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-算数语句、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 a = 3, b = 8, c = 24;
    int final;
    b = (a * a + c / b)  * 4;
    write(b);
    c = (b + 3 * a - c) * 55 / a;
    write(c);
    final = a + 10 * a * b + c * 4 + (b / a - 6);
    write(final);
    return 0;
}
```

程序输入：无；预期输出：48 605 3873

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 a, b, c;
    int cirf, type;
    a = read();
    b = read();
    c = read();
```

```
    if(a + b > c && b + c > a && a + c > b)
    {
        cirf = a + b + c;
        if(a == b && a == c)
            type = 1;
        else if(a == b || a == c || b == c)
            type = 2;
        else if((a * a + b * b == c * c)||(a * a + c * c == b * b)||(
           b * b + c * c == a * a))
            type = 3;
        else
            type = 0;
        write(type);
        write(cirf);
    }
    else
        write(-1);
    return 0;
}
```

输入：1 1 4；输出：-1

输入：3 4 5；输出：3 12

输入：5 5 5；输出：1 15

输入：3 2 2；输出：2 7

输入：8 7 6；输出：0 21

说明：一个输入三条边判断三角形类型和周长的小程序，主要针对分支语句进行测试。注意，程序输入以空格隔开，每次输入一个数（下同）。

1.3 A-3

输入

```
int main()
{
    int i,tu1,tu2,tu3,m;
```

```
    tu1 = 1;
    tu2 = 1;
    m = read();
    if(m < 1)
        write(0);
    else if(m == 1||m == 2)
        write(1);
    else if(m > 2)
    {
        i = 3;
        while(i <= m)
        {
            tu3 = tu1 + tu2;
            tu1 = tu2;
            tu2 = tu3;
            i = i + 1;
        }
        write(tu3);
    }
    return 0;
}
```

输入：2；输出：1

输入：5；输出：5

输入：12；输出：144

输入：-1；输出：0

说明：这个测试用例主要针对循环语句进行测试，迭代计算斐波那契数列。

1.4 A-4

输入

```
int main()
{
    int i = 0, j, t, a[5];
```

```
    while(i < 5)
    {
        a[i] = read();
        i = i + 1;
    }
    i = 0;
    while(i < 4)
    {
        j = i + 1;
        while(j < 5)
        {
            if(a[i] > a[j])
            {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
            j = j + 1;
        }
        i = i + 1;
    }
    i = 0;
    while(i < 5)
    {
        write(a[i]);
        i = i + 1;
    }
    return 0;
}
```

输入：35 25 12 14 12；输出：12 12 14 25 35

说明：这个测试用例主要针对一维数组进行测试，实现升序选择排序。

1.5 A-5

输入

```
int mod(int x,int n)
{
    return x - (x / n) * n;
}

int main()
{
    int year,a;
    year = read();
    if(mod(year, 400) == 0)
        a = 1;
    else
    {
        if(mod(year, 4) == 0 && mod(year, 100) != 0)
            a = 1;
        else
            a = 0;
    }
    write(a);
    return 0;
}
```

输入：2020；输出：1

输入：2000；输出：1

输入：2019；输出：0

输入：2100：输出：0

说明：一个闰年判断的小程序，主要针对函数的调用进行简单测试。

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

```
int mod(int x,int n)
{
    return x -(x / n) * n;
}

int DigitSum(int y)
{
     if(y == 0)
        return 0;
     return mod(y, 10) + DigitSum(y / 10);
}

int main()
{
    int num;
    num = read();
    if(num < 0)
        write(-1);
    else
        write(DigitSum(num));
    return 0;
}
```

输入：23145；输出：15

说明：计算一个正整数各个数位之和的小程序，考察复杂的函数调用和递归。

2.2 B-2

输入

```
int main()
{
    int primes[10], N = 10;
    int pc, m, k;
    primes[0] = 2;
    pc = 1;
    m  = 3;
    while(pc < N)
    {
        k = 0;
        while(primes[k] * primes[k] <= m)
            if(m == primes[k] * (m / primes[k]))
            {
                m = m + 2;
                k = 1;
            }
            else
                k = k + 1;
        primes[pc] = m;
        pc = pc + 1;
        m = m + 2;
    }
    k = 0;
    while(k < pc)
    {
        write(primes[k]);
        k = k + 1;
    }
    return 0;
}
```

输入：无；输出：2 3 5 7 11 13 17 19 23 29

说明：筛法求质数。

2.3 B-3

输入

```
int Joseph(int LEN)
{
    int a[100];
    int i = 0;
    int leftCount = LEN;
    int index = 0,count = 0;
    while (i < LEN)
    {
        a[i] = 1;
        i = i + 1;
    }

    while(leftCount > 2)
    {
        if(a[index] == 1)
        {
            count = count + 1;
            if(3 == count)
            {
                a[index] = 0;
                count = 0;
                leftCount = leftCount - 1;
            }
        }
        index = index + 1;

        if(index == LEN)
```

```
            index = 0;
    }
    i = 0;
    while (i < LEN)
    {
        if(1 == a[i])
            write(i + 1);
        i = i + 1;
    }
    return 0;
}
int main()
{
    int N;
    N = read();
    if (!(N > 3 && N <= 100))
        write(-1);
    else Joseph(N);
    return 0;
}
```

输入：41；输出：16 31

说明：约瑟夫问题，输出剩余两人的编号。

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 mod(int x,int y)
{
    return x -(x / y) * y;
```

```
}

int gcd(int c, int d)
{
    if(c==0)
        return d;
    return gcd(mod(d, c),c);
}

int lcm(int e, int f)
{
    return e * f / (gcd(e, f));
}

int main()
{
    int n, i, g;
    int tmp, sum = 0;
    int a[50];
    int b[50];
    n = read();
    i = 0;
    while(i < n)
    {
        a[i] = read();
        b[i] = read();
        i = i + 1;
    }
    tmp = b[0];
    i = 1;
    while(i < n)
    {
```

```
        tmp = lcm(tmp, b[i]);
        i = i + 1;
    }
    i = 0;
    while(i < n)
    {
        sum = sum + a[i] * (tmp / b[i]);
        i = i + 1;
    }
    g = gcd(sum, tmp);
    sum = sum / g;
    tmp = tmp / g;
    if(tmp == 1)
        write(sum);
    else
    {
        write(sum);
        write(tmp);
    }
    return 0;
}
```

输入：5 2 5 4 15 1 30 2 60 8 3；输出：17 5

说明：分式相加，输入是分式的个数以及每个分式的分子和分母，输出是结果的最简分式的分子和分母。

3.2 C-2

输入

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

```

```
int IsLeap(int y)
{
    if(mod(y, 400) == 0 || mod(y, 4) == 0 && mod(y, 100) != 0)
         return 1;
    else
         return 0;
}

int main()
{
    int year, i, dayofweek, motherday, days=0, leap=0;
    int monthdays[5];
    monthdays[0] = 0;
    monthdays[1] = 31;
    monthdays[2] = 28;
    monthdays[3] = 31;
    monthdays[4] = 30;
    year = read();
    i = 1900;
    while(i < year)
    {
        if(IsLeap(i))
            days = days + 366;
        else
            days = days + 365;
        i = i + 1;
        }
        if(IsLeap(year))
                monthdays[2] = 29;
        i = 1;
        while(i < 5)
        {
```

```
            days = days + monthdays[i];
            i = i + 1;
        }
        dayofweek = mod(days, 7);
        motherday = 14 - dayofweek;
        write(5);
        write(motherday);
        return 0;
}
```

输入：2020；输出：5 10

输入：2100；输出：5 9

说明：输入年份，输出该年的母亲节在哪一天。

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 process(int x)
{
    int y = 4, z;
    z = x + 12;
    z = x + 12;
    y = y;
    y = y + 0 - 0;
    y = y * 1 / 1;
    y = y + 32 - 4 * 7 / 10;
    y = x * y  + x * y - y * x - x + y * x + y * y + x / x - 12 + 59
       / 2;
    y  = y * 3 + 14 * 24 - x * 12 / 4 + 4 * 5 - 10 * 2 - 5 / 6;
    return y;
```

```
}

int mod(int a1, int b1)
{
    return a1 - (a1 / b1) * b1;
}

int main () {
    int a = (-4 * 2 + 108) / 17, b = 32 / 8 * 2 - 1, c = 13 - 1 * 4 /
        2;
    int d = a + b;
    int e = a + b + c / 1;
    int f = a * b - c;
    int g1 = 42, g2 = 4, i = 0, j = 0, array[4];
    int g, h, k;
    f = a + b + c + 1500 - f;
    while (i < 4)
    {
        j = 15 * i;
        array[i] = j;
        i = i + 1;
    }
    while (i < f)
    {
        k = g2 * g2;
        g1 = g1 + k + i * 12 - 4 * g2 + 5 + 7 / 3;
        g = process(f) + 2 * a  - f + c * d;
        if (mod(f,2) > 0)
        {
            h = i + 3;
            h = h - 1;
            h = h + 3;
```

```
            h = h - 3 * 2;
        }
        if (process(a) == process(a + 3 - 2 - 1))
        {
            f = f - 2 + 1;
            array[mod(f, 4)] = array[mod(f, 4)] + h + g - e;
        }
        a = a + 2 + 1;
        i = i + 1;
        i = i + 1;
    }
    a = a + b;
    b = a + b;
    c = a + b;
    f = a + b;
    g = a + b;
    write(c + f + g);
    write(array[0]);
    write(array[1]);
    write(array[2]);
    write(array[3]);
    return 0;
}
```

输入：无；输出：9075 31504125 31527515 31551405 31275325

说明：程序中有多个可优化点，包括常量折叠，公共子表达式等。首先需要保证中间代码的正确性，要能准确输出最后的结果，才能参加后面的效率竞赛。

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 Student
{
    int ID;
    int score;
};

int main()
{
    struct Student s1, s2;
    s1.ID = 1;
    s1.score = 70;
    s2.ID = 2;
    s2.score = 90;
    write(s2.ID * s1.score);
    return 0;
}
```

输入：无；输出：140

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 Book
{
    int book_id;
    int year;
    int price;
};

```

```
int main( )
{
    struct Book books[8];
    int i = 0, j = 0, sum = 0, r;
    while(i < 8)
    {
        books[i].book_id = i;
        books[i].year = 2012 + i;
        books[i].price = 30 + i * 2;
        i = i + 1;
    }
    while(j < 8)
    {
        if(books[j].book_id < 4)
            r = 1;
        else
            r = 2;
        sum = sum + books[j].price * r * (2020 - books[j].year);
        j = j + 1;
    }
   write(sum);
   return 0;
}
```

输入：无；输出：1648

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 Worker
{
    int id;
    int salary;
```

```
    int group;
};

struct Group
{
    int group_id;
    struct Worker workers[10];
    int avg_salary;
};

int calculate_avg(struct Group g)
{
    int sum = 0, k = 0;
    while (k < 10)
    {
        sum = sum + g.workers[k].salary;
        k = k + 1;
    }
    g.avg_salary = sum / 10;
    return g.avg_salary;
}
int main()
{
    struct Group company[5];
    int i = 0 , j, avg = 0;
    while(i < 5)
    {
        j = 0;
        company[i].group_id = i;
        while (j < 10)
        {
            company[i].workers[j].salary = 3000 + i * 100 + j * 150;
```

```
            j = j + 1;
        }
        company[i].avg_salary = calculate_avg(company[i]);
        avg = avg + company[i].avg_salary;
        i = i + 1;
    }
    avg = avg / 5;
    write(avg);
    return 0;
}
```

输入：无；输出：3875

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 i = 0, j, t[15][15], n;
    n = read();
    while(i < n)
    {
        t[i][0] = 1;
        j = 1;
        while(j < i)
        {
            t[i][j] = t[i - 1][j - 1] + t[i - 1][j];
            j = j + 1;
        }
        t[i][j] = 1;
        i = i + 1;
    }
    j = 0;
```

```
    while(j <= n - 1)
    {
        write(t[n - 1][j]);
        j = j + 1;
    }
      return 0;
}
```

输入：8；输出：1 7 21 35 35 21 7 1

说明：输入 n，输出杨辉三角第 n 行。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 binary_search(int key, int a[7], int n)
{
    int low = 0, high = n-1, mid, count = 0, flag = 0;
    while(low <= high && flag == 0)
    {
        count = count + 1;
        mid = (low + high) / 2;
        if(key < a[mid])
            high = mid-1;
        else if(key > a[mid])
            low = mid + 1;
        else if(key == a[mid])
        {
            write(mid+1);
            write(count);
            flag = 1;
        }
    }
    if(flag == 0)
```

```
        write(-1);
    return 0;
}
int main()
{
    int i = 0, k, b[7], N = 7;
    while(i < N)
    {
        b[i] = read();
        i = i + 1;
    }
    k = read();
    binary_search(k,b,N);
    return 0;
}
```

输入：1 3 5 7 9 11 13 11；输出：6 2

输入：2 4 6 8 10 12 14 3；输出：-1

说明：迭代的二分查找法，输入升序排好的 7 个数字和查找的数字，输出目标所在位置和查找次数，不存在则输出-1。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 Swap(int a[8], int l, int h)
{
    int temp;
    temp = a[l];
    a[l] = a[h];
    a[h] = temp;
    return 0;
}

```

```
10 int Partition(int b[8], int low, int high)
11 {
12     int base = b[low];
13     while(low < high)
14     {
15         while(low < high && b[high] >= base)
16         {
17             high = high - 1;
18         }
19         Swap(b, low, high);
20         while(low < high && b[low] <= base)
21         {
22             low = low + 1;
23         }
24         Swap(b, low, high);
25     }
26     return low;
27 }
28
29 int QuickSort(int c[8], int low1, int high1)
30 {
31     if(low1 < high1)
32     {
33         int base1 = Partition(c, low1, high1);
34         QuickSort(c, low1, base1 - 1);
35         QuickSort(c, base1 + 1, high1);
36     }
37     return 0;
38 }
39
40 int main()
41 {
```

```
    int n = 8;
    int arr[8];
    int i = 0;
    n = read();
    while(i < n)
    {
        arr[i] =  read();
        i = i + 1;
    }
    QuickSort(arr, 0, n-1);
    i = 0;
    while(i < n)
    {
        write(arr[i]);
        i = i + 1;
    }
    return 0;
}
```

输入：23 5 19 23 6 6 2 35；输出：2 5 6 6 19 23 23 35

说明：快速排序。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，可以写邮件与李聪助教或陈紫琦助教联系，注意同时抄送给许老师。