编译原理第三次实验测试用例：目录

1 A 组测试用例

本组测试用例共 5 个，均为比较简单的程序，简单检查针对赋值/算术语句、分支语句、循环语句、数组表达式和函数调用的翻译。

1.1 A-1

输入

```
int main() {
  int result = 0;
  int i = 9, j = 32, k = 57;
  i = i * (i * j - k);
  write(i);
  j = i - (j + k * (k - 1)) * (-1);
  write(j);
  k = j - (k + k * j + 1 * (i + j + k)) / (i / 2);
  write(k);
  result = i + j - k;
  write(result);
  return 0;
}
```

程序输入: 无; 预期输出: 2079 5303 5005 2377

说明：这个测试用例针对赋值与算术语句进行测试。注意，预期输入/输出中每个数字会占一行，这里为了节省空间写在同一行，以空格隔开 (下同)。

1.2 A-2

输入

```
int main() {
  int x;
  int y;
  int z;
  x = read();
  y = read();
```

```
  z = read();
  if (x > 0 && y > 0) {
    write(1);
  } else {
    if (x == 0 && y == 0) {
      write(2);
    } else {
      if (x < 0 && y < 0) {
        write(3);
      } else {
        if (x >= y) {
          write(4);
        } else {
          if (x < z && z < y) {
            write(5);
          } else {
            write(6);
          }
        }
      }
    }
  }
  return 0;
}
```

程序输入: 1 2 3; 预期输出: 1

程序输入: 0 0 2; 预期输出: 2

程序输入: -3 2 1; 预期输出: 5

程序输入: -3 4 5; 预期输出: 6

说明：主要针对分支语句进行测试。

1.3 A-3

输入

```
int main() {
  int i;
  int j;
  int o_cnt = 0;
  int i_cnt = 0;
  int len = 10;
  int arr[10];
  int result = 0;

  i_cnt = 0;
  while (i_cnt < len) {
    arr[i_cnt] = 0;
    i_cnt = i_cnt + 1;
  }

  o_cnt = 0;
  while (o_cnt < 3) {
    i = read();
    j = read();
    if (i > 10) {
      i = 1 + i - i / 10 * 10;
    }
    if (j > 10) {
      j = 2 + j - j / 10 * 10;
    }

    i_cnt = 0;
    while (i_cnt < len) {
      arr[i_cnt] = arr[i_cnt] / 2 + i - j;
      i_cnt = i_cnt + 1;
    }
    o_cnt = o_cnt + 1;
```

```
  }

  i_cnt = 0;
  while (i_cnt < len) {
    result = result + arr[i_cnt];
    i_cnt = i_cnt + 1;
  }
  write(result);
  return 0;
}
```

程序输入: 1 2 3 4 5 6; 预期输出: -20

程序输入: 1 3 5 7 9 11; 预期输出: 40

程序输入: -1 2 -3 4 -5 6; 预期输出: -160

程序输入: 1 1 2 3 5 8; 预期输出: -40

说明：主要测试循环语句。

1.4 A-4

输入

```
int main() {
  int i;
  int j;
  int cnt;
  int len = 10;
  int arr1[10];
  int arr2[10];
  int result = 0;

  i = read();
  cnt = 0;
  while (cnt < len) {
    arr1[cnt] = i + cnt;
    cnt = cnt + 1;
```

```
  }

  arr2[0] = 2;
  cnt = 1;
  while (cnt < len) {
    int tmp = arr1[cnt] - arr1[cnt] / 3 * 3;
    if (tmp == 0) {
      arr2[cnt] = arr1[cnt];
    } else {
      arr2[cnt] = - arr1[cnt];
    }
    cnt = cnt + 1;
  }

  cnt = 0;
  while (cnt < len) {
    result = result + arr2[cnt];
    cnt = cnt + 1;
  }
  write(result);
  return 0;
}
```

程序输入: 10; 预期输出: -43

程序输入: 100; 预期输出: -313

程序输入: 45; 预期输出: -142

程序输入: 90; 预期输出: -277

说明：主要测试一维数组。

1.5 A-5

输入

```
int cal_sum(int s1, int s2, int s3) {
  int s_res = s1 + s2 + s3;
```

```
  return s_res;
}

int mod(int m1, int m2) {
  int m_res = m1 - m1 / m2 * m2;
  return m_res;
}

int is_good(int x) {
  int a = x * 2;
  int b = x + 2;
  int c = x * x;
  int sum = cal_sum(a, b, c);
  int rem = mod(sum, 5);
  if (rem > 1) {
    return 1;
  } else {
    return 0;
  }
}

int main() {
  int f;
  int g;
  int h;
  f = read();
  g = read();
  h = read();
  if (is_good(f)) {
    write(cal_sum(f, g, h));
  } else {
    write(0);
```

```
35    }
36    return 0;
37  }
```

程序输入: 2 2 2; 预期输出: 6

程序输入: 1 2 3; 预期输出: 0

说明：一个测试函数调用的小程序。

2 B 组测试用例

本组测试用例共 3 个，较 A 组测试用例复杂，这里不专门针对赋值和算术语句设计测试用例。

2.1 B-1

输入

```
1  int mod(int m1, int m2) {
2    int m_res = m1 - m1 / m2 * m2;
3    return m_res;
4  }
5
6  int is_palindrome_number(int x) {
7    int div = 1;
8    if (x < 0) {
9      return 0;
10   }
11
12   while ((x / div) >= 10) {
13     div = div * 10;
14   }
15
16   while (x) {
17     if ((x / div) != mod(x, 10)) {
18       return 0;
```

```
    }
    x = (mod(x, div)) / 10;
    div = div / 100;
  }
  return 1;
}

int main() {
  int i;
  i = read();
  if (is_palindrome_number(i)) {
    write(1);
  } else {
    write(0);
  }
  return 0;
}
```

程序输入: 1888988881; 预期输出: 0

程序输入: 13344331; 预期输出: 1

说明: 一个判断回文数的程序。

2.2 B-2

输入

```
int main() {
  int n1;
  int n2;
  int a1[10];
  int a2[10];
  int a[20];
  int i = 0, j = 0, k = 0, n = 0;
  int cnt = 0;

```

```
10  n1 = read();
11  if (n1 > 10) {
12    n1 = 10;
13  }
14  while (cnt < n1) {
15    a1[cnt] = read();
16    cnt = cnt + 1;
17  }
18
19  n2 = read();
20  if (n2 > 10) {
21    n2 = 10;
22  }
23  cnt = 0;
24  while (cnt < n2) {
25    a2[cnt] = read();
26    cnt = cnt + 1;
27  }
28
29  while (i < n1 && j < n2) {
30    if (a1[i] < a2[j]) {
31      a[k] = a1[i];
32      k = k + 1;
33      i = i + 1;
34    } else {
35      a[k] = a2[j];
36      k = k + 1;
37      j = j + 1;
38    }
39  }
40
41  while (i < n1) {
```

```
    a[k] = a1[i];
    k = k + 1;
    i = i + 1;
  }
  while (j < n2) {
    a[k] = a2[j];
    k = k + 1;
    j = j + 1;
  }

  if ((n1 + n2 == 0) || (n1 + n2 == 1)) {
    write(a[n1 + n2 - 1]);
  } else if (n1 + n2 == 2) {
    write((a[n1 + n2 - 1] + a[n1 + n2 - 2]) / 2);
  } else {
    n = n1 + n2;
    if ((n - (n / 2) * 2) == 0) {
      write((a[n / 2] + a[n / 2 - 1]) / 2);
    } else {
      write(a[n / 2]);
    }
  }
  return 0;
}
```

程序输入: 3 1 4 5 4 4 8 9 10 预期输出: 5

程序输入: 5 -1 2 4 29 100 3 -10 3 10 预期输出: 3

说明：一个计算两个排序好的数组合并后的中位数的程序。

2.3 B-3

输入

```
int main() {
  int idx, s, e;
```

```
3   int tmp1, tmp2;
4   int len = 5;
5   int a[5];
6   int cnt = 0;
7   int stop = 0;
8   while (cnt < len) {
9     a[cnt] = read();
10    cnt = cnt + 1;
11  }
12
13  cnt = len / 2 - 1;
14  while (cnt >= 0) {
15    s = cnt;
16    e = len;
17    tmp1 = a[s];
18    stop = 0;
19    while ((s * 2 + 1 < e) && !stop) {
20      idx = s * 2 + 1;
21      if ((idx + 1 < e) && a[idx + 1] > a[idx]) {
22        idx = idx + 1;
23      }
24      if (a[idx] > tmp1) {
25        a[s] = a[idx];
26        s = idx;
27      } else {
28        stop = 1;
29      }
30    }
31    a[s] = tmp1;
32    cnt = cnt - 1;
33  }
34
```

```
35    cnt = len - 1;
36    while (cnt >= 0) {
37      tmp2 = a[0];
38      a[0] = a[cnt];
39      a[cnt] = tmp2;
40      s = 0;
41      e = cnt;
42      tmp1 = a[s];
43      stop = 0;
44      while ((s * 2 + 1 < e) && !stop) {
45        idx = s * 2 + 1;
46        if ((idx + 1 < e) && a[idx + 1] > a[idx]) {
47          idx = idx + 1;
48        }
49        if (a[idx] > tmp1) {
50          a[s] = a[idx];
51          s = idx;
52        } else {
53          stop = 1;
54        }
55      }
56      a[s] = tmp1;
57      cnt = cnt - 1;
58    }
59
60    cnt = 0;
61    while (cnt < len) {
62      write(a[cnt]);
63      cnt = cnt + 1;
64    }
65    return 0;
66  }
```

程序输入: 5 4 3 2 1 预期输出: 1 2 3 4 5

程序输入: 10 -3 29 100 2 预期输出: -3 2 10 29 100

说明：堆排序。

3 C 组测试用例

本组测试用例共 2 个，是较经典的问题。

3.1 C-1

输入

```
int cal_mid(int c1, int c2) {
    return (c1 + c2) / 2;
}

int main() {
  int len = 5;
  int a[5];
  int cnt = 0;
  int i = 0, j = 0, tmp = 0;
  int low, high;
  int key;
  while (cnt < len) {
    a[cnt] = read();
    cnt = cnt + 1;
  }

  i = 0;
  while (i < len - 1) {
    j = 0;
    while (j < len - 1) {
      if (a[j] > a[j + 1]) {
        tmp = a[j];
        a[j] = a[j + 1];
```

```
        a[j + 1] = tmp;
      }
      j = j + 1;
    }
    i = i + 1;
  }

  i = 0;
  while (i < len) {
    write(a[i]);
    i = i + 1;
  }

  key = read();

  low = 0;
  high = len - 1;
  while (low <= high) {
    int mid = cal_mid(low, high);
    if (a[mid] == key) {
      write(1);
      return 0;
    } else if (a[mid] < key) {
      low = mid + 1;
    } else {
      high = mid - 1;
    }
  }
  write(0);
  return 0;
}
```

程序输入: 1 100 20 10 2 10 预期输出: 1 2 10 20 100 1

程序输入: 7 20 -1 23 3 4 预期输出: -1 3 7 20 23 0

说明：冒泡排序和二分查找。

3.2 C-2

输入

```
int mod(int m1, int m2) {
  int m_res = m1 - m1 / m2 * m2;
  return m_res;
}

int is_prime(int n) {
  int i = 0;
  if (n < 2) {
    return 0;
  }
  if (n == 2) {
    return 1;
  }
  if (mod(n, 2) == 0) {
    return 0;
  }

  i = 3;
  while ((i * i) <= n) {
    if (mod(n, i) == 0) {
        return 0;
    }
    i = i + 1;
  }
  return 1;
}

```

```
int fabonacci(int f) {
  if (f == 0) {
    return 0;
  } else if (f == 1) {
    return 1;
  }
  return fabonacci(f - 1) + fabonacci(f - 2);
}

int main() {
  int idx = 0;
  while (idx < fabonacci(5)) {
    if(is_prime(idx)) {
      write(idx);
    }
    idx = idx + 1;
  }
  return 0;
}
```

程序输入: 无; 预期输出: 2 3

说明：fabonacci 数列和素数判断。

4 D 组测试用例

本组测试用例共 1 个，主要用于测试中间代码的优化。

4.1 D-1

输入

```
int mod(int a, int b) {
  return a - (a / b) * b;
}

```

```
int do_work(int n) {
  int x1, x2, x3;
  n = mod(n, 5);
  x1 = 2 * n;
  x2 = n + n;
  x3 = 4 * 4;
  x1 = x3 + x2 + x1;
  x2 = x1 + (x1 + 1) * (x2 + 1) * (x3 + 1);
  x3 = x1 * (x1 + 1) + (x2 + 1) * (x2 + 1) + (x3 + 1) * (x3 + 1);
  x1 = (x1 + 1) + (x2 + 1) + (x3 + 1);
  return x1 + x2 + x3;
}

int main() {
  int t1 = 2;
  int t2 = 5;
  int t3 = 10;
  int i = 3 * (t2 * t3) - 100 / 5;
  int j = 42 - (t1 * t2) * (t1 * (t2 * t3) / 32) + 100;
  int k = 3 * 4 * 5 - 10 - (-(t1 * t2) * 3) - (t1 * t2) - (t1 * t2) +
      3 + 2 + 1;
  int cnt = 0;
  int sum = 0;
  int len = t1 * t2;
  int array[10];
  int brray[10];
  while (cnt < k) {
    sum = sum + 1;
    array[mod(cnt, len)] = i;
    i = i + 1;
    cnt = cnt + 1;
  }
```

```
  cnt = 0;
  while (cnt < 100) {
    brray[mod(cnt, len)] = i;
    i = i + 1;
    cnt = cnt + 1;
  }

  cnt = 0;
  while (cnt < mod(do_work(10), 30)) {
    j = array[mod(cnt, len)];
    sum = sum + (-1) * cnt + do_work(j);
    j = j + 2 * cnt;
    j = j + 2 * cnt;
    j = j + 2 * cnt;
    if (mod(do_work(10), 10) == mod(do_work(j), 10)) {
      j = mod(j, 10);
    } else {
      j = mod(j, 20);
    }
    array[mod(cnt, len)] = j * j;
    cnt = cnt + 1;
  }
  j = j + array[0] + array[1];
  j = j + array[0] + array[1];
  j = j + array[0] + array[1];
  j = j + array[0] + array[1];
  write(j);
  write(array[0]);
  write(array[1]);
  write(array[2]);
  write(array[3]);
  write(array[4]);
```

```
  return 0;
}
```

程序输入: 无; 预期输出: 500 0 121 256 361 64

说明：用于效率测试。

5 E 组测试用例

本组测试用例共 6 个，针对不同分组进行测试。

E1 组针对 3.1 分组测试结构体的翻译，E2 组针对 3.2 分组测试一维数组作为参数和高维数组的翻译。每组 3 个测试用例。

5.1 E1-1

输入

```
struct Animal {
  int weight;
  int height;
  int index;
};

int main() {
  struct Animal a, b;
  a.weight = 20;
  a.height = 30;
  b.weight = 25;
  b.height = 15;
  a.index = a.weight * a.weight / (a.height * a.height);
  b.index = b.weight * b.weight / (b.height * b.height);
  write(a.index + b.index);
  return 0;
}
```

程序输入: 无; 预期输出: 2

说明：测试对于简单结构体的翻译，不涉及与数组的交互和结构体作为函数参数调用。针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

5.2 E1-2

输入

```
struct Point {
  int x;
  int y;
  int z;
};

int main() {
  int cnt = 0;
  int sum = 0;
  int len = 10;
  struct Point points[10];

  while (cnt < len) {
    points[cnt].x = cnt;
    points[cnt].y = cnt + cnt;
    points[cnt].z = cnt * cnt;
    cnt = cnt + 1;
  }

  cnt = 0;
  while (cnt < len) {
    if (cnt < 5) {
      sum = sum + points[cnt].y + points[cnt].z;
    } else {
      sum = sum + points[cnt].x + points[cnt].y;
    }
    cnt = cnt + 1;
```

```
  }
  write(sum);
  return 0;
}
```

程序输入: 无; 预期输出: 155

说明：针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

5.3 E1-3

输入

```
struct Point {
  int x;
  int y;
  int z;
};

struct Body {
  int tag;
  struct Point points[3];
};

int dot(struct Point dp1, struct Point dp2) {
  return dp1.x * dp2.x + dp1.y * dp2.y + dp1.z * dp2.z;
}

int sqrt(int si) {
  int s_cnt = 0;
  while (s_cnt <= si) {
    int lower = s_cnt * s_cnt;
    int upper = (s_cnt + 1) * (s_cnt + 1);
    if ((si >= lower) && (si < upper)) {
      return s_cnt;
    }
```

```
    s_cnt = s_cnt + 1;
  }
  return -1;
}

int dist(struct Point dip1, struct Point dip2) {
  return sqrt(dot(dip1, dip2));
}

int main() {
  struct Body bodies[3];
  int o_cnt = 0;
  int i_cnt = 0;
  while (o_cnt < 3) {
    i_cnt = 0;
    while (i_cnt < 3) {
      bodies[o_cnt].points[i_cnt].x = i_cnt * o_cnt + i_cnt;
      bodies[o_cnt].points[i_cnt].y = i_cnt * o_cnt * o_cnt + i_cnt;
      bodies[o_cnt].points[i_cnt].z = i_cnt * i_cnt * o_cnt * o_cnt +
          i_cnt;
      i_cnt = i_cnt + 1;
    }
    bodies[o_cnt].tag = dist(bodies[o_cnt].points[1], bodies[o_cnt].
       points[2]);
    o_cnt = o_cnt + 1;
  }
  write(bodies[0].tag + bodies[1].tag + bodies[2].tag);
  return 0;
}
```

程序输入: 无; 预期输出: 19

说明：测试对于较复杂的结构体及其作为函数参数进行函数的调用。针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

5.4 E2-1

```
int main() {
  int len = 5;
  int mat1[5][5];
  int mat2[5][5];
  int o_cnt = 0;
  int i_cnt = 0;
  int m_res = 0;
  while (o_cnt < len) {
    i_cnt = 0;
    while (i_cnt < len) {
      mat1[o_cnt][i_cnt] = i_cnt;
      mat2[o_cnt][i_cnt] = o_cnt;
      i_cnt = i_cnt + 1;
    }
    o_cnt = o_cnt + 1;
  }

  o_cnt = 0;
  while (o_cnt < len) {
    i_cnt = 0;
    while (i_cnt < len) {
      m_res = m_res + mat1[o_cnt][i_cnt] * mat2[o_cnt][i_cnt];
      i_cnt = i_cnt + 1;
    }
    o_cnt = o_cnt + 1;
  }
  write(m_res);
  return 0;
}
```

程序输入: 无; 预期输出: 100

说明：测试对于简单高维数组的翻译，不涉及数组作为函数参数。针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

5.5 E2-2

输入

```
int quick_sort(int arr[6], int left, int right)
{
  int i, j, pivot;
  i = left;
  j = right;
  pivot = arr[left];

  if (i >= j) {
    return 0;
  }

  while (i < j) {
    while (i < j && arr[j] > pivot) {
      j = j - 1;
    }
    if (i < j) {
      arr[i] = arr[j];
      i = i + 1;
    }

    while (i < j && arr[i] < pivot) {
      i = i + 1;
    }
    if (i < j) {
      arr[j] = arr[i];
      j = j - 1;
    }
```

```
28    }
29    arr[i] = pivot;
30    quick_sort(arr, left, i - 1);
31    quick_sort(arr, i + 1, right);
32    return 0;
33  }
34
35  int main() {
36    int len = 6;
37    int a[6];
38    int cnt = 0;
39    int sum = 0;
40    while (cnt < len) {
41      a[cnt] = read();
42      cnt = cnt + 1;
43    }
44
45    quick_sort(a, 0, len - 1);
46
47    cnt = 0;
48    while (cnt < len) {
49      write(a[cnt]);
50      cnt = cnt + 1;
51    }
52    return 0;
53  }
```

程序输入: 4 -1 100 24 50 -100 预期输出: -100 -1 4 24 50 100

程序输入: 1 -100 255 -300 500 -1000 预期输出: -1000 -300 -100 1 255 500

说明：快速排序，测试对于数组作为函数参数的翻译。针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

5.6 E2-3

输入

```
int get_max(int max_arr[4]) {
  int max_idx = 0;
  int max_val = max_arr[0];

  int max_len = 4;
  int max_cnt = 0;
  while (max_cnt < max_len) {
    if (max_arr[max_cnt] > max_val) {
      max_idx = max_cnt;
      max_val = max_arr[max_cnt];
    }
    max_cnt = max_cnt + 1;
  }
  return max_idx;
}

int mod(int m1, int m2) {
  return m1 - (m1 / m2) * m2;
}

int main() {
  int len = 4;
  int val_arr[4][4];
  int res_arr[4];
  int copy_arr[2];
  int o_cnt = 0;
  int i_cnt = 0;

  o_cnt = 0;
  while (o_cnt < len) {
```

```
31      i_cnt = 0;
32      while (i_cnt < len) {
33        val_arr[o_cnt][i_cnt] = mod(o_cnt + i_cnt, len);
34        i_cnt = i_cnt + 1;
35      }
36      res_arr[o_cnt] = get_max(val_arr[o_cnt]);
37      o_cnt = o_cnt + 1;
38    }
39
40    copy_arr = res_arr;
41    write(copy_arr[0]);
42    write(copy_arr[1]);
43    return 0;
44  }
```

程序输入: 无; 预期输出: 3 2

说明：测试对于较复杂的数组操作的翻译，针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与屈道涵助教联系，注意同时抄送给许老师。