7-2 jmu-ds-顺序表区间元素删除 (15 分)

若一个线性表L采用顺序存储结构存储，其中所有的元素为整数。设计一个算法，删除元素值在[x,y]之间的所有元素，要求算法的时间复杂度为O(n)，空间复杂度为O(1)。

输入格式:

三行数据，第一行是顺序表的元素个数，第二行是顺序表的元素，第三行是x和y。

输出格式:

删除元素值在[x,y]之间的所有元素后的顺序表。

输入样例:

```
10
5 1 9 10 67 12 8 33 6 2
3 10
```

输出样例:

```
1 67 12 33 2
```

```
#include<stdio.h>
#include<stdlib.h>
#define OVERFLOW -1
#define OK 1
#define ERROR 0
#define LIST_ININ_SIZE 100
#define LISTINCREMENT 10

typedef int status;
typedef int Element;

typedef struct {
    Element* elem;
    status length;
    status listsize;
}Sqlist;

status InitList_Sq(Sqlist *L) {
    L->elem = (Element *) malloc(LIST_ININ_SIZE * sizeof(Element*));
    if (!L->elem) {
        exit(OVERFLOW);
    }
    L->length = 0;
    L->listsize =LIST_ININ_SIZE;
    return OK;
}


int main(){
    Sqlist list,newList;
    status x,y;
```

```
if(scanf("%d",&list.length)==1){

}else{
    printf("错误");
}

list.elem=(Element*)malloc(list.length*sizeof(Element*));
if(!list.elem) exit(OVERFLOW);
newList.elem=(Element*)malloc(list.length*sizeof(Element*));
if(!newList.elem) exit(OVERFLOW);
for(status i=0;i<list.length;i++){
    if(scanf("%d ",&list.elem[i])==1){

  }else{
    printf("错误");
  }


}



if(scanf("%d",&x)==1){

}else{
    printf("错误");
}

 if(scanf("%d",&y)==1){

}else{
    printf("错误");
}

newList.length=0;
status p=0;
for(status k=0;k<list.length;k++){
    if(list.elem[k]<x||list.elem[k]>y){
        newList.elem[p++]=list.elem[k];
        newList.length++;
    }

}

list.elem=newList.elem;
for(status m=0;m<newList.length;m++){
    if(m==p-1){
```

```
            printf("%d",list.elem[m]);
        }else{
             printf("%d ",list.elem[m]);
        }
    }

    return 0;

}
```