考试科目名称 程序设计基础

2024—2025 学年 第一学期 教师: 刘奇志、黄书剑、郭延文、孙泽群

院系（专业）_____ 年级_____ （期中）

学号_____ **姓名**_____ **成绩**_____ （闭卷）

题号	一	二	三	四
分数				

一、 简答题（15 分）5+5+5

1. 简述子程序的作用。

子程序是有名字的一段程序代码，它通常完成一个独立的（子）功能。在程序的其他地方通过子程序的名字来使用它们。
除了能减少程序代码 2分 外，
采用子程序的主要作用是实现过程抽象 2分 ，使用者只需知道子程序的功能，而不需要知道它是如何实现的，这有利于大型、复杂程序的设计和理解。
其他 1分

2. 跟循环流程相比，递归调用的函数有什么特点？

递归调用的函数是在不同的变量组（属于函数的不同实例）上进行重复操作。
优势：为某些带有重复性操作的任务提供了一种比采用循环流程更为自然、简洁 2 分 的实现方式。
缺陷：由于递归函数表达的重复操作是通过函数调用来实现的，所以需要更多时空开销 2分 ，栈空间的大小会限制递归的深度，从而降低了递归函数的可行性；有时还会出现重复计算。
（可用“动态规划”加以优化）
其他 1分

3. 全局变量与局部变量有哪些不同之处？

定义位置
初始化
作用域
生存期
副作用

二、 程序分析题（20 分）4+2+3+5+3+3

1.

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
int main()
{
    int a = 1, b = 2, c = 3;
    if (!a >= 1)
        b = 1;
        c = 1;
    printf("%d, %d ", b, c); //cout << b << ", " << c << " ";
    return 0;
}
```

该程序执行后，输出的结果是：（ 2, 1 ）

2.

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
int main()
{
    int i = 0;
    for (; i <= 10; i++);
        printf("%d ", i); //cout << i << " ";
    return 0;
}
```

该程序执行后，输出的结果是：（ 11 ）

3.

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
float p = 1.5;
void Foo(void)
{
    int p = 1;
}
#define p 2.5
int main()
{
    Foo();
    printf("%d ", (int)p); //cout << (int)p << " ";
    return 0;
}
```

该程序执行后，输出的结果是：（ 2 ）

4.

```
#include <stdio.h>  //#include <iostream>
                    //using namespace std;
void Fooo(int i)
{
    static int a = 0, b = 0;
    a++;
    b += 2;
    int sum = a + b + i;
    printf("%d ", sum); //cout << sum << " ";
}
int main()
{
    for (int i = 0; i < 5; i++)
        Fooo(i);
    return 0;
}
```

该程序执行后，输出的结果是：（ 3 7 11 15 19 ）

5.

```
int Foooo(int n)
{
    int cnt = 0;
    for(int i=1; i <= n; ++i)  {
        int m = i;
        while(m%5==0){
            ++cnt;
            m /= 5;
        }
    }
    return cnt;
}
```

该函数的功能是：（ n 的阶乘末尾有几个 0/1~n 的乘积含有多少个因子 5 ）

6.

```
int Fooooo(int n)
{
    int cnt = 0;
    while(n /= 5)
        cnt += n;
    return cnt;
}
```

该函数的功能是：（ 同上 ）

三、 程序纠错题（20 分）

说明：以下三小题的左侧 C/C++代码均存在错误，请在右侧方框中给出对应行的正确、合理的写法，不改变程序的总体算法和结构。

1. 统计所有三位正整数中满足以下条件的数的个数：个位数字与十位数字之和乘以百位数字的结果是 28。如：259、268、277、286 等是满足条件的数。

```
#include <stdio.h>   //#include <iostream>
                     //using namespace std;
int main()
{
1.    int count = 1;
      for (int i = 100; i < 1000; i++) {
         int s = i % 10;
         int t = i / 10 % 10;
2.       int h = i % 100;
3.       if ((s + t) * h = 28)
4.          count += i;
      }
      printf("%d ", count); //cout << count << " ";
      return 0;
}
```

1: 0

%: /

=: ==

i: 1

2. 计算从 start 到 end 中所有偶数的和。如 start=2，end=10，则返回 30。

```
1. int sum(int start, int end)
   {
      int sum = 0;
2.    for (int i = start; i < end; i++)
3.       if ((i & 1) == 1)
            sum += i;
      return sum;
   }
```

sum：Sum 或其他不是 sum 的标识符

<: <=

1: 0

3. 求解埃尔米特（Hermite）多项式中第 n+1 项 $H_n(x)$的值。$H_n(x)$定义为：

$H_0(x) = 1$

$H_1(x) = 2x$

$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \quad n > 1$

```
1. double Hermite_Recur(double x)
   {
      if(n == 0)
         return 1;
      else if(n == 1)
         return 2*x;
      else
2.       return 2*x*(n-1)*Hermite_Recur(x)
3.         - 2*(n-1)*(n-2)*Hermite_Recur(x);
   }
```

(**int n,** double x)

return 2*x*Hermite_Recur(**n-1,** x) -2*(n-1)*Hermite_Recur(**n-2,** x);

四、 程序填空与设计题（45 分）5+9+7+24

1. 不用 goto 或 continue ，改写下面 C/C++程序片段（line 1~5），实现等价的功能。

<table>
<tr>
<td>

```
int d, sum = 0, i = 1;
while (i <= 10) {
1.    scanf("%d", &d);
2.    if(d <= 0)
3.        continue;
4.    sum += d;
5.    ++i;
}
```

</td>
<td>多种写法：如
scanf("%d", &d);
if (d > 0) {
sum += d;
++i;
}</td>
<td>scanf("%d", &d);
while (d <= 0)
scanf("%d", &d);
sum += d;
++i;</td>
</tr>
<tr>
<td colspan="2">scanf("%d", &d);
if (d <= 0) {
sum -= d;
--i;
}
sum += d;
++i;</td>
<td>scanf("%d", &d);
if(d <= 0) {
d=0;
--i;
}
sum += d;
++i;</td>
</tr>
<tr>
<td>scanf("%d", &d);
if (d <= 0)
--i;
else
sum += d;
++i;</td>
<td>scanf("%d", &d);
sum += (d <= 0 ? 0: d);
i+= (d <= 0? 0: 1);</td>
</tr>
</table>

2. 小蓝鲸手机后四位 k 具有以下特征：前两位数字是相同的；后两位数字也是相同的，但与前两位数字不同；四位数刚好是一个整数的平方。请完善下面求解 k 的 C/C++程序片段。

```
for (int i=0; i <= 9; ++i)
    for (int j=0; j <= 9; ++j)
        if (i__!=____j)  {
            int c, k = __ i*1000 + i*100 + j*10 + j __;
            for (c=1; __ c*c < k ___; ++c);
            if(__c*c == k ___)
                return k;
        }
```

3. 自守数（又称同构数）是指一个数的平方的尾数等于该数自身的自然数。例如：25×25=625，76×76=5776，9376×9376=87909376。请完善下面求解自守数的 C/C++函数。

```
bool IsomorphM(int n)
{
    int m = n * n;
    while (____n > 0___) {
        if (____n % 10 != m % 10____)
            return _ false _;
        n /= 10, m /= 10;
    }
    return true;
}
```

4. 一个数 n 满足如下条件，则称 n 是质数 plus:
（1）n 本身是质数（素数）;
（2）n 的逆序数也是质数（注：例如 1234 的逆序数是 4321，230 的逆序数是 32）;
（3）n 不是回文数（注：如 1234321 逆序也是 1234321，则它是回文数）。
请设计 C/C++程序计算区间[a, b]中的质数 plus 的个数。
要求：定义三个函数分别实现质数的判断、逆序数的求解和质数 plus 的判断，并在 main 函数中输入 a 和 b、质数 plus 的计数和结果的输出。禁止使用数组，禁止使用 STL 库中提供的函数。

```
#include <iostream>
#include <cmath>
using namespace std;

bool isPrime(int num) {
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 == 0 || num % 3 == 0) return false;

    for (int i = 5; i * i <= num; i += 6) {
        if (num % i == 0 || num % (i + 2) == 0) return false;
    }

    return true;
}

bool isPalindrome(int num) {
    int originalNum = num;
    int reversedNum = 0;

    while (num > 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
```

```
        num /= 10;
    }

    return originalNum == reversedNum;
}

int reverseNum(int num) {
    int reversedNum = 0;

    while (num > 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
        num /= 10;
    }

    return reversedNum;
}

int countPrimePlus(int a, int b) {
    int count = 0;

    for (int num = a; num <= b; num++) {
        if       (isPrime(num)       &&       isPrime(reverseNum(num))
&& !isPalindrome(num)) {
            count++;
        }
    }

    return count;
}

int main() {
    int a, b;
    cin >> a >> b;

    int primePlusCount = countPrimePlus(a, b);
    cout << primePlusCount << endl;

    return 0;
}
```