第一次上机解题报告

15081070 张雨任

一、A 题

1. 题目代号与评测记录序号：{A249034}
2. 代码：

```
#include <iostream>
using namespace std;
void swap1(int,int);
void swap2(int&,int&);
void swap3(int*,int*);
int main()
{
    int a,b,aa,bb;
    while(cin>>a){
        cin>>b;
        aa=a;bb=b;
        swap1(aa,bb);
        cout<<aa<<" "<<bb<<endl;
        aa=a;bb=b;
        swap2(aa,bb);
        cout<<aa<<" "<<bb<<endl;
        aa=a;bb=b;
        swap3(&aa,&bb);
        cout<<aa<<" "<<bb<<endl;
    }
    return 0;
}

void swap1(int a,int b)
{
    int temp=a;
    a=b;
    b=temp;
}

void swap2(int& a,int& b)
{
    int temp=a;
    a=b;
    b=temp;
}

void swap3(int* p,int *q)
{
    int temp=*p;
    *p=*q;
    *q=temp;
}
```

3. 解题思路：
   这道题考察的是三种函数参数传递的方式，分别是按值传递，按引用传递和指针传递。按值传递是搞了一个原函数的副本出来，并且在副本上对值进行修改，所以在函数外面输出 a 和 b 的时候其实他们真正的值并没有被改变，只有副本被改变了，假设输入 1 和 2，那么输出的还是 1 和 2。而按引用传递则是没有复制一个副本，直接在原函数上对值进行修改，因此就算在函数外面输出两个值，他们也是被交换过，即假设输入 1 和 2，则输出 2 和 1。而第三种方法指针传递，可以真正交换两个数的值，即声明一个 int 类型的变量 temp 作为中间值，交换*p 和*q 的值，意思就是交换 p 和 q 指向的值，其实就是交换了 a 和 b 的值，这个时候在函数外输出 a 和 b，就是交换成功的值了。

二、B 题

1. 题目代号与评测记录序号：{B247784}
2. 代码：

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

int main()
{
    int n;
    while(cin>>n){
        int t=0;
        int tgzh=0;
        int gzh=0;
        int key=0;
        int c[n+1];
        for(int i=1;i<=n;i++){
            cin>>c[i];
            t+=c[i];
        }
        double half=static_cast<double>(t)/2;
        for(int i=1;i<=n;i++){
            tgzh+=c[i];
            gzh++;
            if(tgzh>half){
                key=i;
                gzh--;
                tgzh-=c[i];
                break;
            }
        }
        if(tgzh<=t-tgzh-c[key]){
            gzh++;
        }
        cout<<gzh<<" "<<n-gzh<<"\n";
    }
    return 0;
}
```

3. 解题思路：

   我对于这道题的解决思路是找到总时间的中点，通俗来讲就是让 Gzh 一直吃下去，每吃一次就累加 Gzh 吃的时间（tgzh）和 Gzh 吃掉的巧克力个数（gzh），然后每次循环都判断 Gzh 吃的时间（tgzh）有没有超过总时间的中点（要注意 half 需要是个 double，因为总时间可能是个奇数），如果超过了 half 就让他吐出来这块，也就是减去这块巧克力的时间，并且自己吃的巧克力个数（gzh）也要自减，然后跳出循环。之后比较在争夺最后一块巧克力之前 Gzh 和 Syw 吃巧克力的时间，时间短的人吃掉最后一块也就是处于时间中点的巧克力，时间相等则 Gzh 吃。最后输出 Gzh 的个数（gzh）和 Sym 的个数（n-gzh）。
4. 易错点：

   这道题 wa 了很多次，其实不是不会做，而是没有仔细地想迭代的时候每个变量都是什么状态，凭直觉就写上去了。教训就是写循环不能想当然，应该想清楚每条语句的意思，含糊就容易出错。思路不清晰就不要乱写，可以在

纸上画一下。

5. 调试经验：不确定的地方可以打印到屏幕上一下这样比较清楚。

三、C 题

1. 题目代号与评测记录序号：{C247300}
2. 代码：

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
double boy[10000];
double girl[10000];
double remainboy[10000];
double remaingirl[10000];
int numboy[10000];
int numgirl[10000];

int main()
{
    int n=0;
    while(cin>>n){
        for(int i=0;i<n;i++){
            cin>>boy[i]>>girl[i];
            remainboy[i]=boy[i];
            remaingirl[i]=girl[i];
        }
        sort(boy,boy+n);
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(boy[i]==remainboy[j]){
                    numboy[i]=j+1;
                }
            }
        }
        sort(girl,girl+n);
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(girl[i]==remaingirl[j]){
                    numgirl[i]=j+1;
                }
            }
        }
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(numboy[i]==numgirl[j]){
                    cout<<j+1<<" ";
                }
            }
        }
        cout<<endl;
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(numgirl[i]==numboy[j]){
                    cout<<j+1<<" ";
                }
            }
        }
        cout<<endl;
    }
    return 0;
}
```

3. 解题思路：

这道题看的时候感觉很乱，读了很多遍才明白意思。首先要明确需要保存到变量的值是输入时他们的下标，因此要把输入的身高复制到另一个数组

（remainboy 和 remaingirl）里，防止排序之后他们原先的标号丢失。又因为每个身高的序号在下标中的，使用起来不是很方便，所以再把所有的下标保存到另外的数组 numboy 和 numgirl 中，而这些下标组成的数组的异性（另一个，比如 numboy 的异性数组就是 numgirl）数组的成员的下标就是要输出的对应舞伴的位置。这个解法好像有点暴力……不过思路还是比较清楚的。

4. 易错点：
   解题的时候不能太着急，想不清楚的写在纸上，想不清楚的写在纸上，想不清楚的写在纸上。这很重要。

四、D 题

1. 题目代号与测评记录序号：{D247518}
2. 代码：

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
struct Student
{
    char name[13];
    int id;
    double grade;
};

//int cmp(const void*,const void*);
bool cmps(Student a,Student b);
int main()
{
    int n;
    while(scanf("%d",&n)!=EOF){
        Student student[n];
        for(int i=0;i<n;i++){
            scanf("%s %d %lf",student[i].name,&student[i].id,&student[i].grade);
        }
        //sort方法排序
        //qsort(student,n,sizeof(student[0]),cmp);
        std::sort(student,student+n,cmps);

        //查找
        int m;
        int findTheId=0;
        scanf("%d",&m);
        for(int i=0;i<m;i++){
            scanf("%d",&findTheId);
            bool flag=false;
            for(int j=0;j<n;j++){
                if(student[j].id==findTheId){
                    flag=true;
                    printf("%s %d %.2f %d\n",student[j].name,student[j].id,student[j].grade,j+1);
                    break;
                }
            }
            if(!flag){
                printf("Only god knows where he is.\n");
                //break;服了
            }
        }
    }
    return 0;
}

/*
int cmp(const void *a,const void *b)
{
    Student *c=(Student *)a;
    Student *d=(Student *)b;
    if(c->grade==d->grade){
        return c->id-d->id;//升序
    }
    else{
        return d->grade-c->grade;//降序
    }
}
*/
bool cmps(Student a,Student b)
{
    if(a.grade==b.grade){
        return a.id<b.id;
    }
    else{
        return a.grade>b.grade;
    }
}
```

3. 解题思路
   这道题是一道结构体排序题，思路不是很复杂，就是先用结构体数组保存每个同学的各个信息，要注意姓名不多于 12 个字符，再加上'\0'总共需要 13 个位置。然后排序和查找。这道题主要问题在于时间，如果用两个 for 循环写排序的话时间肯定会超。因此上机之后又学习了 sort 函数和 qsort 函数排序，之后再用遍历查找就可以了。
4. 易错点：
   多学一些常用的函数写代码的效率会好很多。
5. 题目延伸（sort 函数和 qsort 函数总结）：
   这两个函数使用起来还是 sort 顺手一些，但是 qsort 比 sort 运行快一些。sort 属于名字空间 std，可以对数组来用，也可以对 array 对象和 vector 对象使用，对数组（假设数组 a 有 n 项）的话就是 sort(a,a+n,cmp)，a 和 a+n 很好理解，cmp 是一个比较函数，如果想让数组升序，直接 sort(a,a+n)就可以了，如果是降序或者其他有条件的排序（比如此题），就要写 cmp 函数了，cmp 返回一个 bool 值，值是 1 的话代表这两个数不交换，是 0 则交换位置。所以升序写成 return a<b，降序写成 return a>b。qsort 是 stdlib.h 中的函数，qsort(a,n,sizeof(a[o]),cmp)需要四个参数，分别是数组名称、数组的成员个数、每个成员所占的字节数以及 cmp 函数。不过这个 cmp 函数更不好理解一些，它返回 int 类型，参数是两个 const 空类型指针，即 const void*，而且在 cmp 中无法直接使用空类型指针，必须先强制类型转换，把 void*变成结构体声明的类型的指针（Student*），而 cmp 函数返回正数就是说 cmp 传入参数第一个要放在第二个后面，负数就是传入参数第一个要放第二个前面。所以升序就是 return a>b，降序就是 return a<b。这就是 sort 和 qsort 函数，如果这道题用两个 for 循环排序就会超时。

五、E 题

1. 题目代号与测评记录序号：{E248934}
2. 代码：

```c
#include <stdio.h>
int solution(int m,int n,int c[][101])
{
    if(m<1||n<1){
        return 0;
    }
    else if(c[m][n]!=0){
        return c[m][n];
    }
    else if(m==1||n==1){
        return 1;
    }
    else if(m<n){
        return solution(m,m,c);
    }
    else if(m==n){
        return solution(m,n-1,c)+1;
    }
    else{
        return solution(m,n-1,c)+solution(m-n,n,c);
    }
}

int main()
{
    int t,m,n,c[101][101]={0};
    scanf("%d",&t);
    for(int i=1;i<101;i++){
        for(int j=1;j<101;j++){
            c[i][j]=solution(i,j,c);
        }
    }
    for(int i=0;i<t;i++){
        scanf("%d %d",&m,&n);
        printf("%d\n",c[m][n]);
    }
    return 0;
}
```

3. 解题思路：

首先要理解题意，可以当作分拆数来理解，把整数 m 拆分成 n 个非负整数之和，然后写出递推公式，可以这样理解：①人数最少的教室没人去，剩下 m 个人去 n-1 个教室；②人数最少的教室有一个人，这样的话每个教室都要先有一个人去，即 m 个教室总共有 m 个人，剩下就是 m-n 个人去 n 个教室。因此写出的递推公式是 s[m][n]=s[m][n-1]+s[m-n][n]，从而写出递归的最一般情况，也就是 m>n 的情况。然后再写出其他情况的递推公式。这样写出的程序答案已经是对的了，但是会超时。因为 m 和 n 到时 1 到 100 的整数，比如

纯递归方法下 m 和 n 都输入 100 大约需要 1.5s 才能输出结果，已经超时。比如求 s(7,3)需要 s(7,2)的值，而 s(7,2)的值还要去递归求解，这就很浪费时间，我的思路是从小到大把每个值保存到二维数组里这样，这样比如求解 s(7,3)递归到 s(7,2)的时候，就可以直接使用数组 c[7][2]的值了，做个比较，不使用数组的递归，需要递归 14 次才能求得 8，而使用数组只需要递归一次到 s(7,2)+s(4,3)就能使用 c[7][2]+s[4][3]求解。用嵌套的循环给 10000 个数组赋值看起来时间复杂度是 O(n^2)，但其实就是 10000 次赋值操作，瞬间就能完成。

4. 调试经验：

   超时的题目需要找到费时的地方，一般递归和迭代，都是大数据量的时候很浪费时间。遇到大数据量不要使用时间复杂度为 O(n^2)的嵌套循环。而就本题而言，递归可以保存节点到数组中，这样就可以一次递归得出答案。

六、F 题