A.hello

按题目要求输出即可。

```
#include<iostream>
using namespace std;
int main()
{
	string s;
	cin>>s;
	cout<<"hello "<<s<<"!"<<endl;
	return 0;
}
```

B.cola

优先使用大的杯子，便可得到最少个数，要注意当 n 为 0 时，只要用 0 个杯子。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=15;
int p[N];
int main()
{
	int n,m,sum;
	scanf("%d%d",&n,&m);
	sum=0;
	for(int i=0;i<m;i++)
	{
		scanf("%d",&p[i]);
		sum+=p[i];
	}
	if(sum<n)
	{
		printf("NO\n");
	}
	else
```

```
    {
        sort(p,p+m);
        int ans=0;
        for(int i=m-1;i>=0;i--)
        {
            if(n<=0) break;
            n-=p[i];
            ans++;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

C.fengbao

按题目要求处理即可，要注意名字的判重，可以使用 map,set 等 STL 库。

```
#include<iostream>
#include<cstdio>
#include<map>
using namespace std;
map<string,int> m;
int main()
{
    int n;
    string a,b;
    cin>>n;
    int s=0;
    bool flag=true;
    for(int i=0;i<n;i++)
    {
        cin>>a>>b;
        for(int j=0;j<b.length();j++)
            b[j]=tolower(b[j]);
        if(b=="fengbaoyaohuo")
        {
            if(a=="HuangXuDong")
                flag=false;
            else
            {
                if(!m[a])
                {
```

```
                        s++;
                        m[a]++;
                    }
                }
            }
        }
        if(!flag||s==0) cout<<"FengBaoLiangLe"<<endl;
        else cout<<"FengBaoYaoHuo = "<<s<<endl;
        return 0;
}
```

D.chocolate

按题目要求处理，注意到最后喜悦度会变得很大，需要使用 long long 整型变量。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=50+10;
int a[N];
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    sort(a,a+n);
    ll t=0,sum=0;
    for(int i=n-1;i>=0;i--)
    {
        t=a[i]+sum;
        printf("%lld\n",t);
        sum+=t;
    }
    return 0;
}
```

E.fibonacci

大量查询 1e8 以内的斐波那契数列，有三种做法：矩阵快速幂，离线查询，分段打表。这里给出离线查询代码。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;
const int N=1e5+10;
const int MOD=1e9+7;
struct node
{
	int id;
	int n;
	bool operator < (const node &u) const
	{
		return n<u.n;
	}
}q[N];
int ans[N];
int main()
{
	int T;
	scanf("%d",&T);
	int max_n=0;
	for(int i=0;i<T;i++)
	{
		q[i].id=i;
		scanf("%d",&q[i].n);
		max_n=max(max_n,q[i].n);
	}
	sort(q,q+T);
	int p=0,a=0,b=1;
	while(q[p].n==1)
	{
		ans[q[p].id]=1;
		p++;
	}
	for(int i=2;i<=max_n;i++)
	{
		int c=a+b;
		c%=MOD;
```

```
            while(q[p].n==i)
            {
                ans[q[p].id]=c;
                p++;
            }
            a=b;
            b=c;
        }
        for(int i=0;i<T;i++)
            printf("%d\n",ans[i]);
        return 0;
}
```

F.follow

保存每个题目通过的人数，然后找出通过人数最多且自己没有通过的题。注意通过人数为 0 时的情况。

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
const int M=20;
int num[M],vis[M];
int main()
{
        int n,m,p,a;
        memset(num,0,sizeof(num));
        memset(vis,0,sizeof(vis));
        scanf("%d%d%d",&n,&m,&p);
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++)
            {
                scanf("%d",&a);
                if(i==p) vis[j]+=a;
                num[j]+=a;
            }
        int max_num=-1,select=0;
        for(int j=1;j<=m;j++)
            if(!vis[j]&&num[j]>max_num)
            {
```

```
                max_num=num[j];
                select=j;
            }
        printf("%d\n",select);
        return 0;
}
```

G.acm

按题意模拟，较复杂的模拟题。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=300+10;
const int M=30;
struct node
{
        string name;
        int num,t;
        int z[M],c[M];
        bool ac[M];
        bool operator < (const node &u) const
        {
                if(num!=u.num) return num>u.num;
                if(t!=u.t) return t<u.t;
                return name<u.name;
        }
}a[N];
int main()
{
        int n,m,k;
        scanf("%d%d%d",&n,&m,&k);
        for(int i=0;i<n;i++)
        {
                cin>>a[i].name;
                a[i].num=a[i].t=0;
                memset(a[i].z,0,sizeof(a[i].z));
```

```
        memset(a[i].c,0,sizeof(a[i].c));
        memset(a[i].ac,false,sizeof(a[i].ac));
    }
    while(k--)
    {
        string s;
        cin>>s;
        int p=0,id,hour,minute;
        for(;p<n;p++)
            if(a[p].name==s) break;
        cin>>s;
        id=s[0]-'A';
        cin>>s;
        hour=(s[0]-'0')*10+(s[1]-'0');
        minute=(s[3]-'0')*10+(s[4]-'0');
        cin>>s;
        if(a[p].ac[id]==true) continue;
        a[p].z[id]++;
        if(s=="AC")
        {
            a[p].ac[id]=true;
            a[p].num++;
            a[p].t+=hour*60+minute+a[p].c[id]*20;
        }
        else if(s!="CE")
        {
            a[p].c[id]++;
        }
    }
    sort(a,a+n);
    for(int i=0;i<n;i++)
    {
        cout<<a[i].name<<" "<<a[i].num<<" "<<a[i].t;
        for(int j=0;j<m;j++)
        {
            cout<<" ";
            if(a[i].ac[j]) cout<<a[i].z[j];
            else cout<<-a[i].z[j];
        }
        cout<<endl;
    }
    return 0;
}
```

H.tree

在节点上维护子树上的最大值，更新时用懒惰标记即可，类似线段树。
也可直接用 dfs 序+线段树解决。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<vector>
#include<stack>
using namespace std;
typedef long long ll;
const int N=100000+10;
int a[N],f[N],lazy[N];
vector<int> son[N];
int main()
{
	int n,m;
	scanf("%d%d",&n,&m);
	for(int i=1;i<=n;i++)
	{
		scanf("%d%d",&f[i],&a[i]);
		if(f[i]!=i) son[f[i]].push_back(i);
	}
	for(int i=1;i<=n;i++)
		if(son[i].size()==0)
		{
			int u=i;
			while(f[u]!=u)
			{
				a[f[u]]=max(a[f[u]],a[u]);
				u=f[u];
			}
		}
	int x,y;
	while(m--)
	{
		scanf("%d%d",&x,&y);
		int u=x;
		stack<int> s;
		while(f[u]!=u)
		{
```

```
            u=f[u];
            s.push(u);
        }
        int k=0;
        while(!s.empty())
        {
            u=s.top();
            s.pop();
            for(int i=0;i<son[u].size();i++)
            {
                a[son[u][i]]+=lazy[u];
                lazy[son[u][i]]+=lazy[u];
            }
            lazy[u]=0;
        }
        lazy[x]+=y;
        a[x]+=y;
        u=x;
        while(f[u]!=u)
        {
            a[f[u]]=max(a[f[u]],a[u]);
            u=f[u];
        }
        printf("%d\n",a[1]);
    }
    return 0;
}
```

I.reversi

dfs 搜索+回溯，按照步数来深搜，每步要找到可以走的地方，并处理棋子的翻转，详见代码。

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<ctime>
#include<cstdlib>
using namespace std;
const int N=8;
const int dx[]={0,0,-1,1,-1,1,-1,1};
```

```
const int dy[]={-1,1,0,0,-1,1,1,-1};
char a[10][10];
bool judge(int x,int y,char c1,char c2)
{
	bool flag=false;
	for(int i=0;i<8;i++)
	{
		int ux=x+dx[i],uy=y+dy[i];
		bool f=false;
		while(ux>=0&&ux<N&&uy>=0&&uy<N)
		{
			if(a[ux][uy]==c1)
			{
				if(f) flag=true;
				break;
			}
			if(a[ux][uy]=='.')
			{
				break;
			}
			if(a[ux][uy]==c2)
			{
				f=true;
			}
			ux+=dx[i];
			uy+=dy[i];
		}
	}
	return flag;
}
void go(int x,int y,char c1,char c2)
{
	for(int i=0;i<8;i++)
	{
		bool flag=false;
		int ux=x+dx[i],uy=y+dy[i];
		bool f=false;
		while(ux>=0&&ux<N&&uy>=0&&uy<N)
		{
			if(a[ux][uy]==c1)
			{
				if(f) flag=true;
				break;
			}
```

```
                if(a[ux][uy]=='.')
                {
                    break;
                }
                if(a[ux][uy]==c2)
                {
                    f=true;
                }
                ux+=dx[i];
                uy+=dy[i];
            }
            if(flag)
            {
                int ux=x+dx[i],uy=y+dy[i];
                while(ux>=0&&ux<N&&uy>=0&&uy<N)
                {
                    if(a[ux][uy]==c1)
                        break;
                    a[ux][uy]=c1;
                    ux+=dx[i];
                    uy+=dy[i];
                }
            }
        }
        a[x][y]=c1;
}
bool ok()
{
    for(int i=0;i<N;i++)
        for(int j=0;j<N;j++)
            if(a[i][j]=='O')
                return false;
    return true;
}
bool dfs(int d)
{
    if(ok()) return true;
    if(d==5) return false;
    char b[10][10];
    memcpy(b,a,sizeof(a));
    for(int i=0;i<N;i++)
        for(int j=0;j<N;j++)
            if(a[i][j]=='.'&&judge(i,j,'X','O'))
            {
```

```
                        go(i,j,'X','O');
                        if(dfs(d+1)) return true;
                        memcpy(a,b,sizeof(a));
                    }
        return false;
}
int main()
{
        for(int i=0;i<8;i++)
            scanf("%s",a[i]);
        if(dfs(0)) printf("YES\n");
        else printf("NO\n");
        return 0;
}
```