第二套试题

1.流量（tarifa.pas/c/cpp）

Pero和他的运营商签订了一个非常nice的流量套餐。运营商每个月给Pero XMB的上网流量，并且他当月没用完的流量会累积到到下个月继续使用。当然，Pero只能使用他实实在在拥有的流量。

如果我们能够提前预知他前N个月的流量使用情况，我们就可以知道在第N+1个月pero会有多少可支配的流量。

这听起来是不是有点意思，既然你这么感兴趣，那么就亲自动手帮Pero算一下吧。

第二套试题

1.流量（tarifa.pas/c/cpp）

【输入格式】

第1行：包含一个整数X（1≤X≤100），表示他每个月可以支配的流量；

第2行: 包含一个整数N（1≤N≤100），表示前N个月；

第3至n+2行：每行包含一个整数pi（0≤pi≤10，000），表示前N个月每个月所用的流量

【输出格式】

包含一个整数，表示他第N+1月能支配的流量。

【输入样例】

```
10
3
4
6
2
```

【输出样例】

```
28
```

第二套试题

1.流量（tarifa.pas/c/cpp）

思路：模拟每月扣除和累加流量的过程，先求出前N个月用掉的流量，然后拿总流量-已用流量即可。

```
for(int i=0; i<n; i++){
    cin>>a[i];
     sum += a[i];
  }

  cout<<x * (n + 1) - sum<<endl;
```

注意是第n+1个月，那么总流量是(n+1)*x

第二套试题

2.单词替换（dcth.pas/c/cpp）

输入一个字符串，以回车结束（字符串长度<=100）。该字符串由若干个单词组成，单词之间用一个空格隔开，所有单词区分大小写。现需要将其中的某个单词替换成另一个单词，并输出替换之后的字符串。

【输入格式】

第1行是包含多个单词的字符串 s;

第2行是待替换的单词a(长度 <= 100);

第3行是a将被替换的单词b(长度 <= 100);

s, a, b 最前面和最后面都没有空格。

【输出格式】

包含 1 行，将s中所有单词a替换成b之后的字符串。

第二套试题

2.单词替换（dcth.pas/c/cpp）

【输入样例】
You want someone to help you
You
I
【输出样例】
I want someone to help you

第二套试题

2.单词替换（dcth.pas/c/cpp）

思路：此题考察字符串基本操作；三个字符串s，s1，s2，首先是查找s1在s中的位置，然后替换为s2。

tips：可以各字符串前后加‘空格’，方便查找。

```
while(1)
{       p=s.find(s1,p);
        if(p==s.npos) break;
        s.replace(p,len1,s2);
}
```

第二套试题

3.排队拍照(photo.pas/c/cpp)

在一个美丽的景点，有N个同学想每人拍一张照片作为留念，于是他们就排好队一个一个来拍。但是呢，每个人拍照需要的时间是不同的，有些同学是在创作艺术，可能花费时间比较多，有些同学比较随意，只想证明自己到此一游而已。

考虑到排队排在后面的同学可能会等比较长的时间，为了让这个现象有所缓解，现在需要你来帮助他们找到一个排队的方案，使得所有人等待的时间总和最少。

第二套试题

3.排队拍照(photo.pas/c/cpp)

【输入格式】

第一行包含一个整数N（N≤50000），表示有N个同学想拍照；

第二行包含N个用空格隔开的整数，表示每个人拍照所需的时间T1,T2,……,Tn（0≤Ti≤100000）。

【输出格式】

输出文件 photo.out包含一个整数--所有人等待时间总和的最小值。

【输入样例】

5

2 3 1 5 4

【输出样例】

20

第二套试题

3.排队拍照(photo.pas/c/cpp)

【输入格式】

第一行包含一个整数N（N≤50000），表示有N个同学想拍照；

第二行包含N个用空格隔开的整数，表示每个人拍照所需的时间T1,T2,……,Tn（0≤Ti≤100000）。

【输出格式】

输出文件 photo.out包含一个整数--所有人等待时间总和的最小值。

【输入样例】

5

2 3 1 5 4

【输出样例】

20

第二套试题

3.排队拍照(photo.pas/c/cpp)

思路：运用贪心的思想，拍照用时最短的先拍，用时长的后拍，这样能够达到总时长最短的目的。

```
sort(a+1,a+n+1);//对拍照时间进行排序                                    ;
b[1]=0;
for(i=2;i<=n;i++)
{
        b[i]=b[i-1]+a[i-1]; //计算每个人等待时间
}
for(i=1;i<=n;i++)
  s=s+b[i];//等待时间进行累加
```

第二套试题

4.种花(color.pas/c/cpp)

在机房的生活是如此的寂寞，以至于以 will 为首的同志们只能够天天上农场种菜来打发时间。

msh 日复一日地种着她的玫瑰，will 则毫不疲倦地偷着他的花……尽管天天花被偷掉一半，msh 始终没有动摇她种花的决心。原来，一个宏伟计划的蓝图早已埋藏在她的心中。

众所周知，农场的花一共有 4 种颜色，msh 喜欢不喜欢老旧的东西，所以，她希望每天种花的方案都不一样。特别地，她会觉得两种一样颜色的花种在相邻的位置会很无聊。现在，她想知道，一共有多少种花的方案。

这里要注意的是，农场的种花的位置是不规则的。因此我们给出一对一对的相邻的位置的关系。

第二套试题

4.种花(color.pas/c/cpp)

【输入格式】

第一行两个数 N 和 M，表示种花的位置的个数和相邻的位置的对数

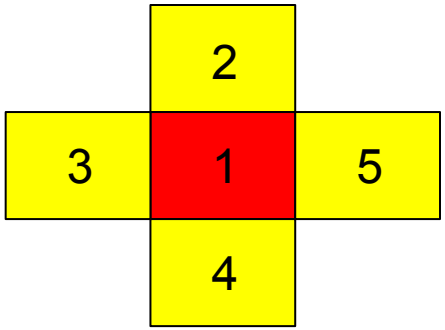
接下来 M 行，每行一组数 A，B 表示 A，B 相邻

【输出格式】

包含一个数，表示种花的方案数。

【输入样例】

5 4
1 2
1 3
1 4
1 5



【输出样例】

324

【数据范围】

40%的数据 N<=5

100%的数据 N<=10,M<=50

第二套试题

4.种花(color.pas/c/cpp)

思路：由题意可知，一共N个格子都要种满花，且有M种限制条件，由数据规模可以看到最多有10个位置，每个位置可以种4种颜色的花，因此最多有4^10种状态，这个数据规模很适合搜索。

搜索的状态为每个位置；

状态拓展为4种颜色的花；

每个状态判断一下当前选择的花的颜色是否与题意矛盾，如果不矛盾就继续穷举下一朵花，矛盾就停止；

结束状态为种满N个位置。

第二套试题

4.种花(color.pas/c/cp)

程序实现：

```
void DFS(int x) {
        if (x==n+1) {
                ++tmp;
                return;
        }//累加可行方案
        int c[5]={1,0,0,0,0};//拓展数组，必须在递归里面定义
        for (int i=1;i<=n;i++)
                if (s[x][i]) c[col[i]]=1;//判断与当前位置相邻的位置并标记
        for (int i=1;i<=4;i++) {
                if (!c[i]) {
                        col[x]=i;
                        DFS(x+1);
                        col[x]=0;//拓展可行颜色递归并回溯
                }
        }
}
```