# 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 x, y, z;
        int a, b, c;
        int t = 3;
        x = 12;
        b = x * x + 13;
        y = b / 13 + 1;
        z = x / b + b * y;
        write(z);
        c = t + z * 2;
        write(c);
        t = x + b + y + x / c + t;
        write(t);
        return 0;
}
```

程序输入：无；预期输出：2041 4085 185

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 dollar, rmb;
        dollar = read();
        rmb = read();
        if (dollar > 100) {
                if (rmb < 50) {
                        write(rmb + dollar);
                } else {
                        write(rmb - dollar);
                }
        } else if (dollar == 100) {
                if (rmb < 100) {
                        write(rmb);
                } else {
                        write(rmb - 100);
                }
        } else if (dollar < 100) {
                if (rmb + dollar > 100){
                        write(dollar + 100);
                } else {
                        write(100);
                }
        }
        write(dollar + rmb);
        return 0;
}
```

输入：105 35；输出：140 140

输入：100 100；输出：0 200

输入：95 40; 输出：195 135

输入：5 5；输入：100 10

说明：这个测试用例主要针对分支语句进行测试的小程序。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 x, n, result;
        x = read();
        n = read();
        if (n == (n / 2 * 2)) {
                result = 1;
        } else {
                result = x;
        }
        n = n / 2;
        while (n > 0) {
                x = x * x;
                if (n != (n / 2 * 2)) {
                        result = result * x;
                }
                n = n / 2;
        }
        write(result);
        return 0;
}
```

输入：2 2；输出：4

输入：3 3；输出：27

输入：5 10；输出：9765625

输入：1 25；输出：1

说明：这个测试用例主要针对循环语句进行测试，求 a 的 b 次幂。

## 1.4 A-4

输入

```
int main() {
        int x[5], tem, i, j;
        i = 0;
        while(i < 5) {
                x[i] = read();
```

```
			i = i + 1;
		}
		i = 1;
		while (i < 5) {
			j = i;
			while (j > 0 && x[j-1] > x[j]) {
				tem = x[j];
				x[j] = x[j-1];
				x[j-1] = tem;
				j = j - 1;
			}
			i = i + 1;
		}
		i = 0;
		while (i < 5) {
			write(x[i]);
			i = i + 1;
		}
		return 0;
}
```

输入：32 20 15 19 12；输出：12 15 19 20 32

说明：这个测试用例主要针对一维数组进行测试，实现升序插入排序。

## 1.5 A-5

输入

```
int swap(int a, int b) {
		int tem = a;
		a = b;
		b = tem;
		write(a);
		write(b);
		return a;
}

int main() {
        int x[5];
        int i = 0;
        while (i < 5) {
                x[i] = read();
                if (i > 0) {
                        swap(x[i-1],x[i]);
                }
                i = i + 1;
        }
        return 0;
}
```

输入：2 3 6 8 10；输出：3 2 6 3 8 6 10 8

说明：这个测试用例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 myPow(int x, int n) {
        int value;
        if (n == 0){ return 1;}
        if (n == 1){ return x;}
        if (n == 2){ return x * x;}
        if (n == (n/2*2)){ return myPow(myPow(x,n/2),2);}
        else {
                value = myPow(myPow(x,n/2),2);
                return x*value;}
}

int main() {
        int x1, n1;
        x1 = read();
        n1 = read();
        write(myPow(x1,n1));
        return 0;
}
```

输入：2 4；输出：16

输入：11 5；输出：161051

输入：12 3；输出：1728

输入：15 0；输出：1

说明：求 a 的 b 次幂的递归版本，考察复杂的函数调用和递归。

## 2.2 B-2

输入

```
int countSort(){
        int x[5], count[10], sorted[5], i;
        i = 0;
        while (i < 10) {
                count[i] = 0;
                i = i + 1;
        }
        i = 0;
        while (i < 5) {
                x[i] = read();
                count[x[i]] = count[x[i]] + 1;
                i = i + 1;
        }
        i = 1;
        while ( i < 10) {
                count[i] = count[i] + count[i-1];
                i = i + 1;
        }
        i = 0;
        while ( i < 5 ) {
                sorted[count[x[i]]-1] = x[i];
                count[x[i]] = count[x[i]] - 1;
                i = i + 1;
        }
        i = 0;
        while (i < 5) {
                write(sorted[i]);
                i = i + 1;
        }
        return 0;
}

int bubbleSort(){
        int a[5], b, c, tem;
        b = 0;
        while (b < 5) {
                a[b] = read();
                b = b + 1;
        }
        c = 1;
        while (c == 1) {
                c = 0;
                b = 1;
                while (b < 5) {
                        if (a[b] < a[b-1]) {
                                c = 1;
                                tem = a[b-1];
                                a[b-1] = a[b];
                                a[b] = tem;
                        }
                        b = b + 1;
                }
        }
        b = 0;
        while (b < 5) {
                write(a[b]);
                b = b + 1;
        }
        return 0;
}

int main(){
        countSort();
        bubbleSort();
        return 0;
}
```

输入：5 3 2 4 1 5 3 2 4 1；输出：1 2 3 4 5 1 2 3 4 5

说明：实现计数排序和冒泡排序的数组排序程序。

### 2.3 B-3

输入

```
int search(int target) {
        int x[5], left, right, index, middle;
        int i = 0;
        while(i < 5) {
                x[i] = read();
                i = i + 1;
        }
```

```c
        left = 0;
        right = 4;
        while (left <= right) {
                index = (left + right) / 2;
                middle = x[index];
                if (middle == target) {return index;}
                if ((middle > x[left] && target >= x[left] && target
                   < middle) || (middle < x[left] && (target >= x[
                   left] || target < middle))) {
                        right = index -1;
                } else {
                        left = index + 1;
                }
        }
        return -1;
}

int main(){
        int n;
        n = read();
        write(search(n));
        return 0;
}
```

输入：0 6 7 0 1 2；输出：2

输入：3 67 68 0 2 4；输出：-1

说明：谷歌笔试题，一个升序有序数组（无重复元素，如输入 1 中 0 1 2 6 7）以未知位置为中心对调（6 7 0 1 2），二分搜索特定元素（0），返回其数组中位置（2）或-1（若不存在）。

## 3 C 组测试用例

本组测试用例共 2 个，是经典问题。

## 3.1 C-1

输入

```
int main(){
        int count, i, valid, j, k, cont, n;
        int place[8];
        int row[8];
        int ldiag[8];
        int rdiag[8];
        count = 0;
        i = 0;
        n = read();
        while(i < n) {
                place[i] = -1;
                i = i + 1;
        }
        i = 0;
        cont = 1;
        while(cont == 1) {
                if (i == n) {
                        valid = 1;
                        j = 0;
                        while (j < n) {
                                row[j] = 1;
                                ldiag[j] = 1;
                                rdiag[j] = 1;
                                j = j + 1;
                        }
                        j = 0;
                        while (j < n) {
                                if(row[place[j]] != 1
                                || ldiag[place[j]] != 1
                                || rdiag[place[j]] != 1) {
```

```
                    valid = 0;
                    j = n;

                } else {
                    row[place[j]] = 0;
                    k = 0;
                    while (k < n-1) {
                            ldiag[k]
                        = ldiag[k + 1];
                            k = k + 1;


                    }
                    ldiag[n-1] = 1;
                    if (place[j] != 0)
                    { ldiag[place[j] - 1] = 0;}
                    k = n-1;
                    while(k > 0) {
                            rdiag[k]
                        = rdiag[k - 1];
                            k = k - 1;


                    }
                    rdiag[0] = 1;
                    if (place[j] != n-1)
                    {rdiag[place[j] + 1] = 0;}
                  j = j + 1;
                }
            }
            if (valid == 1) {
                count = count + 1;
```

```
                                }
                        i = i - 1;
                } else {
                        while (i >= 0 && place[i] >= n-1) {
                                place[i] = -1;
                                i = i - 1;
                        }
                        if (i == -1){
                                cont = 0;
                        } else {
                                place[i] = place[i] + 1;
                                i = i + 1;
                        }
                }
        }
        write(count);
        return 0;
}
```

输入：2；输出：0 输入：4；输出：2

说明：这个测试用例实现了经典 N 皇后问题，输入 N，输出所有可能的摆放方式数目。

### 3.2 C-2

输入

```
int trap() {
        int lh = 0, lIndex = 0, i = 0, count = 0, thisPool = 0;
        int n = 12;
        int rh = 0;
        int height[12];
        while (i < n) {
                height[i] = read();
                i = i + 1;
        }
        i = 0;
        while (i < n && height[i] == 0) {
                i = i + 1;
        }
        if (i >= n) {
                return 0;
        }
        lh = height[i];
        lIndex = i;
        thisPool = 0;
        i = i + 1;
        while (i < n) {
                if (height[i] < lh){
                        thisPool = thisPool + (lh - height[i]);
                }else {
                        count = count + thisPool;
                        lh = height[i];
                        lIndex = i;
                        thisPool = 0;
                }
                i = i + 1;
        }
        thisPool = 0;
        rh = 0;
        i = n - 1;
        while (i > lIndex && height[i] == 0) {
                i = i - 1;
        }
        rh = height[i];
        i = i - 1;
        while (i > lIndex) {
                if (height[i] < rh) {
                        thisPool = thisPool + (rh - height[i]);
                } else {
                        count = count+ thisPool;
                        rh = height[i];
                        thisPool = 0;
                }
                i = i - 1;
        }
        return count + thisPool;
}

int main (){
        int result = trap();
        write(result);
        return 0;
}
```

输入：0 1 0 2 1 0 1 3 2 1 2 1；输出：6

说明：给出一个 n 维的非负数组代表一个海拔图，每个数字代表的宽度为 1，计算下雨后一共可以积存多少水。

## 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 = 3;
        y = 11 * 3 - 2 + 5;
        y = x * 321 * 2 + x * y - x + y * x + y * y + x + x - 23 +
           45;
        y = y / 3 + 14 * 24 - x * 12 / 4  - 20 * 3 + y / 12 *24 + 12
           * 3 + 3 / 2;
        y = x + 4 * 6 + 3 / 2;
        return y;
}

int main () {
        int a = 5 / 2 + 14 -3, b = 7 * 5 / 2 + 3, c = 4 + 5 + 6 - 1 /
            2;
        int d = a + b + c;
        int e = a * b + c / 2;
        int f = a - b - c;
        int g1 = 42, i = 0;
        int g, h;
        f = a + b + c + 1000 * 2 - f;
        while (a + b < f) {
                g1 = g1 + i * 12 + 4 +5 +7 / 3;
                g = process(f) + 2 * a  - f + c * d;
                i = i + i;
                i = i + i;
                i = i + i;
                i = i + i;
                i = i + i;
                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;
                }
                a = a + 2 + 1;
        }
        h = g1 - 3 * 4;
        while (h < g1) {
                f = 15 * 4 - 2 + a;
                g = g1 -12;
                h = h + 1;
                g = g1;
                i = a + b;
                c = a + b;
        }
        write(f);
        a = a + b;
        b = a + b;
        c = a + b;
        f = a + b;
        g = a + b;
        write(c+f+g);
        return 0;

}
```

输入：无；输出：1601 9438

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 Product{
        int type;
        int name;
};

int main () {
        struct Product cola;
        cola.type = read();
        cola.name = 3;
        write(cola.type+cola.name);
        return 0;
}
```

输入：3；输出：6

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 Product{
        int type;
        int name;
};

int main(){
        struct Product cola[10];
        int i, j, add, N = 10;
        i = 0;
        while(i < N) {
                cola[i].type = 10;
                cola[i].name = i;
                i = i + 1;
        }
        i = 0;
        add = 0;
        while (i < N) {
                j = 0;
                add = add + cola[i].name;
                while (j < N) {
                        cola[i].type = cola[i].type + add * cola[j].
                           name;
                        j = j + 1;
                }
                i = i + 1;
        }
        write(cola[N-1].type);
        return 0;
}
```

输入：无；输出：2035

说明：测试对于结构体作为数组的类型。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 Student{
        int name;
        int grade;
};

struct Class{
        struct Student students[50];
        int average;
};

int calculate(struct Class class) {
        int sum = 0, i = 0, N = 50;
        while (i < N) {
                sum = sum + class.students[i].grade;
                i = i + 1;
        }
        class.average = sum / N;
        return sum / N;
}

int main() {
        struct Class school[10];
        int i1 = 0, j1 = 0, N1 = 50, N2 = 10;
        while (i1 < N2) {
                j1 = 0;
                while (j1 < N1) {
                        school[i1].students[j1].grade = i1 + j1 * 5;
                        j1 = j1 + 1;
                }
                i1 = i1 + 1;
        }
        j1 = 0;
        i1 = 0;
        while (i1 < N2) {
                j1 = j1 + calculate(school[i1]);
                i1 = i1 + 1;
        }
        write(j1);
        return 0;
}
```

输入：无；输出：1265

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

输入

```
int main(){
        int dis[10][5][3];
        int d1 = 10, d2 = 5, d3 = 3;
        int i = 0, j = 0, k = 0, sum = 0;
        while (i < d1) {
                j = 0;
                while (j < d2) {
                        k = 0;
                        while (k < d3) {
                                dis[i][j][k] = i * d1 + j * d2 + d3 /
                                    (k+1);
                                k = k + 1;
                        }
                        j = j + 1;
                }
                i = i + 1;
        }
        i = 0;
        j = 0;
        k = 0;
        while (i < d1) {
                j = 0;
                while (j < d2) {
                        k = 0;
                        while (k < d3) {
                                if (dis[i][j][k] > dis[0][0][0]) {
                                        sum = sum + dis[i][j][k];


                                }
                                k = k + 1;
                        }
                        j = j + 1;
                }
                i = i + 1;
        }
        write(sum);
        return 0;
}
```

输入：无；输出：8495

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 qsort(int array[10], int l, int r) {
        int x = array[l], a = l, b = r;
        if (a < b) {
                while (a < b) {
                        while (a < b && array[b] > x) {b = b - 1;}
                        if (a < b) {array[a] = array[b]; a = a + 1;}
                        while (a < b && array[a] < x) {a = a + 1;}
                        if (a < b) {array[b] = array[a]; b = b - 1;}
                }
                array[a] = x;
                qsort(array,l,a-1);
                qsort(array,a+1,r);
        }
    return 0;
}

int main() {
        int number[10], N = 10, i = 0;
        while (i < N) {
                number[i] = read();
                i = i + 1;
        }
        qsort(number,0,N-1);
        i = 0;
        while (i < N) {
                write(number[i]);
                i = i + 1;
        }
        return 0;
}
```

输入：8 9 6 5 4 7 1 2 3 0；输出：0 1 2 3 4 5 6 7 8 9

说明：测试对于数组作为函数参数的翻译，实现快速排序。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 display(int chess[10], int number[1], int sum){
        int board[10][10], i1 = 0, j1 = 0, tem = 1;
        if (number[0] == 1) {
                while (i1 < sum) {
                        j1 = 0;
                        tem = 1;
                        while (j1 < sum) {
                                if (j1 == chess[i1]) {
                                        board[i1][j1] = 1;
                                        tem = tem * 10 + 1;
                                } else {
                                        board[i1][j1] = 0;
                                        tem = tem * 10;
                                }
                                j1 = j1 + 1;
                        }
                        write(tem);
                        i1 = i1 + 1;
                }
        }
        return 0;
}

int dfs(int p[10], int r[10], int ld[10], int rd[10], int current,
   int target, int c[1]){
        int j = 0, nld[10], nrd[10], k;
        if (current == target) {
                c[0] = c[0] + 1;
                display(p,c,target);
                return 0;
        }
        while (j < target) {
                if (r[j] == 1 && ld[j] == 1 && rd[j] == 1 ) {
                        p[current] = j;
                        r[j] = 0;
                        k = 0;
                        while (k< target - 1){
                                nld[k] = ld[k + 1];
                                k = k + 1;
                        }
                        nld[target -1] = 1;
                        if (j != 0) {
                                nld[j - 1] = 0;
                        }
                        k = target-1;
                        while (k > 0){
                                nrd[k] = rd[k-1];
                                k = k - 1;
                        }
                        nrd[0] = 1;
                        if (j != target -1){
                                nrd[j + 1] = 0;
                        }
                        dfs(p, r, nld, nrd, current + 1, target, c);
                        r[j] = 1;
                }
                j =  j + 1;
        }
        return 0;
}
int main() {
        int place[10], N, count[1];
        int row[10], ldiag[10], rdiag[10] ,i = 0;
        N = read();
        if (N == 0 || N > 10) { return 0;}
        while(i < N) {
                row[i] = 1;
                ldiag[i] = 1;
                rdiag[i] = 1;
                i = i + 1;
        }
        count[0] = 0;
        dfs(place,row,ldiag,rdiag,0,N,count);
        write(count[0]);
        return 0;
}
```

输入：8；输出：110000000 100001000 100000001 100000100 100100000 100000010 101000000 100010000 92

说明：测试对于较复杂的数组操作的翻译，是一个八皇后问题，输出第一个搜索到的摆放方案（每行 1 开头，之后八位代表摆放，1 代表放置皇后），并输出总共的解法数目。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，可以写邮件与王珏助教联系，注意同时抄送给许老师，本学期编译原理实验到此结束，祝愿大家都能取得好的成绩。