天河 C 班选拔考试 01-20231027 题解

重要：每场比赛后，一定要复盘、订正！如果只是考完就算了，那就没有啥收获，还不如不做。任何科目的考试都应该这样。

本场比赛标程代码（建议参考学习的代码）的来源：

A：潘伟明
B：严子正
C：吕泽楷
E：黄旻哲
DF：zengfj

A

简单的 if 或 switch 语句即可解决。

```
#include<iostream>
using namespace std;
int main()
{
    string a;
    cin>>a;
    if(a=="tourist")cout<<3858;
    if(a=="ksun48")cout<<3679;
    if(a=="Benq")cout<<3658;
    if(a=="Um_nik")cout<<3648;
    if(a=="apiad")cout<<3638;
    if(a=="Stonefeang")cout<<3630;
    if(a=="ecnerwala")cout<<3613;
    if(a=="mnbvmar")cout<<3555;
    if(a=="newbiedmy")cout<<3516;
    if(a=="semiexp")cout<<3481;
    return 0;
}
```

B

一个简单的数学题。从 N 中减去 M，然后数有几个循环（向上取整）。

```cpp
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int n,m,k;
    cin >> n >> m >> k;
    if (n<k){
        cout << 0;
        return 0;
    }
    cout << (n-m)/k+1;


}
```

C

这也是一个很简单的题目，主要考察二维数组的读写。把 A_i、B_i、C_i、D_i 范围内的元素都标记一下（也即修改该范围内数组元素的值，方法很多，例如将元素值置为 1），最后再遍历一下整个二维数组，统计被标记元素的个数即可。

```cpp
#include<bits/stdc++.h>
using namespace std;
int arr[110][110];
int hasp[110][110];
int ans;
int main(){
    int n;
    cin>>n;
    for(int i=0;i<n;i++) for(int j=0;j<4;j++) cin>>arr[i][j];
    for(int i=0;i<n;i++){
        for(int x=arr[i][0];x<arr[i][1];x++){
            for(int y=arr[i][2];y<arr[i][3];y++){
                if(hasp[x][y]==0){
                    hasp[x][y]++;
                    ans++;
                }
            }
        }
    }
    cout<<ans;
    return 0;
}
```

D

这道题可以当作 26 进制数的转换来做，每一位上的数码乘以该位的权值，例如 BAD = B*26^2 + A*26^1 + D*26^0 = 2*26^2 + 1*26^1 + 4*26^0。

注意，一定不要用 cmath 里的 pow 函数来计算 26^i，因为 pow 函数返回的是 double 值，会涉及精度问题，WA 的可能性很大。解决办法：

（1） 手写自己的 pow 函数，用循环求出 x^y 的值。坏处：会产生很多重复计算（26^i、26^{i-1}、26^{i-2}……重复算了很多次）。

（2） 把算式进行拆分变形，在计算后面结果时，充分利用前面已经算出来的值，如下标程代码中的注释所示。

最后，要注意答案的范围是 10^{16}，因此要使用 long long 存储。

```cpp
#include<bits/stdc++.h>
using namespace std;

int main() {
    string s;
    cin >> s;
    int len = s.size();
    long long ans = s[0] - 'A' + 1;
    for (int i = 1; i <= len - 1; i++) {
        // 通过这句话，用累乘的方式来实现各位上 26^i 的求法。
        // 最左边的那位，需要乘上的 26 次数最多。
        // 即，比如计算 ABEC 的值，可以分解成计算 ((1*26 + 2)*26 + 5)*26 + 3，
        // 而上式是可以通过循环来递推计算的，也就用不到 pow 函数了。
        ans *= 26;
        ans += s[i] - 'A' + 1;
    }
    cout << ans;

    return 0;
}
```

E

回忆一下上课讲过的“计数排序”（A1115 直方图），用一个“桶”数组来记录哪些数出现了多少次。一块披萨，最多能被切 365 刀，那么也可以用一个数组来记录，操作过程中，哪些角度被切了一刀（下面代码中的 bool a[360]）。

注意到无论是顺时针还是逆时针转披萨，都不会影响最后的答案（最大块的饼的中心角度），所以不用特别考虑旋转方向。

标程代码中用变量 temp 记录当前旋转到的角度，那么如何保证 temp 值不超过 360？很简单，temp%=360。标程代码的第 13~18 行其实完全没有必要判断，直接模即可。

```cpp
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int n,temp=0,ls,sum,maxx=0;
    bool a[360] = {};
    cin >> n;
    for (int i=0;i<n;i++)
    {
        cin >> ls;
        temp+=ls;
        if (temp>359){
            temp%=360;
            a[temp]=true;
        }else{
            a[temp]=true;
        }
    }
    sum=0;
    for (int i=0;i<360;i++)
    {
        sum++;
        if (a[i]==true){
            if (sum>maxx)maxx=sum;
            sum=0;
        }
    }
    if (sum>maxx){
        maxx=sum+1;
    }
    cout << maxx;
}
```

F

本题不用太多的技巧和知识就能做出来，因为数据接受 $O(n^4)$ 复杂度的做法，主要考察大家会不会估计算法的时间复杂度，对 TLE 有一个大致的了解。

如果纯暴力做法，时间复杂度是 $O(n^9)$，只需要用到题里的两个条件，就能把时间复杂度降低 5 个数量级。具体做法参考下页的标程代码。

```cpp
#include <bits/stdc++.h>
using namespace std;
int H[3], W[3], ans = 0;
int main() {
  for (int i = 0; i < 3; i++) cin >> H[i];
  for (int j = 0; j < 3; j++) cin >> W[j];

  // 如果非常暴力的枚举所有情况，需要 9 重循环，计算量最大为 30^9，一定会TLE。
  // 但如果只是简单使用“三个参数之和已知”这个简单的条件，
  // 就能把循环嵌套次数降为 30^4，洒洒水啦
  for (int a = 1; a <= 30; a++) {
    for (int b = 1; b <= 30; b++) {
      for (int d = 1; d <= 30; d++) {
        for (int e = 1; e <= 30; e++) {
          int c = H[0] - a - b;
          int f = H[1] - d - e;
          int g = W[0] - a - d;
          int h = W[1] - b - e;
          int i = W[2] - c - f;
          // min({...})，C++11中的新写法，可以一次性求多个值的最大值。
          // 可以理解成 {} 里生成了一个临时数组，然后 min 函数循环求解该数组中的最大值。
          if (min({c, f, g, h, i}) > 0 && g + h + i == H[2])
              ans++;
        }
      }
    }
  }
  cout << ans;

  return 0;
}
```