第二次上机解题报告

15081070 张雨任

一、A 题

1. 题目代号与评测记录序号：{A252420}
2. 代码：

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

typedef struct _sqlist
{
    int data[500];
    int length;
}sqlist;
int main()
{
    int n;
    while(cin>>n){
        sqlist mylist;
        mylist.data[500]={0};
        mylist.length=0;
        string op;
        while(n--)
{
        cin>>op;
        if(op=="sum"){
            int sum=0;
            if(mylist.length==0)cout<<"invalid operation!\n";
            else{
                for(int i=0;i<mylist.length;i++){
                    sum+=mylist.data[i];
                }cout<<sum<<"\n";
            }
        }
        else if(op=="ins"){
            int x,pos;
            cin>>x>>pos;
            if(mylist.length==500||pos<0)cout<<"invalid operation!\n";
```

```
34              else if(pos>mylist.length)cout<<"invalid operation!\n";
35              else{
36                  for(int i=mylist.length;i>=pos;i--){
37                      mylist.data[i+1]=mylist.data[i];
38                  }
39                  mylist.data[pos]=x;
40                  mylist.length++;
41              }
42          }
43          else if(op=="del"){
44              int x;cin>>x;
45              if(mylist.length==0)cout<<"invalid operation!\n";
46              else{
47                  bool flag=true;int pos;
48                  for(int i=0;i<mylist.length;i++){
49                      if(x==mylist.data[i]){
50                          flag=false;
51                          pos=i;
52                          break;
53                      }
54                  }
55                  if(flag)cout<<"invalid operation!\n";
56                  else{
57                      mylist.length--;
58                      for(int i=pos;i<mylist.length;i++){
59                          mylist.data[i]=mylist.data[i+1];
60                      }
61                  }
62              }
63          }
64  }
65      }
66  }
```

3. 解题思路：

这道题考察顺序表的插入、删除和求和，其实就是对数组进行操作，要注意每次操作还要对长度 length 加减。顺序表为空不能求和，表满或者插入位置大于表长或者小于 0 不能插入，表为空或者输入的值不在顺序表中不能删除。插入删除要注意操作数组成员的顺序，插入：先把插入位置的值赋给后一位再插入新值，删除：直接后边的值赋给前面的。

二、B 题

1. 题目代号与评测记录序号：{B252384}
2. 代码：

```cpp
#include <iostream>
using namespace std;
struct SqList{
    int val;
    SqList *next;
};
int main()
{
    int n;
    while(cin>>n){
        SqList *L,*r,*s;
        L=new SqList;
        r=L;
        for(int i=0;i<n;i++){
            s=new SqList;
            cin>>r->val;
            r->next=s;
            r=s;
        }
        r->next=nullptr;
        r=L;
        for(int i=0;i<n;i++){
            cout<<r->val<<" ";
            r=r->next;
        }cout<<endl;
        r=L;
        SqList *p,*q;
        p=L;
        q=p->next;
        while(p->next!=nullptr){
            if(p->val==q->val&&q->next!=nullptr){
                SqList *m=q;
                p->next=q->next;
                q=q->next;
                delete m;
            }
            else if(p->val==q->val&&q->next==nullptr){
                p->next=nullptr;delete q;
            }
            else if(p->val!=q->val){
                p=p->next;q=q->next;
            }
        }
        p=L;
        while(p->next!=nullptr){
            cout<<p->val<<" ";
            p=p->next;
        }cout<<endl;
    }
}
```

3. 解题思路：
   考察链表的基础操作，首先创建链表，创建链表要注意是否有头结点，最后一个节点的 next 为空。然后通过指针的移动比较前后两个节点 data 值是否相等，相等则删去后面的值。要注意使用这种方法要考虑到最后一个节点和倒数第二个节点比较相等的话，直接使倒数第二个节点的 next 为空，再删除该节点即可。链表没有顺序表直观，容易出错。想不清楚可以在纸上画一画。
4. 易错点：
   创建链表要让最后一个结点的 next 为空，创建的时候要记得用 new 分配空间，然后删除结点要记得用 delete 释放空间。要读题，题目要求去重之前的链表也要输出。

三、C 题

1. 题目代号与评测记录序号：{C254147}
2. 代码：

```cpp
#include <iostream>
#include <string>
using namespace std;
struct _words
{
    string data;
    int spaceAfter=1;
    bool enter=false;
}words[250];
int main()
{
    unsigned int n;
    cin>>n;
    cin.ignore();
    unsigned int cntr=0;
    while(cin>>words[cntr].data){
        cntr++;
    }
    unsigned int numOfWords=cntr;
    unsigned int lengthPerLine=0;
    unsigned int lengthPerLineNoSpace=0;
    cntr=0;
    unsigned int start=0;
    unsigned int wordsPerLine=0;
    for(unsigned int i=0;i<numOfWords;i++){
        cntr++;
        lengthPerLine+=(words[i].data.size()+1);
        lengthPerLineNoSpace+=words[i].data.size();
        if(lengthPerLine+words[i+1].data.size()>n){
            if(cntr==1){
                words[i].enter=true;
            }
            else{
                if(cntr==2){
                    words[i-1].spaceAfter=n-words[i].data.size()-words[i-1].data.size();
                }
                else{
                    words[i-1].spaceAfter=n-lengthPerLineNoSpace-(n-lengthPerLineNoSpace)/(cntr-1)*(cntr-2);
                    for(unsigned int j=start;j<i-1;j++){
                        words[j].spaceAfter=(n-lengthPerLineNoSpace)/(cntr-1);
                    }
                }
                words[i].enter=true;
            }
            start=i+1;
            cntr=0;
            lengthPerLine=0;
            lengthPerLineNoSpace=0;
        }
    }
    for(unsigned int i=0;i<numOfWords-1;i++){
        cout<<words[i].data;
        if(words[i].enter)cout<<"\n";
        else{
            for(unsigned int j=0;j<words[i].spaceAfter;j++)cout<<" ";
        }
    }
    cout<<words[numOfWords-1].data;
}
```

3. 解题思路：
   这道题坑点很多，要考虑很多特殊情况。我的思路如下。首先声明一个结构体数组，用来存储每个单词的三个信息：string 类存储单词本身（data），一个整型值存储这个单词后面的空格数量（spaceAfter），默认值为 1 个空格，一个布尔类型来表示这个单词后面是否需要换行（enter），也就是这个单词是否是非最后一行的最后一个单词，false 代表不用换行，true 代表需要换行。然后输入录入每个单词，之后用循环遍历每个单词，用 lengthPerLine 这一行累计每个单词（带空格）所占的宽度，用 lengthPerLineNoSpace 代表不计空格的情况下改行累计的字符宽度，如果包括空格加上下一单词超过 n（每行的字符宽度），就进入分配这一行每个单词后面空格数的算法。Cntr 代表这一行累计的单词数量，每次换行要清空这个计数器。如果 cntr 为 1，说明这一行只有一个单词，加上下一个单词就会超过 n，此时该单词之后的空格数就不重要了，把他的 enter 值置为 true 即可，因为他既是这一行的而第一个单词，也是最后一个。如果 cntr 不是，也要分两种情况，cntr 为 2，cntr 不为 2。Cntr 为 2 说明这一行只有两个单词，那么出去两个单词所占用的字符宽度以外，剩下的宽度都是这一行第一个单词后面的空格，直接把这个值赋给它的 spaceAfter 就可以了。如果 cntr 既不是一也不是二，说明这一行超过两个单词，那么直接把总空格数除以空位置的数量（这一行单词数减一）就是除去最后两个单词前面单词的 spaceAfter 的值，单数第二个单词就得到剩下的所有空格，最后一个单词没有空格，把它的 enter 值置为 true 即可。之后还要把一些计数器重置，start 代表这一行第一个单词在所有单词中的位置，用来给这一行每个单词的 spaceAfter 赋值，把它置为下一行第一个单词的位置，把累计每行单词数量的计数器 cntr、累计每行带空格和不带空格的字符宽度的两个计数器都置为 0，就可以开始下一次循环了。这个循环结束后就可以输出了，遍历结构体数组的每个成员，输出该单词（data），如果 enter 为 true，则在后面输出换行，否则通过循环输出空格的个数。最后一个单词单独输出，因为它既不用换行，后面也没有空格。
4. 易错点：
   首先，输入的时候遇到了麻烦，在想怎么识别换行符，其实就是遇到 EOF 停止输入就可以了。这道题需要考虑的特殊情况比较多，调试的时候出现了除零异常，原因是 cntr-1 做了分母，因此 cntr=1 是单独的一种情况。再由于每一行最后两个单词后面的空格数是不同于之前的，所以要多考虑每行只有一个单词和只有两个单词的情况。还有个小 bug 调了很久。因此不能依赖输入输出样例，还是要自己一行一行的调试，关注每个值的状态，一行行读静态代码。

四、D 题

1. 题目代号与测评记录序号：{D252743}
2. 代码：

```
#include <stdio.h>
int app[100050],start[100050],sq[100050];
int main()
{
    int n,m,k;
    while(scanf("%d",&n)!=EOF){
        scanf("%d %d",&m,&k);
        for(int i=1;i<=n;i++){
            scanf("%d",&app[i]);
            sq[app[i]]=i;
        }
        for(int i=1;i<=m;i++){
            scanf("%d",&start[i]);
        }
        long cntr=0;
        for(int i=1;i<=m;i++){
            if(sq[start[i]]==1){
                cntr++;
            }
            else{
                if(sq[start[i]]%k!=0)cntr+=(sq[start[i]]/k+1);
                else cntr+=sq[start[i]]/k;
                int temp=app[sq[start[i]]];
                app[sq[start[i]]]=app[sq[start[i]]-1];
                app[sq[start[i]]-1]=temp;
                sq[app[sq[start[i]]]]++;
                sq[app[sq[start[i]]-1]]--;
            }
        }
        printf("%ld\n",cntr);
    }
}

```

3. 解题思路：
   这道题是一道查找题，但是在查找之余还要对数组的顺序进行改变，因此我并不能想出怎么使用先排序再二分查找的方法。如果使用遍历查找就会超时，时间复杂度达到了 O(n^2)，最多要循环 10^10 次。所以我想到了使用两个数组，app[]下标代表程序的次序，值代表程序的代号，sq[]下标代表程序的代号，值代表程序的次序。这样在输入了要启动的程序代号就不用遍历了，直接用 sq 数组找到对应的程序的次序，改变代表操作次数的累加器的值，再用 app 数组交换刚刚找到的次序和这个次序值减一的值对应的程序代号。更新完 app 数组再更新 sq 数组，由于 app 数组的两个程序代号已经交换了值，所以下标为原来 app 数组的值的 sq 数组的值其实是另一个 app 数组对应的值的 sq 数组对应的值，所以相应的要进行自加和自减。两个驻足更新后就可以惊醒下一次循环了。时间复杂度 O(n)
4. 易错点：

首先在于想法，原来 O(n^2)算法超时，但是外面的循环是避免不了的，所以要想办法去掉里面的循环。本来想用二分查找，但是查找之后还要对原数组进行操作，所以这个方法也不行。所以用了这种既不用遍历又可以保存数组不被破坏的方法。之后在于数组套数组很容易出错，所以要仔细得想清楚下标和值的关系。最后有一个小 bug 查了很久，就是计数器的值会在 int 范围之外，因为 m 和 n 最大可以取到 10^5，因此计数器的值可以达到 10^10 以上，不在 int 范围内，要用 long，这是一个坑点，以后要多读题多用 long。

六、F 题

1. 题目代号与测评记录序号：{F255562}
2. 代码：

```
#include <stdio.h>
long matrix[1002][1002];
int a[10000],b[10000],c[10000],d[10000],h[10000],w[10000];
int main()
{
    long n,m,q;
    while(scanf("%ld",&n)!=EOF){
        scanf("%ld",&m);
        scanf("%ld",&q);
        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++){
                scanf("%ld",&matrix[i][j]);
            }
        }
        for(int i=0;i<q;i++){
            scanf("%d %d %d %d %d %d",&a[i],&b[i],&c[i],&d[i],&h[i],&w[i]);
        }
        for(int k=0;k<q;k++){
            for(int i=a[k];i<=a[k]+h[k]-1;i++){
                for(int j=b[k];j<=b[k]+w[k]-1;j++){
                    int temp=matrix[i][j];
                    matrix[i][j]=matrix[i+c[k]-a[k]][j+d[k]-b[k]];
                    matrix[i+c[k]-a[k]][j+d[k]-b[k]]=temp;
                }
            }
        }
        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++){
                printf("%ld ",matrix[i][j]);
            }
            printf("\n");
        }
    }
}
```

3. 解题思路：

这道题没 a，超时了，但是还是稍微写一下超时的思路吧。因为是矩阵，所以建立了一个二维数组，遍历二维数组的每个值并且交换对应位置的两个值。这个算法本身没有问题，但是时间复杂度很恐怖。这道题暂时只能想出来这个方法，之后还要好好考虑别的算法。我认为是因为这种二维数组只能整行整列的遍历，如果把算法改成可以随意的遍历，例如从 matrix[3][4]可以向下也可以向右的话，就可以降低时间复杂度。但是还没能实现。