验证机制间交互的正确性
3. \*\*分析可行性\*\*: 确保综合分析的可行性和合理性
4. \*\*答案验证\*\*: 严格验证答案的准确性和唯一性

### ## 输出格式要求

```
```json
[
  {
    "id": "MF-MIX-V001",
    "metadata": {
      "name": "MultiFunc-Mix-VariantName",
      "category": "MultiFunc-Level",
      "subcategory": "Mix",
      "type": "variant",
      "source": "Generated",
      "language": "target_language",
      "difficulty": "easy/medium/hard/expert/master/grandmaster",
      "intervention": 0,
      "variant_type": "variant_type_label",
      "mixed_mechanisms": ["call chain", "Parameter Passing", "State Propagation", "Side Effect"],
    }
  }
]
```

```
        "primary_focus": "主要关注的机制组合",
        "analysis_complexity": "low/medium/high/extreme/ultimate"
    },
    "task": {
        "description": "English description of the mixed multi-function analysis task",
        "code": "Code containing multiple cross-function mechanisms to analyze",
        "answer": "Unique and accurate comprehensive analysis result"
    }
},
{下一个变式...}
]
```

特殊字段说明

mixed_mechanisms: 标识该变式包含的跨函数机制组合

primary_focus: 标识主要的分析焦点

analysis_complexity: 标识综合分析的复杂程度

生成目标

生成总计约60个跨函数推理大混合变式，分布如下：

简单混合变式：12个（2种机制组合）

中等混合变式：18个（2-3种机制组合）

复杂混合变式：18个（3种机制组合）

专家级混合变式：8个（3-4种机制组合）

大师级混合变式：3个（4种机制完全融合）

宗师级混合变式：1个（极限复杂场景）

重点测试场景

多机制协同分析：同时分析多种跨函数机制的协同效应

交互效应理解：理解不同机制间的相互影响和制约

综合状态追踪：跟踪多个维度的程序状态变化

复杂数据流分析：分析复杂数据流中的多重机制

系统级交互理解：理解系统级的跨函数交互模式

实际场景模拟：模拟实际软件系统的复杂交互

英文描述模板示例

"Analyze the comprehensive interaction of function calls, parameter passing, state changes, and side effects..."

"Determine the final result considering the combined effects of call chains, parameter modifications, and state propagation..."

"Trace the multi-dimensional effects through recursive calls with state management and side effects..."

"Calculate the result of complex cross-function interactions involving all major mechanisms..."

"Evaluate the comprehensive system behavior under multi-function scenario..."

混合模式优先级

实用导向：优先设计实际编程中常见的机制组合

教学价值：重点关注有教学和测试价值的组合

渐进复杂：从简单组合逐步增加到复杂综合

系统完整：确保各种机制组合的系统完整覆盖

答案唯一性保证策略

明确机制定义：清晰定义每种机制的语义和边界

具体执行路径：使用确定性的执行路径，避免歧义

避免竞争条件：避免产生竞争条件的并发场景

清晰时序：确保操作的时序和顺序明确

精确计算：确保数值计算的精确性和一致性

常见混合模式

数据处理管道：数据在多函数间的处理和传递

状态机实现：状态机模式的多机制实现

事务处理系统：事务处理的综合机制分析

缓存管理系统：缓存系统的多机制协同

异步处理框架：异步处理的复杂机制交互

资源管理系统：资源管理的综合机制分析

请基于此提示词和提供的所有种子任务，生成约60个跨函数推理大混合变式集合，严格确保答案准确且唯一，所有描述使用英文，输出格式为包含所有变式的JSON数组。

```
[  
  {  
    "id": "MF-MIX-V001",  
    "metadata": {  
      "name": "MultiFunc-Mix-ChainParameterBasic",  
      "category": "MultiFunc-Level",  
      "subcategory": "Mix",  
      "type": "variant",  
      "source": "Generated",  
      "language": "c",  
      "difficulty": "easy",  
      "intervention": 0,  
      "variant_type": "chain_parameter_mix",  
      "mixed_mechanisms": ["Call Chain", "Parameter Passing"],  
      "primary_focus": "Basic call chain with parameter transformation",  
      "analysis_complexity": "low"  
    },  
    "task": {  
      "description": "Analyze the parameter transformation through a simple call chain. What is the final return value when input is 15?",  
      "code": "int add_ten(int value) {\n        return value + 10;\n    }\n\nint multiply_by_two(int value) {\n    int result = add_ten(value);\n    return result * 2;\n}\n\nint subtract_five(int value) {\n    int result = multiply_by_two(value);\n    return result - 5;\n}\n\n// What is returned by subtract_five(15)?",  
      "answer": 45  
    },  
    {  
      "id": "MF-MIX-V002",  
      "metadata": {  
        "name": "MultiFunc-Mix-ChainStateBasic",  
        "category": "MultiFunc-Level",  
        "subcategory": "Mix",  
        "type": "variant",  
        "source": "Generated",  
        "language": "c",  
        "difficulty": "easy",  
        "intervention": 0,  
        "variant_type": "chain_state_mix",  
        "mixed_mechanisms": ["Call Chain", "State Propagation"],  
        "primary_focus": "Call chain affecting global state",  
        "analysis_complexity": "low"  
      }  
    }  
]
```

```

},
"task": {
  "description": "Track global state changes through function call chain. what is the final value of global_sum?",
  "code": "int global_sum = 0;\nvoid add_value(int value) {\n    global_sum += value;\n}\nvoid double_and_add(int value) {\n    add_value(value * 2);\n}\nvoid process_sequence() {\n    double_and_add(5);\n    double_and_add(8);\n    add_value(3);\n}\n// After calling process_sequence(), what is the value of global_sum?",
  "answer": 29
},
{
  "id": "MF-MIX-V003",
  "metadata": {
    "name": "MultiFunc-Mix-ParameterSideEffect",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "parameter_sideeffect_mix",
    "mixed_mechanisms": ["Parameter Passing", "Side Effect"],
    "primary_focus": "Parameter passing with side effects",
    "analysis_complexity": "low"
  },
  "task": {
    "description": "Analyze parameter modification side effects. What is the value of array[1] after function execution?",
    "code": "void modify_array(int *arr, int size) {\n    for (int i = 0; i < size; i++) {\n        arr[i] = arr[i] * 2 + 1;\n    }\n}\nint test_modification() {\n    int array[3] = {4, 6, 8};\n    modify_array(array, 3);\n    return array[1];\n}\n// What value is returned by test_modification()?",
    "answer": 13
  }
},
{
  "id": "MF-MIX-V004",
  "metadata": {
    "name": "MultiFunc-Mix-StateSideEffect",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "easy",
    "intervention": 0,
    "variant_type": "state_sideeffect_mix",
    "mixed_mechanisms": ["State Propagation", "Side Effect"],
    "primary_focus": "State changes with side effects",
    "analysis_complexity": "low"
  }
},

```

```

"task": {
    "description": "Track state changes with logging side effects. What is the final operation_count?",
    "code": "operation_count = 0\ncurrent_value = 10\n\ndef log_and_increment():\n    global operation_count, current_value\n    operation_count += 1\n    print(f\"Operation {operation_count}: value = {current_value}\")\n    current_value += 5\n\ndef process_operations():\n    log_and_increment()\n    log_and_increment()\n    log_and_increment()\n    return operation_count\n\n# What value is returned by process_operations()?",
    "answer": 3
},
{
    "id": "MF-MIX-V005",
    "metadata": {
        "name": "MultiFunc-Mix-RecursiveParameter",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "recursive_parameter_mix",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing"],
        "primary_focus": "Recursive calls with parameter accumulation",
        "analysis_complexity": "low"
    },
    "task": {
        "description": "Trace recursive parameter accumulation. What is the final return value?",
        "code": "int sum_to_n(int n, int accumulator) {\n    if (n <= 0) {\n        return accumulator;\n    }\n    return sum_to_n(n - 1, accumulator + n);\n}\n\nint calculate_result() {\n    return sum_to_n(5, 0);\n}\n\n// What value is returned by calculate_result()?",
        "answer": 15
    }
},
{
    "id": "MF-MIX-V006",
    "metadata": {
        "name": "MultiFunc-Mix-PointerChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "easy",
        "intervention": 0,
        "variant_type": "pointer_chain_mix",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing"],
        "primary_focus": "Pointer parameter passing through call chain",
        "analysis_complexity": "low"
    }
}

```

```

    },
    "task": {
        "description": "Follow pointer modifications through call chain. What is the final value pointed to by ptr?",
        "code": "void increment_value(int *ptr) {\n    *ptr += 10;\n}\n\nvoid double_value(int *ptr) {\n    *ptr *= 2;\n    increment_value(ptr);\n}\n\nint test_pointer_chain() {\n    int value = 5;\n    double_value(&value);\n    return value;\n}\n\n// What value is returned by test_pointer_chain()?",
        "answer": 20
    }
},
{
    "id": "MF-MIX-V007",
    "metadata": {
        "name": "MultiFunc-Mix-CallbackState",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "callback_state_mix",
        "mixed_mechanisms": ["Call Chain", "State Propagation"],
        "primary_focus": "Callback functions affecting global state",
        "analysis_complexity": "medium"
    },
    "task": {
        "description": "Analyze callback function impact on global state. What is the final state_counter value?",
        "code": "int state_counter = 0;\n\ntypedef void (*operation_func)(int);\n\nvoid add_operation(int value) {\n    state_counter += value;\n}\n\nvoid multiply_operation(int value) {\n    state_counter *= value;\n}\n\nvoid apply_operation(int value, operation_func op) {\n    op(value);\n}\n\nint execute_sequence() {\n    state_counter = 2;\n    apply_operation(3, add_operation);\n    apply_operation(4, multiply_operation);\n    apply_operation(10, add_operation);\n    return state_counter;\n}\n\n// What value is returned by execute_sequence()?",
        "answer": 30
    }
},
{
    "id": "MF-MIX-V008",
    "metadata": {
        "name": "MultiFunc-Mix-StructParameterChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "struct_parameter_chain",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing"],
        "primary_focus": "Struct parameter chains affecting global state"
    }
}

```

```

    "primary_focus": "Structure parameter transformation through chain",
    "analysis_complexity": "medium"
},
"task": {
    "description": "Track structure parameter changes through function chain. What is the final result.total value?",
    "code": "typedef struct {\n    int x;\n    int y;\n    int total;\n} Point;\n\nPoint calculate_sum(Point p) {\n    p.total = p.x + p.y;\n    return p;\n}\n\nPoint scale_point(Point p) {\n    p.x *= 2;\n    p.y *= 3;\n    return calculate_sum(p);\n}\n\nPoint add_offset(Point p) {\n    p.x += 5;\n    p.y += 10;\n    return scale_point(p);\n}\n\nint test_point_chain() {\n    Point p = {3, 4, 0};\n    Point result = add_offset(p);\n    return result.total;\n}\n\n// What value is returned by test_point_chain()?",
    "answer": 58
},
{
    "id": "MF-MIX-V009",
    "metadata": {
        "name": "MultiFunc-Mix-StaticParameterChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "static_parameter_chain",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
        "primary_focus": "Static variables with parameter passing in chain",
        "analysis_complexity": "medium"
    },
    "task": {
        "description": "Analyze static variable interaction with parameter passing. What is the final return value?",
        "code": "int increment_with_static(int value) {\n    static int counter = 0;\n    counter += value;\n    return counter;\n}\n\nint process_with_chain(int input) {\n    int step1 = increment_with_static(input);\n    int step2 = increment_with_static(step1);\n    return step2;\n}\n\nint test_static_chain() {\n    int result1 = process_with_chain(5);\n    int result2 = process_with_chain(3);\n    return result2;\n}\n\n// What value is returned by test_static_chain()?",
        "answer": 16
    }
},
{
    "id": "MF-MIX-V010",
    "metadata": {
        "name": "MultiFunc-Mix-ArraySideEffectChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
    }
}

```

```

        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "array_sideeffect_chain",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "Side Effect"],
        "primary_focus": "Array modifications through call chain",
        "analysis_complexity": "medium"
    },
    "task": {
        "description": "Track array modifications through function call chain. What is the final sum of array elements?",
        "code": "void double_elements(int *arr, int size) {\n    for (int i = 0; i < size; i++) {\n        arr[i] *= 2;\n    }\n}\nvoid add_indices(int *arr, int size) {\n    for (int i = 0; i < size; i++) {\n        arr[i] += i;\n    }\n    double_elements(arr, size);\n}\nint calculate_array_sum(int *arr, int size) {\n    add_indices(arr, size);\n    int sum = 0;\n    for (int i = 0; i < size; i++) {\n        sum += arr[i];\n    }\n    return sum;\n}\nint test_array_chain() {\n    int data[4] = {2, 4, 6, 8};\n    return calculate_array_sum(data, 4);\n} // What value is returned by test_array_chain()?",
        "answer": 52
    }
},
{
    "id": "MF-MIX-V011",
    "metadata": {
        "name": "MultiFunc-Mix-ExceptionStateChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "exception_state_chain",
        "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
        "primary_focus": "Exception handling affecting state through chain",
        "analysis_complexity": "medium"
    },
    "task": {
        "description": "Analyze exception handling effects on state through call chain. What is the final error_count?",
        "code": "error_count = 0\nsuccess_count = 0\n\ndef risky_division(a, b):\n    global error_count, success_count\n    try:\n        result = a / b\n        success_count += 1\n    except ZeroDivisionError:\n        error_count += 1\n    return 0\n\ndef process_division_chain(values):\n    result = 100\n    for i in range(len(values) - 1):\n        result = risky_division(result, values[i])\n        if result == 0:\n            break\n    return error_count\n\ndef test_exception_chain():\n    return process_division_chain([2, 5, 0, 4])\n\n# What value is returned by test_exception_chain()?",
        "answer": 1
    }
},
{
    "id": "MF-MIX-V012",
    "metadata": {

```

```

        "name": "MultiFunc-Mix-ClosureParameterChain",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "medium",
        "intervention": 1,
        "variant_type": "closure_parameter_chain",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
        "primary_focus": "Closure state with parameter passing chain",
        "analysis_complexity": "medium"
    },
    "task": {
        "description": "Track closure state through parameter passing chain. What is the final captured value?",
        "code": "def create_processor(initial):\n    value = initial\n    def process(input_val):\n        nonlocal value\n        value += input_val\n        return value\n    return process\n\ndef chain_processing(processor, values):\n    result = 0\n    for val in values:\n        result = processor(val)\n    return result\n\ndef test_closure_chain():\n    proc = create_processor(10)\n    result1 = chain_processing(proc, [5, 8, 3])\n    result2 = chain_processing(proc, [2, 7])\n    return result2\n\n# What value is returned by test_closure_chain()?",
        "answer": 35
    }
},
{
    "id": "MF-MIX-V013",
    "metadata": {
        "name": "MultiFunc-Mix-RecursiveStateSideEffect",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "recursive_state_sideeffect",
        "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
        "primary_focus": "Recursive calls with state and side effects",
        "analysis_complexity": "high"
    },
    "task": {
        "description": "Analyze recursive function with global state modifications and side effects. What is the final call_count?",
        "code": "int call_count = 0;\nint accumulator = 0;\n\nint recursive_accumulate(int n) {\n    call_count++;\n    accumulator += n;\n    if (n <= 1) {\n        return accumulator;\n    }\n    return recursive_accumulate(n - 1) + n;\n}\n\nvoid reset_counters() {\n    call_count = 0;\n    accumulator = 0;\n}\n\nint test_recursive_effects() {\n    reset_counters();\n    int result = recursive_accumulate(4);\n    return call_count;\n}\n\n// What value is returned by test_recursive_effects()?",
        "answer": 4
    }
}

```

```

        }
    },
    {
        "id": "MF-MIX-V014",
        "metadata": {
            "name": "MultiFunc-Mix-PointerStateModification",
            "category": "MultiFunc-Level",
            "subcategory": "Mix",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "pointer_state_modification",
            "mixed_mechanisms": ["Parameter Passing", "State Propagation", "Side Effect"],
            "primary_focus": "Pointer parameters affecting global state",
            "analysis_complexity": "high"
        },
        "task": {
            "description": "Track pointer parameter effects on global state. What is the final global_total?",
            "code": "int global_total = 0;\nint modification_count = 0;\n\nvoid\nmodify_and_track(int *ptr, int multiplier) {\n    *ptr *= multiplier;\n    global_total += *ptr;\n    modification_count++;\n}\n\nvoid chain_modifications(int *values, int size) {\n    for (int i = 0; i < size; i++) {\n        modify_and_track(&values[i], i + 1);\n    }\n}\n\nint test_pointer_state() {\n    int data[3] = {5, 10, 15};\n    chain_modifications(data, 3);\n    return global_total;\n}\n\n// What value is returned by test_pointer_state()?",
            "answer": 80
        }
    },
    {
        "id": "MF-MIX-V015",
        "metadata": {
            "name": "MultiFunc-Mix-CallbackStatePropagation",
            "category": "MultiFunc-Level",
            "subcategory": "Mix",
            "type": "variant",
            "source": "Generated",
            "language": "c",
            "difficulty": "hard",
            "intervention": 2,
            "variant_type": "callback_state_propagation",
            "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
            "primary_focus": "Callback functions with state propagation",
            "analysis_complexity": "high"
        },
        "task": {
            "description": "Analyze callback function chain with state propagation. What is the final processor_state?"
        }
    }
}

```



```

        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "object_method_chain",
        "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
        "primary_focus": "Object method chain with state changes",
        "analysis_complexity": "high"
    },
    "task": {
        "description": "Analyze object method call chain with state modifications. What is the final balance?",
        "code": "class Calculator:\n    def __init__(self, initial_value):\n        self.value = initial_value\n        self.operation_count = 0\n        \n    def add(self, amount):\n        self.value += amount\n        self.operation_count += 1\n        return self\n        \n    def multiply(self, factor):\n        self.value *= factor\n        self.operation_count += 1\n        return self\n        \n    def apply_bonus(self):\n        if self.operation_count >= 3:\n            self.value += 10\n        return self\n        \n    def get_final_value(self):\n        return self.value\n\ndef test_calculator_chain():\n    calc = Calculator(20)\n    result = calc.add(15).multiply(2).add(5).apply_bonus().get_final_value()\n    return result\n\n# What value is returned by test_calculator_chain()?",

        "answer": 85
    }
},
{
    "id": "MF-MIX-V018",
    "metadata": {
        "name": "MultiFunc-Mix-NestedCallbackState",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "c",
        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "nested_callback_state",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
        "primary_focus": "Nested callback functions affecting state",
        "analysis_complexity": "high"
    },
    "task": {
        "description": "Trace nested callback execution with state changes. What is the final execution_count?",
        "code": "int execution_count = 0;\nint state_value = 10;\n\ntypedef void\n(*callback_t)(int);\n\nvoid increment_callback(int value) {\n    execution_count++;\n    state_value += value;\n}\n\nvoid multiply_callback(int value) {\n    execution_count++;\n    state_value *= value;\n}\n\nvoid execute_with_callback(int input, callback_t cb) {\n    cb(input);\n    if (execution_count < 3) {\n        execute_with_callback(input + 1, cb);\n    }\n}\n\nint test_nested_callbacks() {\n    execute_with_callback(2, increment_callback);\n    return execution_count;\n}\n\n// What value is returned by test_nested_callbacks()?",

        "answer": 3
    }
},

```

```
{
  "id": "MF-MIX-V019",
  "metadata": {
    "name": "MultiFunc-Mix-ComplexParameterTransform",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "complex_parameter_transform",
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
    "primary_focus": "Complex parameter transformation with state",
    "analysis_complexity": "high"
  },
  "task": {
    "description": "Analyze complex parameter transformation through multiple function layers. What is the final computed result?",
    "code": "typedef struct {\n    int data[3];\n    int multiplier;\n    int sum;\n} DataStruct;\n\nstatic int transform_count = 0;\n\nDataStruct\ntransform_data(DataStruct input) {\n    transform_count++;\n    DataStruct result = input;\n\n    for (int i = 0; i < 3; i++) {\n        result.data[i] *= result.multiplier;\n    }\n\n    result.sum = 0;\n    for (int i = 0; i < 3; i++) {\n        result.sum +=\n        result.data[i];\n    }\n\n    result.multiplier++;\n\n    return result;\n}\n\nDataStruct\napply_chain_transform(DataStruct input, int iterations) {\n    DataStruct current = input;\n\n    for (int i = 0; i < iterations; i++) {\n        current = transform_data(current);\n    }\n\n    return current;\n}\n\nint test_complex_transform() {\n    DataStruct initial = {{2,\n3, 4}, 2, 0};\n\n    DataStruct result = apply_chain_transform(initial, 2);\n\n    return\n    result.sum + transform_count;\n}\n\n// What value is returned by test_complex_transform()?",
    "answer": 146
  }
},
{
  "id": "MF-MIX-V020",
  "metadata": {
    "name": "MultiFunc-Mix-RecursiveStateAccumulation",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "recursive_state_accumulation",
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation", "Side Effect"],
    "primary_focus": "Recursive state accumulation with side effects",
    "analysis_complexity": "extreme"
  },
  "task": {
    "description": "Track recursive state accumulation with multiple side effects. What is the final global_accumulator value?"
  }
}
```

```

        "code": "int global_accumulator = 0;\nint call_depth = 0;\nint max_depth = 0;\n\nint fibonacci_with_state(int n) {\n    call_depth++;\n    if (call_depth > max_depth)\n        max_depth = call_depth;\n    global_accumulator += n;\n    int result;\n    if (n <= 1) {\n        result = n;\n    } else {\n        result = fibonacci_with_state(n - 1) + fibonacci_with_state(n - 2);\n    }\n    call_depth--;\n    return result;\n}\n\nvoid reset_state() {\n    global_accumulator = 0;\n    call_depth = 0;\n    max_depth = 0;\n}\n\nint test_fibonacci_state() {\n    reset_state();\n    int fib_result = fibonacci_with_state(4);\n    return global_accumulator;\n}\n\n// what value is returned by test_fibonacci_state()?",\n        "answer": 19\n    },\n},\n{\n    "id": "MF-MIX-V021",\n    "metadata": {\n        "name": "MultiFunc-Mix-DoublePointerChain",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "double_pointer_chain",\n        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "Side Effect"],\n        "primary_focus": "Double pointer manipulation through call chain",\n        "analysis_complexity": "extreme"\n    },\n    "task": {\n        "description": "Analyze double pointer manipulation chain with memory effects.\nWhat is the final dereferenced value?",\n        "code": "int allocation_counter = 0;\n\nvoid allocate_and_set(int **ptr, int value) {\n    *ptr = malloc(sizeof(int));\n    **ptr = value;\n    allocation_counter++;\n}\n\nvoid reallocate_and_modify(int **ptr, int new_value) {\n    free(*ptr);\n    allocate_and_set(ptr, new_value * 2);\n}\n\nvoid chain_reallocations(int **ptr, int *values, int count) {\n    allocate_and_set(ptr, values[0]);\n    for (int i = 1; i < count; i++) {\n        reallocate_and_modify(ptr, values[i]);\n    }\n}\n\nint test_double_pointer_chain() {\n    int *ptr = NULL;\n    int values[] = {5, 8, 12};\n    chain_reallocations(&ptr, values, 3);\n    int result = *ptr;\n    free(ptr);\n    return result;\n}\n\n// what value is returned by test_double_pointer_chain()?",\n        "answer": 24\n    },\n},\n{\n    "id": "MF-MIX-V022",\n    "metadata": {\n        "name": "MultiFunc-Mix-ExceptionPropagationChain",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "expert",\n    }\n}
```

```

    "intervention": 3,
    "variant_type": "exception_propagation_chain",
    "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
    "primary_focus": "Exception propagation through call chain",
    "analysis_complexity": "extreme"
},
"task": {
    "description": "Track exception propagation effects through call chain. what is the final recovery_count?",
    "code": "recovery_count = 0\nprocessed_items = 0\n\ndef risky_processor(value):\n    global processed_items\n    if value < 0:\n        raise\nValueError(\"Negative value\")\n    if value > 50:\n        raise\nOverflowError(\"Value too\nlarge\")\n    processed_items += 1\n    return value * 2\n\ndef safe_processor(value):\n    global recovery_count\n    try:\n        return risky_processor(value)\n    except\nValueError:\n        recovery_count += 1\n    return 0\n    except\nOverflowError:\n        recovery_count += 2\n    return 100\n\ndef chain_processor(values):\n    results = []\n    for value in values:\n        try:\n            result = safe_processor(value)\n            results.append(result)\n        except\nException:\n            recovery_count += 10\n# Unexpected exception\n    results.append(-1)\n    return recovery_count\n\ndef\ntest_exception_chain():\n    global recovery_count, processed_items\n    recovery_count = 0\n    processed_items = 0\n    values = [10, -5, 60, 25, 80]\n    return\nchain_processor(values)\n\n# What value is returned by test_exception_chain()?",\n    "answer": 5
}
{
    "id": "MF-MIX-V023",
    "metadata": {
        "name": "MultiFunc-Mix-ClosureNestedState",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "closure_nested_state",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
        "primary_focus": "Nested closure state management",
        "analysis_complexity": "extreme"
},
"task": {
    "description": "Analyze nested closure state interactions. what is the final captured state value?",\n
}

```

```

    "code": "def create_stateful_processor(initial_state):\n    state = {'value':\ninitial_state, 'count': 0}\n    def create_inner_processor(multiplier):\n        def\nprocess(input_val):\n            state['value'] += input_val * multiplier\n            state['count'] += 1\n            return state['value']\n        return process\n    \n    def get_state():\n        return state['value']\n    \n    return create_inner_processor,\nget_state\n\ndef test_nested_closure():\n    processor_factory, get_state =\ncreate_stateful_processor(10)\n    \n    proc1 = processor_factory(2)\n    proc2 =\nprocessor_factory(3)\n    \n    result1 = proc1(5) # 10 + 5*2 = 20\n    result2 = proc2(4)\n    # 20 + 4*3 = 32\n    result3 = proc1(3) # 32 + 3*2 = 38\n    \n    return get_state()\n\n#\nWhat value is returned by test_nested_closure()?",\n    "answer": 38\n}\n},\n{\n    "id": "MF-MIX-V024",\n    "metadata": {\n        "name": "MultiFunc-Mix-StaticArrayChain",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "static_array_chain",\n        "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],\n        "primary_focus": "Static array state through call chain",\n        "analysis_complexity": "extreme"\n    },\n    "task": {\n        "description": "Track static array state modifications through function chain.\nWhat is the final array sum?",\n        "code": "int rotate_and_modify(int value, int position) {\n    static int\nbuffer[4] = {1, 2, 3, 4};\n    static int current_pos = 0;\n    // Rotate position\n    current_pos = (current_pos + position) % 4;\n    // Modify at current position\n    buffer[current_pos] += value;\n    // Calculate sum\n    int sum = 0;\n    for (int i = 0; i < 4; i++) {\n        sum += buffer[i];\n    }\n    return sum;\n}\n\nint\napply_operations() {\n    int result = 0;\n    result = rotate_and_modify(5, 1); // pos 1,\n    buffer[1] = 7\n    result = rotate_and_modify(3, 2); // pos 3, buffer[3] = 7\n    result =\n    rotate_and_modify(8, 1); // pos 0, buffer[0] = 9\n    return result;\n}\n\nint\ntest_static_array_chain() {\n    return apply_operations();\n}\n\n// What value is returned by test_static_array_chain()?",\n        "answer": 25\n    },\n    "id": "MF-MIX-V025",\n    "metadata": {\n        "name": "MultiFunc-Mix-MemoryLeakTracking",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n    }\n}
```

```
        "language": "c",
        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "memory_leak_tracking",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "Side Effect"],
        "primary_focus": "Memory allocation tracking through chain",
        "analysis_complexity": "extreme"
    },
    "task": {
        "description": "Track memory allocation and deallocation through function chain.  
What is the final net_allocations?",  

        "code": "int net_allocations = 0;\nint total_allocated_bytes = 0;\n\nvoid* tracked_malloc(size_t size) {\n    net_allocations++;\n    total_allocated_bytes += size;\n    return malloc(size);\n}\n\nvoid tracked_free(void* ptr) {\n    if (ptr != NULL) {\n        net_allocations--;\n        free(ptr);\n    }\n}\n\nint* create_array_chain(int size, int initial_value) {\n    int* arr = (int*)tracked_malloc(size * sizeof(int));\n    if (arr == NULL) return NULL;\n    for (int i = 0; i < size; i++) {\n        arr[i] = initial_value + i;\n    }\n    return arr;\n}\n\nvoid process_arrays() {\n    int* arr1 = create_array_chain(5, 10);\n    int* arr2 = create_array_chain(3, 20);\n    int* arr3 = create_array_chain(4, 30);\n    tracked_free(arr1);\n    tracked_free(arr3);\n    // arr2 is intentionally not freed to test leak detection\n}\n\nint test_memory_tracking() {\n    net_allocations = 0;\n    total_allocated_bytes = 0;\n    process_arrays();\n    return net_allocations;\n}\n\n// What value is returned by test_memory_tracking()?",  

        "answer": 1
    }
},
{
    "id": "MF-MIX-V026",
    "metadata": {
        "name": "MultiFunc-Mix-SystemSimulation",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "master",
        "intervention": 4,
        "variant_type": "system_simulation",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation", "Side Effect"],
        "primary_focus": "Complex system simulation with all mechanisms",
        "analysis_complexity": "ultimate"
    },
    "task": {
        "description": "Simulate a complex processing system with resource management, error handling, and state tracking. What is the final system_health_score?",
```

```

    "code": "class ResourceManager:\n        def __init__(self):\n            self.allocated_resources = {}\n            self.error_count = 0\n            self.operation_count = 0\n            self.health_score = 100\n            \n        def allocate_resource(self, resource_id, size):\n            self.operation_count += 1\n            if resource_id in self.allocated_resources:\n                self.error_count += 1\n                self.health_score -= 5\n            return False\n            \n        def deallocate_resource(self, resource_id):\n            self.operation_count += 1\n            if resource_id not in self.allocated_resources:\n                self.error_count += 1\n                self.health_score -= 3\n            return False\n            \n        self.allocated_resources.pop(resource_id)\n        self.health_score = min(100, self.health_score + (size // 20))\n        return True\n        \n    def process_resource(self, resource_id, processor_func):\n        self.operation_count += 1\n        if resource_id not in self.allocated_resources:\n            self.error_count += 1\n            return None\n            \n        try:\n            result = processor_func(self.allocated_resources[resource_id])\n            self.health_score = min(100, self.health_score + 2)\n            return result\n        except Exception:\n            self.error_count += 1\n            self.health_score -= 10\n            return None\n\nrm = ResourceManager()\n\n#safe_divide_processor(size):\n    if size == 0:\n        raise ValueError(\"Cannot divide by zero\")\n    return 1000 // size\n\n#multiply_processor(size):\n    return size * 3\n\n#complex_workflow(resource_configs):\n    results = []\n    for config in resource_configs:\n        resource_id, size, processor_type = config\n        # Allocation phase\n        if rm.allocate_resource(resource_id, size):\n            # Processing phase\n            if processor_type == 'divide':\n                result = rm.process_resource(resource_id, safe_divide_processor)\n            elif processor_type == 'multiply':\n                result = rm.process_resource(resource_id, multiply_processor)\n            else:\n                result = None\n        results.append(result if result is not None else 0)\n        # Deallocation phase (only for even resource_ids)\n        if resource_id % 2 == 0:\n            rm.deallocate_resource(resource_id)\n        else:\n            results.append(-1)\n    return rm.health_score\n\n# test_system_simulation():\nconfigs = [\n    (1, 50, 'multiply'),\n    # Allocate, process, keep\n    (2, 25, 'divide'),\n    # Allocate, process, deallocate\n    (3, 0, 'divide'),\n    # Allocate, process (error), keep\n    (4, 100, 'multiply'),\n    # Allocate, process, deallocate\n    (1, 30, 'multiply'),\n    # Allocation error (duplicate)\n    (5, 75, 'unknown')\n    # Allocate, process (error), keep\n]\n\nreturn complex_workflow(configs)\n\n# What value is returned by test_system_simulation()?",\n    "answer": 64\n}\n},\n{\n    "id": "MF-MIX-V027",\n    "metadata": {\n        "name": "MultiFunc-Mix-DatabaseTransaction",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "master",\n        "intervention": 4,\n        "variant_type": "database_transaction",\n    }\n}

```

```
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation",  
"Side Effect"],  
        "primary_focus": "Database transaction simulation",  
        "analysis_complexity": "ultimate"  
    },  
    "task": {  
        "description": "Simulate database transaction processing with rollback  
mechanisms and state consistency. What is the final committed_transaction_count?",  
    }  
}
```

```

    "code": "class DatabaseSimulator:\n        def __init__(self):\n            self.tables\n            = {'users': {}, 'accounts': {}}\n            self.transaction_log = []\n            self.active_transaction = None\n            self.committed_transaction_count = 0\n            self.rollback_count = 0\n            \n            def begin_transaction(self, tx_id):\n                if\n                    self.active_transaction is not None:\n                        return False\n                    \n                    self.active_transaction = {\n                        'id': tx_id,\n                        'operations': []\n                    }\n                    \n                    'backup': {\n                        'users': self.tables['users'].copy(),\n                        'accounts': self.tables['accounts'].copy()\n                    }\n                    \n                    return True\n                \n                def insert_record(self, table, key, value):\n                    if self.active_transaction is\n                        None:\n                            return False\n                        \n                        if key in self.tables[table]:\n                            return False # Duplicate key\n                            \n                            self.tables[table][key] = value\n                        \n                        self.active_transaction['operations'].append(('INSERT', table, key, value))\n                        \n                        return True\n                    \n                    def update_record(self, table, key, value):\n                        if\n                            self.active_transaction is None:\n                                return False\n                                \n                                if key not in\n                                    self.tables[table]:\n                                        return False # Key not found\n                                        \n                                        old_value\n                                        = self.tables[table][key]\n                                        \n                                        self.tables[table][key] = value\n                                \n                                self.active_transaction['operations'].append(('UPDATE', table, key, old_value, value))\n                                \n                                return True\n                            \n                            def commit_transaction(self):\n                                if self.active_transaction\n                                    is None:\n                                        return False\n                                        \n                                        self.transaction_log.append(self.active_transaction)\n                                        \n                                        self.committed_transaction_count += 1\n                                        \n                                        self.active_transaction = None\n                                        \n                                        return True\n                                    \n                                    def rollback_transaction(self):\n                                        if self.active_transaction\n                                            is None:\n                                                return False\n                                                \n                                                # Restore from backup\n                                                \n                                                self.tables = self.active_transaction['backup']\n                                                \n                                                self.rollback_count += 1\n                                                \n                                                self.active_transaction = None\n                                                \n                                                return True\n                                            \n                                            ndb = DatabaseSimulator()\n                                            \n                                            def\n                                                execute_user_creation(user_id, name, initial_balance):\n                                                    success = True\n                                                    \n                                                    #\n                                                    # Insert user\n                                                    if not db.insert_record('users', user_id, {'name': name}):\n                                                        success\n                                                        = False\n                                                        \n                                                        # Insert account\n                                                        if success and not db.insert_record('accounts',\n                                                            user_id, {'balance': initial_balance}):\n                                                                success\n                                                                = False\n                                                                \n                                                                return\n                                                                success\n                                                                \n                                                                def\n                                                                execute_transfer(from_user, to_user, amount):\n                                                                    # Check if users exist\n                                                                    if from_user not in db.tables['accounts'] or to_user not in db.tables['accounts']:\n                                                                        return False\n                                                                        \n                                                                        # Check balance\n                                                                        if db.tables['accounts'][from_user]['balance']\n                                                                            < amount:\n                                                                                return False\n                                                                                \n                                                                                # Update balances\n                                                                                new_from_balance\n                                                                                =\n                                                                                db.tables['accounts'][from_user]['balance'] - amount\n                                                                                \n                                                                                new_to_balance\n                                                                                =\n                                                                                db.tables['accounts'][to_user]['balance'] + amount\n                                                                                \n                                                                                success\n                                                                                = True\n                                                                                \n                                                                                success\n                                                                                &=\n                                                                                db.update_record('accounts', from_user, {'balance': new_from_balance})\n                                                                                \n                                                                                success\n                                                                                &=\n                                                                                db.update_record('accounts', to_user, {'balance': new_to_balance})\n                                                                                \n                                                                                return\n                                                                                success\n                                                                \n                                                                def\n                                                                process_banking_operations():\n                                                                    operations = [\n                                                                        ('create_user', 1,\n                            'Alice', 1000),\n                            ('create_user', 2, 'Bob', 500),\n                            ('transfer', 1, 2, 200),\n                            ('create_user', 1, 'Charlie', 300),\n                            # Duplicate user\n                            ('transfer', 2, 1, 100),\n                            ('transfer', 1, 3, 50),\n                            # Non-existent user\n                            ('create_user', 3, 'Charlie', 300),\n                            ('transfer', 1, 3, 150)\n                        ]\n                        \n                        for operation in\n                            operations:\n                            if operation[0] == 'create_user':\n                                \n                                _, user_id, name,\n                                balance = operation\n                                \n                                if\n                                    db.begin_transaction(f"create_{user_id}"):
                                    \n                                    execute_user_creation(user_id, name, balance)\n                                    \n                                    db.commit_transaction()\n                                \n                                else:\n                                    db.rollback_transaction()\n                            \n                            elif operation[0] == 'transfer':\n                                \n                                _, from_user, to_user, amount = operation\n                                \n                                if\n                                    db.begin_transaction(f"transfer_{from_user}_{to_user}"):
                                    \n                                    execute_transfer(from_user, to_user, amount)\n                                    \n                                    db.commit_transaction()\n                                \n                                else:\n                                    db.rollback_transaction()\n                            \n                            return\n                        \n                    \n                \n            \n        \n    "

```

```
db.committed_transaction_count\n\nndef test_database_simulation():\n    return\nprocess_banking_operations()\n\n# what value is returned by test_database_simulation()?",\n    "answer": 5\n}\n},\n{\n    "id": "MF-MIX-V028",\n    "metadata": {\n        "name": "MultiFunc-Mix-CompilerOptimizer",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "c",\n        "difficulty": "master",\n        "intervention": 4,\n        "variant_type": "compiler_optimizer",\n        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation",\n"Side Effect"],\n        "primary_focus": "Compiler optimization simulation",\n        "analysis_complexity": "ultimate"\n    },\n    "task": {\n        "description": "Simulate a compiler optimization pipeline with multiple passes\nand state tracking. What is the final optimization_score?",\n    }
```

```

"code": "typedef struct {\n    int *instructions;\n    int count;\n    int\nallocated;\n} InstructionList;\n\ntypedef struct {\n    int dead_code_removed;\n    int\nconstants_folded;\n    int redundant_ops_removed;\n    int optimization_score;\n} OptimizationStats;\n\nstatic OptimizationStats global_stats = {0, 0, 0,\n0};\n\nInstructionList* create_instruction_list(int initial_capacity) {\n    InstructionList* list = malloc(sizeof(InstructionList));\n    list->instructions =\n    malloc(initial_capacity * sizeof(int));\n    list->count = 0;\n    list->allocated =\n    initial_capacity;\n    return list;\n}\n\nvoid add_instruction(InstructionList* list, int\ninstruction) {\n    if (list->count >= list->allocated) {\n        list->allocated *= 2;\n\n        list->instructions =\n        realloc(list->instructions, list->allocated * sizeof(int));\n\n        list->instructions[list->count++] = instruction;\n    }\n\n    nint\nremove_dead_code_pass(InstructionList* list) {\n        int removed = 0;\n\n        // Remove\n        NOPs (instruction code 0)\n        int write_pos = 0;\n        for (int read_pos = 0; read_pos <\n            list->count; read_pos++) {\n            if (list->instructions[read_pos] != 0) {\n                list->instructions[write_pos++] = list->instructions[read_pos];\n            } else {\n                removed++;\n            }\n        }\n\n        list->count = write_pos;\n\n        global_stats.dead_code_removed += removed;\n        global_stats.optimization_score += removed * 2;\n\n        return removed;\n    }\n\n    nint constant_folding_pass(InstructionList* list) {\n        int folded = 0;\n\n        // Look for ADD_CONST followed by ADD_CONST and combine them\n        // ADD_CONST = 100 + value\n        for (int i = 0; i < list->count - 1; i++) {\n            if (list->instructions[i] >= 100 && list->instructions[i] < 200 &&\n                list->instructions[i + 1] >= 100 && list->instructions[i + 1] < 200) {\n                int val1 = list->instructions[i] - 100;\n                int val2 = list->instructions[i + 1] -\n                100;\n                int combined = val1 + val2;\n                list->instructions[i] = 100 + combined;\n                list->instructions[i + 1] = 0; // Mark as\n                NOP\n                folded++;\n            }\n        }\n\n        global_stats.constants_folded += folded;\n        global_stats.optimization_score += folded * 3;\n\n        return folded;\n    }\n\n    nint redundancy_elimination_pass(InstructionList* list) {\n        int eliminated = 0;\n\n        // Remove consecutive identical instructions\n        int\nwrite_pos = 0;\n        for (int read_pos = 0; read_pos < list->count; read_pos++) {\n            if (read_pos == 0 || list->instructions[read_pos] != list->instructions[read_pos - 1]) {\n                list->instructions[write_pos++] = list->instructions[read_pos];\n            } else {\n                eliminated++;\n            }\n        }\n\n        list->count = write_pos;\n\n        global_stats.redundant_ops_removed += eliminated;\n        global_stats.optimization_score +=\n        eliminated * 1;\n\n        return eliminated;\n    }\n\n    nint run_optimization_pipeline(InstructionList* list, int passes) {\n        int total_improvements = 0;\n\n        for (int pass = 0; pass < passes; pass++) {\n            int\nimprovements_this_pass = 0;\n            improvements_this_pass +=\n            constant_folding_pass(list);\n            improvements_this_pass +=\n            remove_dead_code_pass(list);\n            improvements_this_pass +=\n            redundancy_elimination_pass(list);\n            total_improvements +=\n            improvements_this_pass;\n\n            // If no improvements in this pass, stop early\n            if (improvements_this_pass == 0) {\n                break;\n            }\n        }\n\n        return total_improvements;\n    }\n\n    nint test_compiler_optimization() {\n        // Reset global\n        stats\n        global_stats.dead_code_removed = 0;\n        global_stats.constants_folded = 0;\n        global_stats.redundant_ops_removed = 0;\n        global_stats.optimization_score = 0;\n\n        InstructionList* program = create_instruction_list(20);\n\n        // Add sample\n        instructions\n        int sample_instructions[] = {\n            105, // ADD_CONST 5\n            103,\n            // ADD_CONST 3\n            0, // NOP\n            200, // LOAD\n            200, // LOAD\n            (duplicate)\n            107, // ADD_CONST 7\n            0, // NOP\n            102, // ADD_CONST\n            2\n            300, // STORE\n            300, // STORE (duplicate)\n            0 // NOP\n        };\n\n        for (int i = 0; i < 11; i++) {\n            add_instruction(program,\n            sample_instructions[i]);\n        }\n\n        // Run optimization pipeline\n    }\n}
```

```
run_optimization_pipeline(program, 3);\\n      \\n      int final_score =  
global_stats.optimization_score;\\n      \\n      // Cleanup\\n      free(program->instructions);\\n  
free(program);\\n      \\n      return final_score;\\n}\\n\\n// what value is returned by  
test_compiler_optimization()?",  
      "answer": 11  
    }  
,  
{  
  "id": "MF-MIX-V029",  
  "metadata": {  
    "name": "MultiFunc-Mix-NetworkProtocolStack",  
    "category": "MultiFunc-Level",  
    "subcategory": "Mix",  
    "type": "variant",  
    "source": "Generated",  
    "language": "python",  
    "difficulty": "grandmaster",  
    "intervention": 5,  
    "variant_type": "network_protocol_stack",  
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation",  
"Side Effect"],  
    "primary_focus": "Network protocol stack simulation",  
    "analysis_complexity": "ultimate"  
  },  
  "task": {  
    "description": "Simulate a complete network protocol stack with error handling,  
retry mechanisms, and state management. what is the final successful_transmissions count?",  
  }  
}
```

```

    "code": "import random\nrandom.seed(42) # For reproducible results\n\nclass NetworkStats:\n    def __init__(self):\n        self.packets_sent = 0\n        self.packets_received = 0\n        self.packets_dropped = 0\n        self.retransmissions = 0\n        self.successful_transmissions = 0\n        self.error_count = 0\n\nclass Packet:\n    def __init__(self, data, sequence_num, destination):\n        self.data = data\n        self.sequence_num = sequence_num\n        self.destination = destination\n        self.checksum = self.calculate_checksum()\n        self.retry_count = 0\n    self.max_retries = 3\n    def calculate_checksum(self):\n        return sum(ord(c) for c in self.data) % 256\n    def is_corrupted(self):\n        # Simulate corruption based on data content\n        return self.checksum % 7 == 0 # Deterministic corruption\n\nclass NetworkLayer:\n    def __init__(self, stats):\n        self.stats = stats\n        self.routing_table = {'A': 1, 'B': 2, 'C': 3}\n        self.congestion_window = 4\n    def route_packet(self, packet):\n        if packet.destination not in self.routing_table:\n            self.stats.error_count += 1\n            return None\n        # Simulate routing delay affecting congestion\n        route_cost = self.routing_table[packet.destination]\n        if route_cost > self.congestion_window:\n            self.congestion_window = max(1, self.congestion_window - 1)\n            return None\n        self.congestion_window = min(8, self.congestion_window + 1)\n        return packet\n\nclass TransportLayer:\n    def __init__(self, stats):\n        self.stats = stats\n        self.sequence_number = 0\n        self.acknowledged_packets = set()\n        self.pending_packets = {}\n    def send_packet(self, data, destination, network_layer):\n        packet = Packet(data, self.sequence_number, destination)\n        self.sequence_number += 1\n        self.pending_packets[packet.sequence_num] = packet\n        # Try to route through network layer\n        routed_packet = network_layer.route_packet(packet)\n        if routed_packet is None:\n            return False\n        self.stats.packets_sent += 1\n        return self.transmit_packet(routed_packet)\n    def transmit_packet(self, packet):\n        # Simulate transmission\n        if packet.is_corrupted():\n            self.stats.packets_dropped += 1\n            return self.retry_packet(packet)\n        # Successful transmission\n        self.stats.packets_received += 1\n        self.acknowledged_packets.add(packet.sequence_num)\n        if packet.sequence_num in self.pending_packets:\n            del self.pending_packets[packet.sequence_num]\n            self.stats.successful_transmissions += 1\n            return True\n        def retry_packet(self, packet):\n            packet.retry_count += 1\n            if packet.retry_count > packet.max_retries:\n                self.stats.error_count += 1\n                if packet.sequence_num in self.pending_packets:\n                    del self.pending_packets[packet.sequence_num]\n                return False\n            self.stats.retransmissions += 1\n            return self.transmit_packet(packet)\n\nclass ApplicationLayer:\n    def __init__(self):\n        self.stats = NetworkStats()\n        self.network_layer = NetworkLayer(self.stats)\n        self.transport_layer = TransportLayer(self.stats)\n        self.message_queue = []\n    def send_message(self, message, destination):\n        # Fragment large messages\n        fragments = self.fragment_message(message)\n        success_count = 0\n        for fragment in fragments:\n            if self.transport_layer.send_packet(fragment, destination, self.network_layer):\n                success_count += 1\n        return success_count == len(fragments)\n    def fragment_message(self, message):\n        # Split message into 10-character fragments\n        fragments = []\n        for i in range(0, len(message), 10):\n            fragments.append(message[i:i+10])\n        return fragments\n    def batch_send_messages(self, messages):\n        results = []\n        for message, destination in messages:\n            success = self.send_message(message, destination)\n            results.append(success)\n        return results\n\ndef test_network_protocol_stack():\n    app = ApplicationLayer()\n    # Test messages with various characteristics\n    test_messages = [\n        ("Hello", "A"),\n        ("World", "B"),\n        ("!", "C"),\n        (" ", "D"),\n        ("1234567890", "E"),\n        ("A" * 100, "F")\n    ]\n    for message, destination in test_messages:\n        app.send_message(message, destination)\n    print(f"\n\nNetwork Layer Statistics:\n  Packets Sent: {app.stats.packets_sent}\n  Packets Received: {app.stats.packets_received}\n  Packets Dropped: {app.stats.packets_dropped}\n  Retransmissions: {app.stats.retransmissions}\n  Successful Transmissions: {app.stats.successful_transmissions}\n  Error Count: {app.stats.error_count}\n\nTransport Layer Statistics:\n  Sequence Number: {app.stats.sequence_number}\n  Acknowledged Packets: {app.stats.acknowledged_packets}\n  Pending Packets: {app.stats.pending_packets}\n\nApplication Layer Statistics:\n  Network Layer: {app.network_layer}\n  Transport Layer: {app.transport_layer}\n  Message Queue: {app.message_queue}\n\n")

```

```

world!", \"A\"),           # Short message\n      \"This is a longer message that will
be fragmented\", \"B\"),  # Long message\n      \"Error prone data with checksum
issues\", \"C\"),       # Potentially corrupted\n      \"Quick msg\", \"A\"),           #
short message\n      \"Another test message for reliability\", \"B\"),  # Medium
message\n      \"Final transmission test\", \"C\"),   # Final message\n      \"Invalid
destination\", \"D\"),  # Invalid destination\n      \"Retry test message\", \"A\"),  #
# Retry test\n  ]\n  \n  # Send all messages\n  results =
app.batch_send_messages(test_messages)\n  \n  return
app.stats.successful_transmissions\n\n# what value is returned by
test_network_protocol_stack()?",

  "answer": 17
}

},
{
  "id": "MF-MIX-V030",
  "metadata": {
    "name": "MultiFunc-Mix-RealTimeSystem",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "realtime_system_mix",
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
    "primary_focus": "Real-time system task scheduling",
    "analysis_complexity": "medium"
  },
  "task": {
    "description": "Simulate real-time task scheduling with priority and timing
constraints. what is the final completed_tasks count?",

    "code": "typedef struct {\n    int task_id;\n    int priority;\n    int
execution_time;\n    int deadline;\n} Task;\n\nstatic int completed_tasks = 0;\nstatic int
current_time = 0;\nstatic int missed_deadlines = 0;\n\nint execute_task(Task task) {\n    current_time += task.execution_time;\n    if (current_time <= task.deadline) {\n        completed_tasks++;\n        return 1; // Success\n    } else {\n        missed_deadlines++;\n        return 0; // Missed deadline\n    }\n}\n\nint
schedule_tasks(Task *tasks, int count) {\n    // Simple priority-based scheduling\n    for
(int i = 0; i < count - 1; i++) {\n        for (int j = 0; j < count - i - 1; j++) {\n            if (tasks[j].priority < tasks[j + 1].priority) {\n                Task temp =
tasks[j];\n                tasks[j] = tasks[j + 1];\n                tasks[j + 1] = temp;\n            }
        }
    }
    int successful = 0;\n    for (int i = 0; i < count; i++) {\n        successful += execute_task(tasks[i]);\n    }
    return
successful;\n}\n\nint test_realtime_system() {\n    completed_tasks = 0;\n    current_time =
0;\n    missed_deadlines = 0;\n    Task tasks[] = {\n        {1, 3, 5, 20}, // High
priority, short execution\n        {2, 1, 8, 25}, // Low priority, medium execution\n        {3, 2, 4, 15}, // Medium priority, short execution\n        {4, 3, 6, 30}, // High
priority, medium execution\n        {5, 1, 3, 35} // Low priority, short execution\n    };
    schedule_tasks(tasks, 5);\n    return completed_tasks;\n}\n\n// what
value is returned by test_realtime_system()?",

  "answer": 4
}

```

```

},
{
  "id": "MF-MIX-V031",
  "metadata": {
    "name": "MultiFunc-Mix-CacheSystem",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "cache_system_mix",
    "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
    "primary_focus": "Cache management with LRU policy",
    "analysis_complexity": "medium"
  },
  "task": {
    "description": "Implement LRU cache with access tracking and eviction counting.
what is the final cache_hits count?",
    "code": "class LRUcache:\n    def __init__(self, capacity):\n        self.capacity = capacity\n        self.cache = {}\n        self.access_order = []\n        self.cache_hits = 0\n        self.cache_misses = 0\n        self.evictions = 0\n    def get(self, key):\n        if key in self.cache:\n            # Move to end (most recently used)\n            self.access_order.remove(key)\n            self.access_order.append(key)\n            self.cache_hits += 1\n            return self.cache[key]\n        else:\n            self.cache_misses += 1\n            return None\n    def put(self, key, value):\n        if key in self.cache:\n            # Update existing key\n            self.cache[key] = value\n            self.access_order.remove(key)\n            self.access_order.append(key)\n        else:\n            # Add new key\n            if len(self.cache) >= self.capacity:\n                lru_key = self.access_order.pop(0)\n                self.evictions += 1\n                self.access_order.append(key)\n            self.cache[key] = value\n            self.access_order.append(key)\n        def access_pattern(self, operations):\n            if op_type == 'get':\n                self.get(key)\n            elif op_type == 'put':\n                self.put(key, value)\n            return self.cache_hits\n        test_cache_system():\n            cache = LRUcache(3)\n            operations = [\n                ('put', 'a', 1),\n                ('put', 'b', 2),\n                ('get', 'a', None),\n                ('put', 'c', 3),\n                ('get', 'b', None),\n                ('put', 'd', 4),\n                ('get', 'c', None),\n                ('put', 'e', 5),\n                ('get', 'd', None),\n                ('get', 'b', None),\n                ('get', 'd', None),\n                ('get', 'e', None)\n            ]\n            return cache.access_pattern(operations)\n\n# what value is returned by test_cache_system()?",\n    "answer": 5
  }
},
{
  "id": "MF-MIX-V032",
  "metadata": {
    "name": "MultiFunc-Mix-FileSystemOperations",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",

```

```

    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "medium",
    "intervention": 1,
    "variant_type": "filesystem_operations_mix",
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "Side Effect"],
    "primary_focus": "File system operation tracking",
    "analysis_complexity": "medium"
  },
  "task": {
    "description": "Simulate file system operations with error handling and resource tracking. What is the final open_file_count?",
    "code": "typedef struct {\n    char name[50];\n    int size;\n    int\n    is_open;\n} File;\n\nstatic File file_system[10];\nstatic int file_count = 0;\nstatic int\nopen_file_count = 0;\nstatic int operation_count = 0;\n\nint create_file(const char* name,\n    int size) {\n    operation_count++;\n    if (file_count >= 10) {\n        return -1;\n    } // No space\n    // Check for duplicate names\n    for (int i = 0; i < file_count; i++) {\n        if (strcmp(file_system[i].name, name) == 0) {\n            return -1; // Duplicate name\n        }\n    }\n    file_system[file_count].size = size;\n    file_system[file_count].is_open = 0;\n    return file_count++;\n}\n\nint\nopen_file(const char* name) {\n    operation_count++;\n    for (int i = 0; i < file_count; i++) {\n        if (strcmp(file_system[i].name, name) == 0) {\n            if (!file_system[i].is_open) {\n                file_system[i].is_open = 1;\n                open_file_count++;\n                return i;\n            } // Already open\n            return -1; // File not found\n        }\n    }\n    return -1; // File not found\n}\n\nint\nclose_file(const char* name) {\n    operation_count++;\n    for (int i = 0; i < file_count; i++) {\n        if (strcmp(file_system[i].name, name) == 0) {\n            if (file_system[i].is_open) {\n                file_system[i].is_open = 0;\n                open_file_count--;\n                return 0;\n            } // Not open\n            return -1; // File not found\n        }\n    }\n    return -1; // File not found\n}\n\nint\nprocess_file_operations() {\n    // Reset counters\n    file_count = 0;\n    open_file_count = 0;\n    operation_count = 0;\n    // Create files\n    create_file(\"file1.txt\", 100);\n    create_file(\"file2.txt\", 200);\n    create_file(\"file3.txt\", 150);\n    // Open files\n    open_file(\"file1.txt\");\n    open_file(\"file2.txt\");\n    open_file(\"file3.txt\");\n    // Try to open already open file\n    open_file(\"file1.txt\"); // Should fail\n    // Close some files\n    close_file(\"file2.txt\");\n    // Open again\n    open_file(\"file2.txt\");\n    return open_file_count;\n}\n\nint\ntest_filesystem_operations() {\n    return\n    process_file_operations();\n}\n\n// What value is returned by\n// test_filesystem_operations()?",\n    "answer": 3
  }
},
{
  "id": "MF-MIX-V033",
  "metadata": {
    "name": "MultiFunc-Mix-EventDrivenSystem",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
  }
}

```



```

    "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
    "primary_focus": "State machine with side effect tracking",
    "analysis_complexity": "high"
  },
  "task": {
    "description": "Implement state machine processor with transition counting and state validation. What is the final transition_count?",
    "code": "typedef enum {\n    STATE_IDLE = 0, \n    STATE_PROCESSING = 1, \n    STATE_WAITING = 2, \n    STATE_ERROR = 3, \n    STATE_COMPLETE = 4\n} SystemState;\n\nstatic SystemState current_state = STATE_IDLE;\nstatic int transition_count = 0;\nstatic int error_count = 0;\nstatic int processing_value = 0;\n\nint is_valid_transition(SystemState from, SystemState to) {\n    switch (from) {\n        case STATE_IDLE: \n            return (to == STATE_PROCESSING);\n        case STATE_PROCESSING: \n            return (to == STATE_WAITING || to == STATE_ERROR || to == STATE_COMPLETE);\n        case STATE_WAITING: \n            return (to == STATE_PROCESSING || to == STATE_ERROR);\n        case STATE_ERROR: \n            return (to == STATE_IDLE || to == STATE_PROCESSING);\n        case STATE_COMPLETE: \n            return (to == STATE_IDLE);\n        default: \n            return 0;\n    }\n\n    int transition_to_state(SystemState new_state) {\n        if (!is_valid_transition(current_state, new_state)) {\n            error_count++;\n            current_state = STATE_ERROR;\n            return 0;\n        }\n        current_state = new_state;\n        transition_count++;\n        return 1;\n    }\n\n    int process_input(int input) {\n        switch (current_state) {\n            case STATE_IDLE: \n                if (input > 0) {\n                    processing_value = input;\n                    return transition_to_state(STATE_PROCESSING);\n                }\n                break;\n            case STATE_PROCESSING: \n                processing_value += input;\n                if (processing_value > 100) {\n                    return transition_to_state(STATE_COMPLETE);\n                }\n                else if (processing_value < 0) {\n                    return transition_to_state(STATE_ERROR);\n                }\n                break;\n            case STATE_WAITING: \n                if (input == 0) {\n                    return transition_to_state(STATE_PROCESSING);\n                }\n                break;\n            case STATE_ERROR: \n                if (input == 999) { // Reset signal\n                    processing_value = 0;\n                    return transition_to_state(STATE_IDLE);\n                }\n                break;\n            case STATE_COMPLETE: \n                processing_value = 0;\n                return transition_to_state(STATE_IDLE);\n        }\n        default: \n            return 0;\n    }\n\n    return 1;\n}\n\nint test_state_machine() {\n    // Reset state\n    current_state = STATE_IDLE;\n    transition_count = 0;\n    error_count = 0;\n    processing_value = 0;\n\n    // Process input sequence\n    int inputs[] = {10, 20, 0, 30, 50};\n    for (int i = 0; i < 5; i++) {\n        process_input(inputs[i]);\n    }\n\n    return transition_count;\n}\n\n// What value is returned by test_state_machine()?",\n    "answer": 6\n  }
},
{
  "id": "MF-MIX-V035",
  "metadata": {
    "name": "MultiFunc-Mix-ThreadPoolsimulation",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "python",
  }
}

```

```

        "difficulty": "hard",
        "intervention": 2,
        "variant_type": "thread_pool_simulation_mix",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],
        "primary_focus": "Thread pool task scheduling simulation",
        "analysis_complexity": "high"
    },
    "task": {
        "description": "Simulate thread pool with task queuing and completion tracking.  
What is the final completed_tasks count?",
        "code": "class Task:\n    def __init__(self, task_id, work_units, priority=1):\n        self.task_id = task_id\n        self.work_units = work_units\n        self.priority = priority\n        self.completed = False\n\nclass Worker:\n    def __init__(self, worker_id, capacity):\n        self.worker_id = worker_id\n        self.capacity = capacity\n        self.current_load = 0\n        self.completed_tasks = 0\n\ndef can_accept_task(self, task):\n    return self.current_load + task.work_units <= self.capacity\n\ndef execute_task(self, task):\n    if self.can_accept_task(task):\n        self.current_load += task.work_units\n        task.completed = True\n        self.completed_tasks += 1\n    return True\n    return False\n\ndef finish_work(self):\n    # Simulate finishing current work\n    self.current_load = 0\n\nclass ThreadPool:\n    def __init__(self, num_workers, worker_capacity):\n        self.workers = [Worker(i, worker_capacity) for i in range(num_workers)]\n        self.task_queue = []\n        self.completed_tasks = 0\n\ndef submit_task(self, task):\n    self.task_queue.append(task)\n\ndef schedule_tasks(self):\n    # Sort tasks by priority (higher first)\n    self.task_queue.sort(key=lambda t: t.priority, reverse=True)\n    for task in self.task_queue[:]:\n        for worker in self.workers:\n            if worker.execute_task(task):\n                self.task_queue.remove(task)\n                self.completed_tasks += 1\n                break\n\ndef process_work_cycle(self):\n    # Finish current work and schedule new tasks\n    for worker in self.workers:\n        worker.finish_work()\n    self.schedule_tasks()\n    return self.completed_tasks\n\ndef test_thread_pool():\n    pool = ThreadPool(3, 10) # 3 workers, capacity 10 each\n    # Submit various tasks\n    tasks = [\n        Task(1, 5, 3), # High priority, medium work\n        Task(2, 8, 1), # Low priority, high work\n        Task(3, 3, 2), # Medium priority, low work\n        Task(4, 12, 3), # High priority, oversize (won't fit)\n        Task(5, 4, 2), # Medium priority, low work\n        Task(6, 6, 1), # Low priority, medium work\n        Task(7, 2, 3), # High priority, very low work\n        Task(8, 9, 2), # Medium priority, high work\n    ]\n    for task in tasks:\n        pool.submit_task(task)\n    # Process multiple work cycles\n    cycle1 = pool.process_work_cycle()\n    cycle2 = pool.process_work_cycle()\n    return pool.completed_tasks\n\n# What value is returned by test_thread_pool()?",\n        "answer": 6
    }
},
{
    "id": "MF-MIX-V036",
    "metadata": {
        "name": "MultiFunc-Mix-CircuitBreakerPattern",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
    }
}

```

```

        "difficulty": "expert",
        "intervention": 3,
        "variant_type": "circuit_breaker_pattern_mix",
        "mixed_mechanisms": ["Call Chain", "State Propagation", "Side Effect"],
        "primary_focus": "Circuit breaker pattern with failure tracking",
        "analysis_complexity": "extreme"
    },
    "task": {
        "description": "Implement circuit breaker pattern with state transitions and failure counting. What is the final successful_calls count?",
        "code": "from enum import Enum\nimport time\n\nclass CircuitState(Enum):\n    CLOSED = 1\n    OPEN = 2\n    HALF_OPEN = 3\n\nclass CircuitBreaker:\n    def __init__(self, failure_threshold=3, recovery_timeout=5, success_threshold=2):\n        self.failure_threshold = failure_threshold\n        self.recovery_timeout = recovery_timeout\n        self.success_threshold = success_threshold\n\n        self.state = CircuitState.CLOSED\n        self.failure_count = 0\n        self.success_count = 0\n        self.last_failure_time = 0\n        self.total_calls = 0\n        self.successful_calls = 0\n        self.failed_calls = 0\n        self.rejected_calls = 0\n\n    def call(self, operation, *args):\n        self.total_calls += 1\n\n        if self.state == CircuitState.OPEN:\n            if time.time() - self.last_failure_time < self.recovery_timeout:\n                self.rejected_calls += 1\n                raise Exception(\"Circuit breaker is OPEN\")\n            else:\n                self.state = CircuitState.HALF_OPEN\n                self.success_count = 0\n\n                try:\n                    result = operation(*args)\n                    self.on_success()\n                except Exception as e:\n                    self.on_failure()\n                    raise e\n\n        def on_success(self):\n            self.successful_calls += 1\n            self.failure_count = 0\n\n            if self.state == CircuitState.HALF_OPEN:\n                self.success_count += 1\n\n                if self.success_count >= self.success_threshold:\n                    self.state = CircuitState.CLOSED\n\n            def on_failure(self):\n                self.failed_calls += 1\n                self.failure_count += 1\n                self.last_failure_time = time.time()\n\n                if self.failure_count >= self.failure_threshold:\n                    self.state = CircuitState.OPEN\n\n# Simulate time for testing\n\n\ncurrent_test_time = 0\n\n\ndef mock_time():\n    return current_test_time\n\n\ntime.time = mock_time\n\n\ndef unreliable_service(value):\n    # Simulate service that fails for certain inputs\n    if value < 0:\n        raise Exception(\"Negative value error\")\n    if value > 100:\n        raise Exception(\"value too large error\")\n    return value * 2\n\n\ndef test_circuit_breaker():\n    global current_test_time\n    current_test_time = 0\n\n    cb = CircuitBreaker(failure_threshold=2, recovery_timeout=3, success_threshold=2)\n\n    test_inputs = [\n        10,  # Success\n        20,  # Success\n        -5,  # Failure\n        -10, # Failure 2 - Circuit opens\n        30,  # Rejected (circuit open)\n        40,  # Rejected (circuit open)\n    ]\n\n    for i, input_val in enumerate(test_inputs):\n        if i == 4: # Advance time to allow recovery\n            current_test_time += 4\n\n        try:\n            result = cb.call(unreliable_service, input_val)\n        except Exception:\n            pass # Ignore exceptions for testing\n\n        # Continue testing after circuit goes to half-open\n        test_inputs_2 = [\n            25,  # Success 1 in half-open\n            35,  # Success 2 in half-open - Circuit closes\n            45,  # Success in closed\n            -15, # Failure 1 in closed\n            55,  # Success in closed\n        ]\n\n        for input_val in test_inputs_2:\n            try:\n                result = cb.call(unreliable_service, input_val)\n            except Exception:\n                pass\n\n    return cb.successful_calls\n\n# What value is returned by test_circuit_breaker()?",\n    \"answer\": 5\n}

```

```
},
{
  "id": "MF-MIX-V037",
  "metadata": {
    "name": "MultiFunc-Mix-MemoryPoolManager",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "c",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "memory_pool_manager_mix",
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "Side Effect"],
    "primary_focus": "Memory pool management with fragmentation tracking",
    "analysis_complexity": "extreme"
  },
  "task": {
    "description": "Implement memory pool manager with allocation tracking and fragmentation analysis. What is the final allocated_blocks count?"
  }
}
```

```

    "code": "typedef struct Block {\n        int size;\n        int is_allocated;\n\n    struct Block* next;\n} Block;\n\ntypedef struct {\n    Block* head;\n    int total_size;\n    int allocated_blocks;\n    int free_blocks;\n    int fragmentation_score;\n} MemoryPool;\n\nstatic MemoryPool pool;\n\nvoid init_pool(int size) {\n    pool.head =\n        malloc(sizeof(Block));\n    pool.head->size = size;\n    pool.head->is_allocated = 0;\n    pool.head->next = NULL;\n    pool.total_size = size;\n    pool.allocated_blocks = 0;\n    pool.free_blocks = 1;\n    pool.fragmentation_score = 0;\n}\n\nBlock* find_free_block(int size) {\n    Block* current = pool.head;\n    while (current != NULL) {\n        if (!current->is_allocated && current->size >= size) {\n            return current;\n        }\n        current = current->next;\n    }\n    return NULL;\n}\n\nvoid\nsplit_block(Block* block, int size) {\n    if (block->size > size) {\n        Block* new_block =\n            malloc(sizeof(Block));\n        new_block->size = block->size - size;\n        new_block->is_allocated = 0;\n        new_block->next = block->next;\n        block->size = size;\n        block->next = new_block;\n        pool.fragmentation_score++;\n    }\n}\n\nint\nallocate_memory(int size) {\n    Block* block = find_free_block(size);\n    if (block == NULL) {\n        return 0; // Allocation failed\n    }\n    block->is_allocated = 1;\n    pool.allocated_blocks++;\n    pool.free_blocks--;\n    return 1; // Allocation successful\n}\n\nint\ndeallocate_memory(int size) {\n    Block* current = pool.head;\n    while (current != NULL) {\n        if (current->is_allocated && current->size == size) {\n            current->is_allocated = 0;\n            pool.allocated_blocks--;\n            pool.free_blocks++;\n\n            // Try to merge with next block\n            if (current->next &&\n                !current->next->is_allocated) {\n                Block* next = current->next;\n                current->size += next->size;\n                current->next = next->next;\n                free(next);\n                pool.free_blocks--;\n                pool.fragmentation_score =\n                    pool.fragmentation_score > 0 ?\n                        pool.fragmentation_score - 1 : 0;\n            }\n            return 1;\n        }\n        current = current->next;\n    }\n    return 0; // Deallocation failed\n}\n\nint\nprocess_allocation_sequence() {\n    init_pool(1000);\n    int allocations[] = {100, 200, 150, 50, 300, 75, 125};\n    int deallocations[] = {200, 50, 100};\n\n    // Perform allocations\n    for (int i = 0; i < 7; i++) {\n        allocate_memory(allocations[i]);\n    }\n\n    // Perform some deallocations\n    for (int i = 0; i < 3; i++) {\n        deallocate_memory(deallocations[i]);\n    }\n\n    // Try more allocations\n    allocate_memory(80);\n    allocate_memory(120);\n\n    return\n    pool.allocated_blocks;\n}\n\nint\ntest_memory_pool() {\n    return\n    process_allocation_sequence();\n}\n\n// What value is returned by test_memory_pool()?",\n    "answer": 6\n}\n},\n{\n    "id": "MF-MIX-V038",\n    "metadata": {\n        "name": "MultiFunc-Mix-AsyncTaskProcessor",\n        "category": "MultiFunc-Level",\n        "subcategory": "Mix",\n        "type": "variant",\n        "source": "Generated",\n        "language": "python",\n        "difficulty": "expert",\n        "intervention": 3,\n        "variant_type": "async_task_processor_mix",\n        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation"],\n    }\n}

```

```
  "primary_focus": "Asynchronous task processing simulation",
  "analysis_complexity": "extreme"
},
"task": {
  "description": "Simulate async task processing with dependencies and completion tracking. What is the final completed_tasks count?",
```

```

    "code": "from collections import deque\nfrom enum import Enum\n\nclass TaskState(Enum):\n    PENDING = 1\n    RUNNING = 2\n    COMPLETED = 3\n    FAILED = 4\n\nclass Task:\n    def __init__(self, task_id, processing_time, dependencies=None):\n        self.task_id = task_id\n        self.processing_time = processing_time\n        self.dependencies = dependencies or []\n        self.state = TaskState.PENDING\n        self.result = None\n        self.start_time = None\n        def can_start(self, completed_tasks):\n            return all(dep_id in completed_tasks for dep_id in\nself.dependencies)\n        def process(self, current_time):\n            if self.state ==\n                TaskState.PENDING:\n                    self.state = TaskState.RUNNING\n                    self.start_time = current_time\n                elif self.state == TaskState.RUNNING:\n                    if current_time -\nself.start_time >= self.processing_time:\n                        self.state = TaskState.COMPLETED\n                        self.result = self.task_id * 10 # Simple result calculation\n            return True\n        return False\n\nclass AsyncProcessor:\n    def __init__(self, max_concurrent=3):\n        self.max_concurrent = max_concurrent\n        self.task_queue = deque()\n        self.running_tasks = []\n        self.completed_tasks = set()\n        self.failed_tasks = set()\n        self.current_time = 0\n        self.total_completed = 0\n\n    def submit_task(self, task):\n        self.task_queue.append(task)\n\n    def can_schedule_task(self, task):\n        return (len(self.running_tasks) <\nself.max_concurrent and\n\n    def schedule_ready_tasks(self):\n        scheduled = []\n        remaining_queue = deque()\n\n        while self.task_queue:\n            task = self.task_queue.popleft()\n            if self.can_schedule_task(task):\n                scheduled.append(task)\n                if len(self.running_tasks) >=\nself.max_concurrent:\n                    # Put remaining tasks back\n                    remaining_queue.extend(self.task_queue)\n                    break\n            else:\n                remaining_queue.append(task)\n\n        self.task_queue = remaining_queue\n        return len(scheduled)\n\n    def process_running_tasks(self):\n        completed_this_cycle = []\n        for task in self.running_tasks[:]:\n            if task.process(self.current_time):\n                if task.state == TaskState.COMPLETED:\n                    self.completed_tasks.add(task.task_id)\n                    completed_this_cycle.append(task)\n                    self.total_completed += 1\n                elif task.state == TaskState.FAILED:\n                    self.failed_tasks.add(task.task_id)\n                    completed_this_cycle.append(task)\n\n                # Remove completed tasks from running list\n                for task in\ncompleted_this_cycle:\n                    self.running_tasks.remove(task)\n\n        return len(completed_this_cycle)\n\n    def run_simulation(self, max_time=100):\n        while\n(self.task_queue or self.running_tasks) and self.current_time < max_time:\n            self.schedule_ready_tasks()\n            self.process_running_tasks()\n            self.current_time += 1\n\n        return self.total_completed\n\n    def test_async_processor():\n        processor = AsyncProcessor(max_concurrent=2)\n\n        # Create tasks with dependencies\n        tasks = [\n            Task(1, 3, []),\n            # No dependencies\n            Task(2, 2, []),\n            # No dependencies\n            Task(3, 4, [1]),\n            # Depends on task 1\n            Task(4, 1, [2]),\n            # Depends on task 2\n            Task(5, 3, [1, 2]),\n            # Depends on tasks 1 and 2\n            Task(6, 2, [3, 4]),\n            # Depends on tasks 3 and 4\n            Task(7, 1, [5]),\n            # Depends on task 5\n            Task(8, 2, [6, 7]),\n            # Depends on tasks 6 and 7\n        ]\n\n        # Submit all tasks\n        for task in tasks:\n            processor.submit_task(task)\n\n        # Run simulation\n        return processor.run_simulation()\n\n    # What value is returned by test_async_processor()?\n    \"answer\": 8\n\n}\n\n{\n
```

```
"id": "MF-MIX-V039",
"metadata": {
    "name": "MultiFunc-Mix-LoadBalancersimulation",
    "category": "MultiFunc-Level",
    "subcategory": "Mix",
    "type": "variant",
    "source": "Generated",
    "language": "python",
    "difficulty": "expert",
    "intervention": 3,
    "variant_type": "load_balancer_simulation_mix",
    "mixed_mechanisms": ["Call chain", "State Propagation", "Side Effect"],
    "primary_focus": "Load balancer with health checking and routing",
    "analysis_complexity": "extreme"
},
"task": {
    "description": "Simulate load balancer with server health monitoring and request routing. What is the final successful_requests count?"
},
```

```

    "code": "import random\nrandom.seed(42) # For reproducible results\n\nclass\nServer:\n    def __init__(self, server_id, capacity, failure_rate=0.1):\n        self.server_id = server_id\n        self.capacity = capacity\n        self.current_load = 0\n        self.is_healthy = True\n        self.failure_rate = failure_rate\n        self.total_requests = 0\n        self.successful_requests = 0\n        self.failed_requests = 0\n    def can_handle_request(self):\n        return self.is_healthy and self.current_load < self.capacity\n    def process_request(self, request_size):\n        if not self.can_handle_request():\n            return False\n        self.total_requests += 1\n        self.current_load += request_size\n        # Simulate request processing with potential failure\n        if random.random() < self.failure_rate:\n            self.failed_requests += 1\n            # High failure rate might affect server health\n            if self.failed_requests > 3:\n                self.is_healthy = False\n                return False\n            else:\n                self.successful_requests += 1\n                return True\n        \n        def complete_request(self, request_size):\n            self.current_load = max(0, self.current_load - request_size)\n\n    def health_check(self):\n        # Server recovers if load is low and few recent failures\n        if not self.is_healthy and self.current_load == 0:\n            if self.failed_requests <= 2:\n                self.is_healthy = True\n                return self.is_healthy\n\nclass LoadBalancer:\n    def __init__(self, servers):\n        self.servers = servers\n        self.total_requests = 0\n        self.successful_requests = 0\n        self.failed_requests = 0\n        self.current_server_index = 0\n    def get_next_server_round_robin(self):\n        # Round-robin with health checking\n        attempts = 0\n        while attempts < len(self.servers):\n            server = self.servers[self.current_server_index]\n            self.current_server_index = (self.current_server_index + 1) % len(self.servers)\n            if server.health_check() and server.can_handle_request():\n                return server\n            attempts += 1\n        return None # No available server\n    def route_request(self, request_size):\n        self.total_requests += 1\n        server = self.get_next_server_round_robin()\n        if server is None:\n            self.failed_requests += 1\n            return False\n        success = server.process_request(request_size)\n        if success:\n            self.successful_requests += 1\n            # Simulate request completion after processing\n            server.complete_request(request_size)\n        else:\n            self.failed_requests += 1\n        return success\n    def process_request_batch(self, requests):\n        for request_size in requests:\n            self.route_request(request_size)\n        return self.successful_requests\n    def run_health_checks(self):\n        healthy_count = 0\n        for server in self.servers:\n            if server.health_check():\n                healthy_count += 1\n        return healthy_count\n\ndef test_load_balancer():\n    # Create servers with different capacities and failure rates\n    servers = [\n        Server(1, 10, 0.1), # High capacity, low failure rate\n        Server(2, 8, 0.15), # Medium capacity, medium failure rate\n        Server(3, 6, 0.05), # Lower capacity, very low failure rate\n        Server(4, 12, 0.2), # Highest capacity, high failure rate\n    ]\n    lb = LoadBalancer(servers)\n\n    # Simulate request patterns\n    request_batches = [\n        [2, 3, 1, 4, 2], # Light load\n        [5, 6, 4, 7, 3, 2], # Medium load\n        [8, 9, 6, 5, 7], # Heavy load\n        [3, 2, 1, 4, 2, 3], # Mixed load\n        [10, 11, 9, 8], # very heavy load (some failures expected)\n    ]\n    for batch in request_batches:\n        lb.process_request_batch(batch)\n        lb.run_health_checks() # Periodic health checking\n\n    return lb.successful_requests\n\n# What value is returned by test_load_balancer()?",\n    "answer": 18\n}\n",

```

```
{  
  "id": "MF-MIX-V040",  
  "metadata": {  
    "name": "MultiFunc-Mix-DistributedConsensus",  
    "category": "MultiFunc-Level",  
    "subcategory": "Mix",  
    "type": "variant",  
    "source": "Generated",  
    "language": "python",  
    "difficulty": "master",  
    "intervention": 4,  
    "variant_type": "distributed_consensus_mix",  
    "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation",  
"Side Effect"],  
    "primary_focus": "Distributed consensus algorithm simulation",  
    "analysis_complexity": "ultimate"  
  },  
  "task": {  
    "description": "Simulate simplified distributed consensus algorithm with leader  
election and state replication. What is the final committed_entries count?",  
  }  
}
```

```

    "code": "import random\nrandom.seed(42)\n\nclass LogEntry:\n    def\n__init__(self, term, index, data):\n        self.term = term\n        self.index = index\n        self.data = data\n        self.committed = False\n\nclass Node:\n    def __init__(self, node_id, cluster_size):\n        self.node_id = node_id\n        self.cluster_size = cluster_size\n        self.current_term = 0\n        self.voted_for = None\n        self.log = []\n        self.commit_index = 0\n        self.is_leader = False\n        self.vote_count = 0\n        self.heartbeat_responses = 0\n    \n    def start_election(self):\n        self.current_term += 1\n        self.voted_for = self.node_id\n        self.vote_count = 1\n    # Vote for self\n        self.is_leader = False\n        return self.current_term\n    \n    def request_vote(self, candidate_term, candidate_id):\n        if candidate_term > self.current_term:\n            self.current_term = candidate_term\n        self.voted_for = None\n        if (self.voted_for is None or self.voted_for == candidate_id) and candidate_term >= self.current_term:\n            self.voted_for = candidate_id\n            return True\n        return False\n    \n    def receive_vote(self):\n        self.vote_count += 1\n        if self.vote_count > self.cluster_size // 2:\n            self.is_leader = True\n            return True\n        return False\n    \n    def append_entry(self, term, data):\n        if self.is_leader:\n            entry = LogEntry(term, len(self.log), data)\n            self.log.append(entry)\n            return True\n        return False\n    \n    def replicate_entry(self, entry):\n        # Follower receives entry from leader\n        if len(self.log) == entry.index:\n            self.log.append(entry)\n            return True\n        return False\n    \n    def commit_entries(self, leader_commit_index):\n        old_commit_index = self.commit_index\n        self.commit_index = min(leader_commit_index, len(self.log) - 1)\n        \n        committed_count = 0\n        for i in range(old_commit_index + 1, self.commit_index + 1):\n            if i < len(self.log):\n                self.log[i].committed = True\n        committed_count += 1\n        \n        return committed_count\n\nclass\nDistributedSystem:\n    def __init__(self, num_nodes):\n        self.nodes = [Node(i, num_nodes) for i in range(num_nodes)]\n        self.num_nodes = num_nodes\n        self.current_leader = None\n        self.total_committed_entries = 0\n    \n    def simulate_election(self):\n        # Random node starts election\n        candidate_id = random.randint(0, self.num_nodes - 1)\n        candidate = self.nodes[candidate_id]\n        \n        term = candidate.start_election()\n        \n        # Other nodes vote\n        for node in self.nodes:\n            if node.node_id != candidate_id:\n                if node.request_vote(term, candidate_id):\n                    candidate.receive_vote()\n                self.current_leader = candidate_id\n                for node in self.nodes:\n                    node.is_leader = False\n            if node.node_id != candidate_id:\n                node.current_term = term\n                return True\n            return False\n    \n    def replicate_log_entry(self, data):\n        if self.current_leader is None:\n            return False\n            \n            leader = self.nodes[self.current_leader]\n            \n            # Leader appends entry\n            if not leader.append_entry(leader.current_term, data):\n                return False\n                \n                entry = leader.log[-1]\n                replication_count = 1 # Leader has the entry\n                \n                for node in self.nodes:\n                    if node.node_id != self.current_leader:\n                        if node.replicate_entry(entry):\n                            replication_count += 1\n                \n                # Commit if majority has the entry\n                if replication_count > self.num_nodes // 2:\n                    # Update commit index for all nodes\n                    for node in self.nodes:\n                        committed = node.commit_entries(entry.index)\n                        \n                        if node.node_id == self.current_leader:\n                            self.total_committed_entries += committed\n                        \n                        return True\n                    \n                    return False\n            \n            def\nsimulate_consensus_operations(self):\n                operations = [\n                    \"SET key1 value1\", \n                    \"SET key2 value2\", \n                    \"SET key3 value3\", \n                    \"SET key4 value4\", \n                    \"DEL key1\", \n                    \"SET key2\", \n                    \"DEL key2\", \n                    \"DEL key3\", \n                    \"SET key4\", \n                    \"DEL key4\"]

```

```

\"SET key5 value5\",\n        ]\n        \n        # Start with an election\n
election_success = False\n        attempts = 0\n        while not election_success and
attempts < 3:\n            election_success = self.simulate_election()\n            attempts
+= 1\n            \n            if not election_success:\n                return 0\n            \n            #
Process operations\n            successful_operations = 0\n            for operation in
operations:\n                if self.replicate_log_entry(operation):\n
successful_operations += 1\n                \n                # Simulate occasional leader failure\n
                if random.random() < 0.15: # 15% chance of leader failure\n
self.current_leader = None\n                \n                # Try to elect new leader\n
self.simulate_election()\n            \n            return self.total_committed_entries\n\n
def test_distributed_consensus():\n    system = DistributedSystem(5) # 5-node cluster\n
return system.simulate_consensus_operations()\n\n# What value is returned by
test_distributed_consensus()?",\n
    "answer": 5
}
},
{
    "id": "MF-MIX-V041",
    "metadata": {
        "name": "MultiFunc-Mix-StreamProcessingPipeline",
        "category": "MultiFunc-Level",
        "subcategory": "Mix",
        "type": "variant",
        "source": "Generated",
        "language": "python",
        "difficulty": "master",
        "intervention": 4,
        "variant_type": "stream_processing_pipeline_mix",
        "mixed_mechanisms": ["Call Chain", "Parameter Passing", "State Propagation",
"Side Effect"],
        "primary_focus": "Stream processing pipeline with windowing and aggregation",
        "analysis_complexity": "ultimate"
    },
    "task": {
        "description": "Implement stream processing pipeline with windowed aggregations
and watermark handling. What is the final processed_windows count?"
    }
}

```

```

    "code": "from collections import defaultdict, deque\nimport heapq\n\nclass
StreamEvent:\n    def __init__(self, timestamp, key, value, event_type='data'):\n        self.timestamp = timestamp\n        self.key = key\n        self.value = value\n        self.event_type = event_type\n\nclass Window:\n    def __init__(self, start_time, end_time):\n        self.start_time = start_time\n        self.end_time = end_time\n        self.events = []\n        self.aggregated_value = 0\n        self.is_closed = False\n\n    def add_event(self, event):\n        if self.start_time <= event.timestamp <
self.end_time:\n            self.events.append(event)\n            return True\n        return False\n\n    def compute_aggregation(self):\n        if not self.is_closed:\n            self.aggregated_value = sum(event.value for event in self.events)\n        self.is_closed = True\n        return self.aggregated_value\n\nclass StreamProcessor:\n    def __init__(self, window_size=10, watermark_delay=5):\n        self.window_size =
window_size\n        self.watermark_delay = watermark_delay\n        self.windows =
defaultdict(dict) # key -> {window_start: Window}\n        self.current_watermark = 0\n        self.event_buffer = [] # Min-heap for out-of-order events\n        self.processed_windows = 0\n        self.total_events_processed = 0\n        self.late_events_dropped = 0\n\n    def create_window(self, key, event_timestamp):\n        window_start = (event_timestamp // self.window_size) * self.window_size\n        window_end = window_start + self.window_size\n        if window_start not in self.windows[key]:\n            self.windows[key][window_start] = Window(window_start,
window_end)\n        return self.windows[key][window_start]\n\n    def update_watermark(self, event_timestamp):\n        # watermark is the maximum timestamp minus
delay\n        new_watermark = event_timestamp - self.watermark_delay\n        if
new_watermark > self.current_watermark:\n            self.current_watermark =
new_watermark\n            self.close_expired_windows()\n\n    def close_expired_windows(self):\n        for key in list(self.windows.keys()):\n            for window_start in list(self.windows[key].keys()):\n                window = self.windows[key][window_start]\n                # Close windows that end before the
watermark\n                if window.end_time <= self.current_watermark and not
window.is_closed:\n                    window.compute_aggregation()\n\n    self.processed_windows += 1\n\n    def process_event(self, event):\n        # Check if
event is too late (before watermark)\n        if event.timestamp < self.current_watermark:\n            self.late_events_dropped += 1\n            return False\n        # Add
to appropriate window\n        window = self.create_window(event.key, event.timestamp)\n        success = window.add_event(event)\n        if success:\n            self.total_events_processed += 1\n            # Update watermark based on this event\n            self.update_watermark(event.timestamp)\n            return success\n\n    def process_event_stream(self, events):\n        # Sort events by timestamp to simulate ordered
processing\n        # But introduce some out-of-order events\n        processed_events =
[]\n\n        for i, event in enumerate(events):\n            # Occasionally delay
events to simulate out-of-order arrival\n            if i > 0 and i % 7 == 0: # Every 7th
event arrives late\n                # Insert delayed event into buffer\n                heapq.heappush(self.event_buffer, (event.timestamp, event))\n            else:\n                processed_events.append(event)\n\n        events that should be processed now\n        while (self.event_buffer and
self.event_buffer[0][1].timestamp <= event.timestamp +
self.watermark_delay):\n            _, buffered_event =
heapq.heappop(self.event_buffer)\n            self.process_event(buffered_event)\n\n            # Process current event\n            self.process_event(event)\n            # Process remaining buffered events\n        while self.event_buffer:\n            _, buffered_event = heapq.heappop(self.event_buffer)\n            self.process_event(buffered_event)\n            # Force close any remaining
windows\n            self.current_watermark = float('inf')\n

```

```
self.close_expired_windows()\nget_aggregated_results(self):\n    results[key] = []\n    sorted(self.windows[key].keys()):\n        [window_start]\n            if window.is_closed:\n                results[key].append({\n                    'end': window.end_time,\n                    'start': window.start_time,\n                    'value': window.aggregated_value\n                })\n            return results\n\n@dataclass\nclass StreamEvent:\n    timestamp: int\n    user_id: str\n    value: int\n\nprocessor = StreamProcessor(window_size=10, watermark_delay=3)\n\n# Create stream of events with different keys and timestamps\nevents = [\n    StreamEvent(5, 'user1', 10),\n    StreamEvent(7, 'user2', 15),\n    StreamEvent(15, 'user2', 25),\n    StreamEvent(22, 'user2', 35),\n    StreamEvent(28, 'user2', 45),\n    StreamEvent(35, 'user2', 55),\n    StreamEvent(42, 'user2', 65),\n    StreamEvent(48, 'user2', 75),\n    StreamEvent(52, 'user1', 80),\n]\n\nresult = processor.process_event_stream(events)\n\n# What value is returned by test_stream_processing()?",\n    "answer": 10\n}\n}\n]
```