第二次书面作业

这次作业题面都比较长，但是代码量不大，~~希望训练大家阅读理解能力~~。

1.计算多边形内积

高中时我们学习过两个向量的内积；同样地，二维向量可以定义外积。

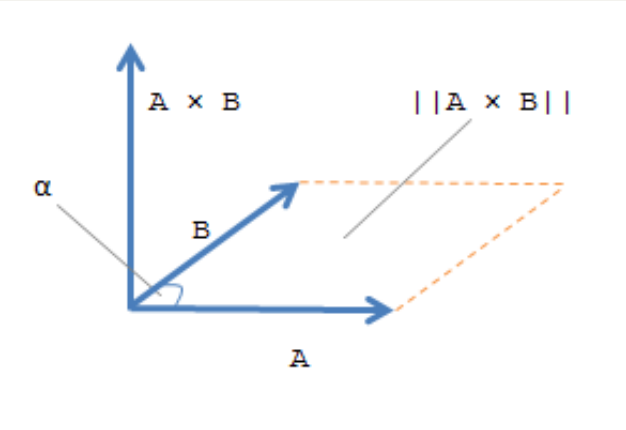
$$(x_1, y_1) \times (x_2, y_2) = x_1 y_2 - y_1 x_2$$

也叫做叉乘。可以看出，二维外积的结果是标量，不符合交换律，方向的变化会导致结果符号的变化。

类似于内积，容易得到一个结论：

$$\mathbf{a} \times \mathbf{b} = |a|\,|b| sin\,\langle a, b \rangle$$

正负号的判定上，如果a经过逆时针转向b，则外积大于0。



可以看出向量外积的一个几何意义，也就是**a和b之间张成的平行四边形的有向面积**；而外积除以2就是其张成的三角形的面积。这个性质可以帮助我们简单计算多边形面积。

现在，假设我们已经按照逆时针顺序给出了一个n边型的所有顶点，我们要求一个n边型的面积，请补全下述函数。

```
double area(int n,double x[],double y[]){
  //n是多边形的变数，第i个点（从1开始）的坐标为(x[i],y[i])
  double ans=0;
  ___(1)___
  ...
  _________
  return ans;
}
```

2.字符串翻转

（1）一个字符串为全为英文、符号、数字等单字节的字符时，我们可以很简单地将之翻转。一般地，我们要将一个序列从a到b位翻转过来，我们只需要以(a+b)/2为界，将两边对应位置的字符交换即可（注意，这个序列长度的奇偶性是无关紧要的，除以二的时候可以向下取整）。请补全这样的函数。

```
void reverse(char s[],int a,int b){
  //s是要操作的序列，希望将s[a]到s[b]翻转过来
  ___(1)___
  ...
  _________
}
```

这里希望操作的效果如下：

```
char t[10]="12345"
reverse(t,0,4);
```

在执行完这段代码后，t的内容是54321

（2）假如字符串内存在非ASCII字符，比如GBK编码的中文，那么会稍微麻烦一些。

这里简单讲一下GBK中文的编码，在GBK中，一个汉字占两个字节，而且第一个字节的最高位为1（也就是第一个字节是负数）。系统一个一个读取，读到负数时，就能判断它和下一个字节组成一个汉字，然后在字库中寻找。

例如，t[]="1测2试3"的长度为7，t[1]、t[4]的ASCII码值都是负数

请完整代码，将中英文混杂的字符串翻转。**这里你可以使用第一题的reverse函数。**

```
void reverse_GBK(char s[],int len){
  //s是一个字符串，len是它的长度（注意这里的长度是指字符占据的长度，不包括结尾的'\0'）
  ___(2)___
  ...
  _________
}
```

效果如下：

```
char t[]="1测2试3";
reverse_GBK(t,7);
```

结果t的值为 3试2测1

注意，不是所有显示终端都用GBK编码。大家可以使用devc++来测试。

3.前缀和

假设有如下的问题：

现有长度为n的数列a，有q个询问，每次询问都会给一对整数l<r,希望知道a[l]到a[r]中所有数的和。

面对这个问题，初学时，我们可能会对每一个询问从左到右循环一次，将区间里的数都累加起来。这样做的效率是很低下的。

高中在做数列题时，我们常常接触到前n项和的概念；借助这个概念，对每次的询问只需要一次减法就能知道询问的答案。

请根据提示，补全下列代码。

```
#include <stdio.h>

int a[1000],n,s[1000];

void init_prefix_sum(){
  //将a[]的前n项和求出，放到s[]中
  ____(1)_____
  ...
  ____________
}

int main(){
  int q,l,r;
  scanf("%d%d",&n,&q);
  for(int i=1;i<=n;i++) scanf("%d",&a[i]);//输入a数组
  init_prefix_sum();
  for(int i=1;i<=q;i++){
    scanf("%d%d",&l,&r);
    printf("a[&d] 到 a[&d]的和为%d\n",l,r,____(2)____);
  }
}
```

上面牵扯到的知识点叫做前缀和，是非常简单但是深刻的数据结构，它的思想是将区间问题转换成端点问题。

4.差分序列

继续考虑这么一个问题：有一个长度为n的序列a，q次操作，每次操作我们都希望将a[l]到a[r]里的每个数加上1。问在q次操作以后，数列里的每个值变为了多少?

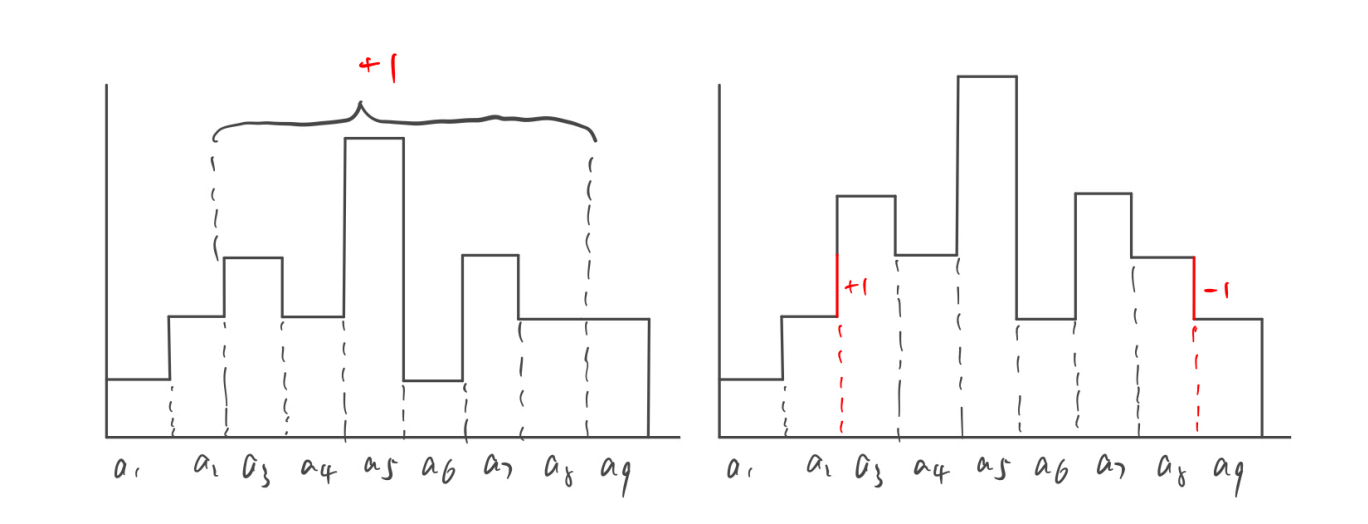
同样地，初学的时候，我们会对每次询问，都从l循环到r，对a[i]都执行加一的操作。这样的效率一样不高。

这里，我们介绍差分序列。什么叫做差分呢？简单来说，就是数组里面每个数和之前一个数的差。

具体来说，我们已知数组a,那么它的差分序列**s就定义为s[i]=a[i]-a[i-1]**。注意，这里的a从1开始，a[0]约定为0。

可以看出来，**差分序列s的前n项和，就是原数组a**。（这里就类似于求导和积分，大家可以仔细体会）也即是说，差分序列的前缀和就是原序列。

有了差分序列，我们如何进行整个区间加1的操作呢？这里有一幅图可以形象地表示。



这样，我们可以快速的执行每一次加一操作，最后再统计原来的序列变成了什么样子。

根据上面的提示，完成下面的代码。

```
#include <stdio.h>

int a[1000],n,s[1000];

void init_diff_seq(){
  //求出数组a的差分序列，存放到s中
  ___(1)___
  ...
  _________
}

void interval_add1(int l,int r){
  //操作差分序列，将区间[l,r]中的数都加1
  ___(2)___
  ...
  _________
}
```

```
int main(){
  int q,l,r;
  scanf("%d",&n);
  for(int i=1;i<=n;i++) scanf("%d",&a[i]);

  init_diff_seq();//根据输入的a，计算差分序列s

  scanf("%d",&q);
  for(int i=1;i<=q;i++){
    scanf("%d%d",&l,&r);
    interval_add1(l,r);//操作s数组，将a[l]到a[r]的值都加上1
  }

  //根据操作完的s数组，计算q次操作后得到的a数组。
  ___(3)___
  ...
  _________

  for(int i=1;i<=n;i++) printf("%d ",a[i]);
  return 0;
}
```