2018国庆NOIP集训测试赛第五场（综合测试赛）

by $ZHAOJing$ zj@webturing.com

2018.10.05

A: a^b末尾数字：

- 难度：2
- 知识点: 快速幂 、二分、幂的周期性
- 参考解答：

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int a,b;
    while(cin>>a>>b) {
        a%=10;//保留最后一位
        b=b%4+4;//最后一位数的周期为4、注意到a^0=1所以要及时+4避免计算错误
        int s=1;
        while(b--) {//到这一步b已经不大于4，a不超过10直接模拟计算即可
            s*=a;
        }
        cout<<s%10<<endl;//结果再模10
    }
    return 0;
}
```

B: 逆波兰式求值

- 难度：4
- 知识点:栈的操作和字符串处理
- 参考解答：

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    stack<double> S;//结果为浮点数所以开浮点栈
    for(string tok; cin>>tok;) {//连续读入每一个符号
        istringstream iss(tok);//构造该符号的字符串输入流
        double x;
        if(iss>>x) {//如果该流对double x赋值成功则表示 tok就是一个数字
            S.push(x);//数值入栈
        } else {//赋值失败说明该符号是运算符，开始计算
            double b=S.top();S.pop();//弹出第二个运算数
```

```
                double a=S.top();S.pop();//弹出第一个运算数
                //以下模拟计算结果
                if(tok=="+")S.push(a+b);
                else if(tok=="-")S.push(a-b);
                else if(tok=="*")S.push(a*b);
                else S.push(a/b);
            }
        }
        cout<<fixed<<setprecision(2)<<S.top()<<endl;//格式化输出栈顶元素
        return 0;
    }
```

C: 奖学金

- 难度：3
- 知识点: 自定义排序
- 参考解答：

```
#include <bits/stdc++.h>
using namespace std;
struct Stu {//学生结构体
    int id;//编号
    int x,y,z;//分别代表语数外的成绩
    int total;//总分
    bool operator<(const Stu&that)const {//默认比较器实现,重载<运算符
        if(total-that.total) return total>that.total;//先按照总成绩降序
        if(x-that.x)return x>that.x;//相同的总分，按照语文成绩降序
        return id<that.id;//相同总分，相同语文成绩，按照学号降序
    }
};
int main() {
    int n;
    cin>>n;
    vector<Stu> S(n);//建立长度为n学生数组
    for(int i=0; i<n; i++) {
        S[i].id=i+1;//初始化学号
        cin>>S[i].x>>S[i].y>>S[i].z;//初始化成绩
        S[i].total=S[i].x+S[i].y+S[i].z;//初始化总分，为后续排序做好准备
    }
    sort(S.begin(),S.end());//开始排序了
    for(int i=0; i<5&&i<n; i++)//最多输出前5名
        cout<<S[i].id<<" "<<S[i].total<<endl;
    return 0;
}
```

D: 最优找零II

- 难度：4

- 知识点: 动态规划:状态转移方程为 $F_n = Min_{k=1}^{k\leq 5}(F_{n-b_k}+1)$
- 参考解答：

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin>>n;
    int B[5]= {1,2,4,5,10};//货币单位
    vector<int> F(n+1);//F[i]存储i元的最优找零个数
    for(int i=0; i<=n; i++)F[i]=i;//由于存在一元钱所以可以将F[i]初始化为i,i个1元就可以找零
    for(int i=0; i<5; i++)F[B[i]]=1;//如果是货币的单位显然可以用一张就可以找零
    for(int i=3; i<=n; i++) {//以下开始动态规划，自底向上计算每一个找零的方案
        int m=i;//i元最差可以用i个一元来找零
        for(int j=0; j<5; j++) {
            if(i<B[j])continue;
            m=min(F[i-B[j]]+1,m);
        }
        F[i]=m;//这些方案中的最小值为i的最优找零数
    }
    cout<<F[n]<<endl;
    return 0;
}
```

E: 骨牌覆盖

- 难度：3
- 知识点: 动态规划： 第一个骨牌如果纵向放置则剩余方方案为F_{n-1},第一个骨牌如果横向放置则剩余方案数为F_{n-3}，则动态规划状态转移方程为 $F_n = F_{n-1} + F_{n-3}$
- 参考解答：

```
#include<bits/stdc++.h>
using namespace std;
const int MOD=1000000007;//模常数定义
int main() {
    int n;
    cin>>n;
    vector<int> F(n+1);//建立数组F F[n]存储长度为3*n的骨牌放置方案数
    F[1]=F[2]=1;    F[3]=2; //初始化
    for(int i=4; i<=n; i++)//递推
        F[i]=(F[i-1]+F[i-3])%MOD;
    cout<<F[n]<<endl;
    return 0;
}
```

F: 最小生成树

- 难度：4
- 知识点:图论、最小生成树、Prim算法，Kruskal
- 参考解答

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

const int N=100;//图的最大顶点数
int main() {
    int n;
    cin >> n ;
    int sum = 0;
    int a[N + 1][N + 1] = {0};//邻接矩阵
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cin>>a[i][j];//录入a[i][j]
            //注意到题目中如果不是对角线上的0表示i,j是不联通的，所以需要初始化为INT_MAX
            if(i!=j&&a[i][j]==0)
                a[i][j]=INT_MAX;
        }
    }

    int dis[N + 1];//源点到各点的距离数组（动态规划）
    int vis[N + 1] = {0};//访问标记数组
    for (int i = 1; i <= n; i++) {
        dis[i] = a[1][i]; //由于开始源点集合只有{1}所以先将第i个点到1的距离，初始化为原始距离
    };
    vis[1] = 1;//1加入到源点集合S中
    for (int i = 1; i <= n - 1; i++) {//一下加入n-1个节点，
        int Min = INT_MAX;
        int u;
        for (int j = 1; j <= n; j++) {//更新Min
            if (vis[j] == 0 && Min > dis[j]) {
                Min = dis[j];
                u = j;
            }
        }
        vis[u] = 1;
        sum += dis[u];//把u加入到源点集合S中，用最新的S集合更新dis数组
        for (int j = 1; j <= n; j++) {
            if (vis[j] == 0 && dis[j] > a[u][j]) {
                dis[j] = a[u][j];
            }
        }
    }
    cout << sum << endl;//sum就是最小生成树长度
    return 0;
}
```