第五章　完善程序

第1节　数组

1.【NOIP2013】序列重排

全局数组变量a定义如下：const int SIZE=100;int a[SIZE],n;它记录着一个长度为n的序列a[1],a[2],…,a[n]。

现在需要一个函数，以整数p(1≤p≤n)为参数，实现如下功能：将序列a的前p个数与后n−p个数对调，且不改变这p个数(或n−p个数)之间的相对位置。例如，长度为5的序列1,2,3,4,5，当p=2时重排结果为3,4,5,1,2。

有一种朴素的算法可以实现这一需求，其时间复杂度为O(n)、空间复杂度为O(n)：

```
void swap1(intp){
    int i,j,b[SIZE];
    for (i=1;i<=p;i++)
        b[  ①  ]=a[i];
    for (i=p+1;i<=n;i++)
        b[i-p]=  ②  ;
    for (i=1;i<=  ③  ;i++)
        a[i]=b[i];
}
```

我们也可以用时间换空间，使用时间复杂度为$O(n^2)$、空间复杂度为O(1)的算法：

```
void swap2(intp){
    int i,j,temp;
    for (i=p+1;i<=n;i++){
        temp=a[i];
        for (j=i;j>=  ④  ;j--)
            a[j]=a[j-1];
          ⑤  =temp;
    }
}
```

选择题

(1)①处应填(　B　)。

A. p+i　　B. n-p+i

C. n-p+i-1　　D. i

(2)②处应填(　A　)。

A. a[i]　　B. a[i-p]

C. a[n-i]　　D. b[i]

(3)③处应填(　A　)。

A. n　　B. p

C. p+1　　D. n-1

(4)④处应填(　B　)。

A. 1　　B. i-p

C. n-i　　D. i-p+1

(5)⑤处应填(　C　)。

A. a[i-p+1]　　B. a[i]

C. a[i-p]　　D. a[i-1]

2.【NOIP2014】最大子矩阵和

给出 m 行 n 列的整数矩阵，求最大的子矩阵和(子矩阵不能为空)。

输入第一行包含两个整数 m 和 n，即矩阵的行数和列数。之后 m 行，每行 n 个整数，描述整个矩阵。程序最终输出最大的子矩阵和。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int matrix[SIZE+1][SIZE+1];
int rowsum[SIZE+1][SIZE+1];
//rowsum[i][j]记录第 i 行前 j 个数的和
int m,n,i,j,first,last,area,ans;
int main(){
    cin>>m>>n;
    for (i=1;i<=m;i++)
        for (j=1;j<=n;j++)
            cin>>matrix[i][j];
    ans=matrix[①][①];
    for (i=1;i<=m;i++)
        ②;
    for (i=1;i<=m;i++)
    for (j=1;j<=n;j++)
        rowsum[i][j]=③;
for (first=1;first<=n;first++)
    for (last=first;last<=n;last++){
        ④;
        for (i=1;i<=m;i++){
            area+=⑤;
            if (area>ans)ans=area;
            if (area<0)area=0;
        }
    }
cout<<ans<<endl;
return 0;
```

选择题

(1)①处应填(　　)。

A. [0][0]　　B. [0][n]

C. [m][0]　　D. [1][1]

(2)②处应填(　　)。

A. rowsum[i][0]=matrix[i][1]　　B. rowsum[i][0]=1

C. rowsum[i][0]=0　　D. rowsum[i][0]=ans;

(3)③处应填(　　)。

A. rowsum[i][j-1]　　B. rowsum[i-1][j]+matrix[i][j]

C. matrix[i][j]　　D. rowsum[i][j-1]+matrix[i][j]

(4)④处应填(　　)。

A. area=0　　B. ans=area

C. area=ans　　D. ans=0

(5)⑤处应填(　　)。

A. matrix[i][last]　　B. rowsum[i][last]-rowsum[i][first-1]

C. matrix[i][fist]　　D. rowsum[i][last]-rowsum[i][first]

3.【NOIP2008】矩阵中的数字

有一个 n * n(1≤n≤5000)的矩阵 a，对于 1≤i<n，1≤j≤n，a[i][j]<a[i+1][j]，a[j][i]<a[j][i+1]。即矩阵中左右相邻的两个元素，右边的元素一定比左边的大。上下相邻的两个元素，下面的元素一定比上面的大。给定矩阵 a 中的一个数字 k，找出 k 所在的行列(注意：输入数据保证矩阵中的数各不相同)。

```
#include<iostream>
using namespace std;
int n,k,answerx,answery;
int a[5001][5001];
void FindKPosition()
{
    int i=n,j=n;
    while (j>0)
    {
        if (a[n][j]<k) break;
        j--;
    }
    ①
    while (a[i][j]!=k)
    {
        while (②&&i>1)i--;
        while (③&&j<=n)j++;
    }
    ④;
    ⑤
}
int main()
{
    int i,j;
    cin>>n;
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            cin>>a[i][j];
    cin>>k;
    FindKPosition();
    cout<<answerx<<""<<answery<<endl;
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. j--　　B. j++

C. i++　　D. i--

(2)②处应填(　　)。

A. a[i][j]>k　　B. a[i][j]<k

C. a[i][j]<=k　　D. a[i][j]!=k

(3)③处应填(　　)。

A. a[i][j]>k　　B. a[i][j]<k

C. a[i][j]>=k　　D. a[i][j]!=k

(4)④处应填(　　)。

A. answerx=i+1　　B. answerx=i-1

C. answerx=j　　D. answerx=i

(5)⑤处应填(　　)。

A. answery=j+1　　B. answery=j-1

C. answery=i　　D. answery=j

第2节 字符处理

1.【NOIP2008】字符串替换

给定一个字符串 S(S 仅包含大小写字母),下面的程序将 S 中的每个字母用规定的字母替换,并输出 S 经过替换后的结果。程序的输入是两个字符串,第一个字符串是给定的字符串 S,第二个字符串 S'由 26 个字母组成,它是 a—z 的任一排列,大小写不定,S'规定了每个字母对应的替换字母:S'中的第一个字母是字母 A 和 a 的替换字母,即 S 中的 A 用该字母的大写替换,S 中的 a 用该字母的小写替换;S'中的第二个字母是字母 B 和 b 的替换字母,即 S 中的 B 用该字母的大写替换,S 中的 b 用该字母的小写替换;……以此类推。

```
#include<iostream>
#include<string.h>
char change[26],str[5000];
using namespace std;
void CheckChangeRule(){
    int i;
    for (i=0;i<26;i++)
    {
        if (  ①  )
            change[i]-='A'-'a';
    }
}
voidChangeString(){
    int i;
    for (i=0;i<strlen(str);i++)
    {
        if (  ②  )
            str[i]=change[str[i]-'A']-'a'+'A';
        else
          ③
    }
}
int main()
{
    int i;
    cin>>str;
    cin>>change;
    CheckChangeRule();
    ④
    cout<<str<<endl;
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. change[i]>='a'&&change[i]<='z'　　B. change[i]<'A'||change[i]>'Z'

C. change[i]>='A'&&change[i]<='Z'　　D. change[i]<'a'&&change[i]>'z'

(2)②处应填(　　)。

A. str[i]>='a'&&str[i]<='z'　　B. str[i]<'A'&&str[i]>'Z'

C. str[i]<'a'&&str[i]>'z'　　D. str[i]>='A'&&str[i]<='Z'

(3)③处应填(　　)。

A. change[str[i]-'a']=str[i];　　B. str[i]=change[str[i]-'a'];

C. str[i]=change[str[i]-'A'];　　D. change[str[i]-'z']=str[i];

(4)④处应填(　　)。

A. int len=strlen(str);　　B. Changestring();

C. ChangeString();　　D. changeString();

2.【NOIP2016】读入整数

请完善下面的程序,使得程序能够读入两个 int 范围内的整数,并将这两个整数分别输出,每行一个。

输入的整数之间和前后只会出现空格或者回车。输入数据保证合法。

例如:

输入:

123

-789

输出:

123

-789

```
#include<iostream>
using namespace std;
int readint(){
```

```
    int num=0;              //存储读取到的整数
    int negative=0;         //负数标识
    char c;                 //存储当前读取到的字符
        c=cin.get();
    while ((c<'0'||c>'9')&&c!='-')
        c=  ①  ;
    if (c=='-')
        negative=1;
    else
        ②  ;
    c=cin.get();
    while (  ③  ){
        ④  ;
        c=cin.get();
    }
    if (negative==1)
        ⑤  ;
    return num;
}
int main(){
    int a,b;
    a=readint();
    b=readint();
    cout<<a<<endl<<b<<endl;
    return 0;
}
```

选择题

(1)①处应填(　　)。

A. c-'0'　　B. '0'

C. c+'0'　　D. cin.get()

(2)②处应填(　　)。

A. num=0　　B. num=c-'0'

C. num=c-'a'　　D. num=c

(3)③处应填(　　)。

A. c>='0'&&c<='9'　　B. c>='a'&&c<='z'

C. c<'0'||c>'9'　　D. c!='-'

(4)④处应填(　　)。

A. num=num+c-'0'　　B. num=num*10+c-'0'

C. num=num+c　　D. num=num*10+c

(5)⑤处应填(　　)。

A. num=-num　　B. num=num+num

C. num--　　D. num++

3.【NOIP2009】最大连续子段和

给出一个数列(元素个数不多于100)，数列元素均为负整数、正整数、0。请找出数列中的一个连续子数列，使得这个子数列中包含的所有元素之和最大，在和最大的前提下还要求该子数列包含的元素个数最多，并输出这个最大和以及该连续子数列中元素的个数。例如数列为4 −5 3 2 4时，输出9和3；数列为1 2 3 −5 0 7 8时，输出16和7。

```
#include<iostream>
using namespace std;
int a[101];
int n,i,ans,len,tmp,beg;
int main(){
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>a[i];
    tmp=0;
    ans=0;
    len=0;
    beg=__①__;
    for (i=1;i<=n;i++){
        if (tmp+a[i]>ans){
            ans=tmp+a[i];
            len=i-beg;
        }
        else if (__②__&&i-beg>len)
            len=i-beg;
        if (tmp+a[i]__③__){
        beg=__④__;
        tmp=0;
    }
    else
        __⑤__;
}
cout<<ans<<""<<len<<endl;
return 0;
}
```

●选择题

(1)①处应填(　　)。

A. 1　　B. 0

C. n　　D. 101

(2)②处应填(　　)。

A. tmp+a[i]>ans　　B. tmp+a[i]=ans

C. tmp+a[i]>n　　D. tmp+a[i]==ans

(3)③处应填(　　)。

A.<0　　B. <ans

C. <mp　　D. <min(0,ans,tmp)

(4)④处应填(　　)。

A. 1　　B. 0

C. n　　D. i

(5)⑤处应填(　　)。

A. tmp+ = a[i]　　B. tmp= i+ 1

C. beg= i　　D. beg= i+ 1