North America Qualifier 2015 Solutions

整套题没有难题，难度分布在国内的铁牌到铜牌题。

A. All about that base

判断表达式在 $1-36$ 进制下是否合法。

需要注意当某位数不在该进制范围内，或某个数在 unsigned int 范围之外也是不合法的。

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ll;

ll a,b,c,flag;
char op,A[105],B[105],C[105];
vector<int> valid;

inline int c2i(char x)
{
    if(x>='0'&&x<='9') return x-'0';
    return x-'a'+10;
}

inline char i2c(int x)
{
    if(x==36) return '0';
    if(x>=10) return x-10+'a';
    return x+'0';
}

ll gao(int base,char s[])
{
    ll ret=0;
    for(int j=0;s[j];j++)
    {
        ret*=base,ret+=c2i(s[j]);
        if(ret>=1ll<<32) {flag=0;break;}
    }
    return ret;
```

```
}

void solve()
{
    int low=1,a1=0;
    valid.clear();
    scanf("%s %c %s = %s",A,&op,B,C);
    for(int i=0;A[i];i++) low=max(low,c2i(A[i])),a1+=A[i]=='0';
    for(int i=0;B[i];i++) low=max(low,c2i(B[i])),a1+=B[i]=='0';
    for(int i=0;C[i];i++) low=max(low,c2i(C[i])),a1+=C[i]=='0';
    if(a1||low!=1) low++;
    for(int i=low;i<=36;i++)
    {
        flag=1;
        a=gao(i,A),b=gao(i,B),c=gao(i,C);
        if(!flag) continue;
        if(op=='+'&&a+b==c||op=='*'&&a*b==c
           ||op=='-'&&a-b==c||op=='/'&&b!=0&&a%b==0&&a/b==c)
            valid.push_back(i);
    }
    if(valid.empty()) printf("invalid");
    else for(auto c:valid) printf("%c",i2c(c));
    printf("\n");
}

int main()
{
    int _;
    scanf("%d",&_);
    while(_--) solve();
    return 0;
}
```

B. Bobby's Bet

简单的概率计算，期望赢钱数为

$$W \times \sum_{i=X}^{Y} \binom{Y}{i} (1-\frac{R-1}{S})^i (\frac{R-1}{S})^{Y-i}$$

由于要求答案是严格大于，要注意一些精度问题。

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

int c[21][21];

double mp(double x,int y)
{
    double ret=1;
    while(y--) ret*=x;
    return ret;
}

int main()
```

```
{
    c[0][0]=1;
    for(int i=1;i<=20;i++)
    {
        c[i][0]=c[i][i]=1;
        for(int j=1;j<i;j++) c[i][j]=c[i-1][j]+c[i-1][j-1];
    }
    int _;
    scanf("%d",&_);
    while(_--)
    {
        int r,s,x,y,w;
        scanf("%d%d%d%d%d",&r,&s,&x,&y,&w);
        double p=0;
        for(int i=x;i<=y;i++)
            p+=mp(1-(r-1.0)/s,i)*mp((r-1.0)/s,y-i)*c[y][i];
        if(p*w>1.0+1e-8) printf("yes\n");
        else printf("no\n");
    }
    return 0;
}
```

C. Cantina of Babel

按照题意模拟可以把图建出来，去掉最少的点使得剩下的点之间两两可达。

剩下的点就是图里最大的 SCC ，跑一遍 tarjan 即可。

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

int n,dfn[105],low[105],st[105],ins[105],num,top,ans;
string s[105];
set<string> u[105];
vector<int> e[105];
char ss[1000],t[1000];

void tarjan(int x)
{
    dfn[x]=low[x]=++num;
    st[++top]=x,ins[x]=1;
    for(auto to:e[x])
        if(!dfn[to])
        {
            tarjan(to);
            low[x]=min(low[x],low[to]);
        }
        else if(ins[to])
            low[x]=min(low[x],dfn[to]);
    if(dfn[x]==low[x])
    {
        int cnt=0,y;
        do
        {
            ins[y=st[top--]]=0;
```

```
            cnt++;
        }while(x!=y);
        ans=max(cnt,ans);
    }
}

int main()
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int n,cur=0;
        scanf(" %[^\n]",ss);
        n=strlen(ss);
        sscanf(ss+cur,"%s",t);cur+=strlen(t);
        sscanf(ss+cur,"%s",t);cur+=strlen(t)+1;
        s[i]=string(t);
        while(cur<n)
        {
            sscanf(ss+cur,"%s",t);cur+=strlen(t)+1;
            u[i].emplace(t);
        }
    }
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
        {
            if(i==j) continue;
            if(s[i]==s[j]||u[j].find(s[i])!=u[j].end()) e[i].push_back(j);
        }
    for(int i=0;i<n;i++)
        if(!dfn[i]) tarjan(i);
    printf("%d",n-ans);
    return 0;
}
```

D. Circuit Counting

注意到坐标范围一定在 $[-400,400]$ 内，我们用 $dp[i][x][y]$ 表示前 i 个点选出若干个得到 (x,y) 的方案数。

有转移

$$dp[i][x][y]=dp[i-1][x][y]+dp[i-1][x-X_i][y-Y_i]$$

可以滚动掉一维。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int n;
ll cnt[2][1005][1005];

int main()
{
```

```
    scanf("%d",&n);
    cnt[1][500][500]=1;
    for(int i=0,x,y;i<n;i++)
    {
        scanf("%d%d",&x,&y);
        for(int X=0;X<=1000;X++)
            for(int Y=0;Y<=1000;Y++)
            {
                cnt[i&1][X][Y]=cnt[(i&1)^1][X][Y];
                if(X-x>=0&&X-x<=1000&&Y-y>=0&&Y-y<=1000)
                    cnt[i&1][X][Y]+=cnt[(i&1)^1][X-x][Y-y];
            }
    }
    n=n&1,n^=1;
    printf("%lld",cnt[n][500][500]-1);
    return 0;
}
```

E. Cutting Brownies

对于先手来说，如果所有决策都无法让后手必败，那么在这个状态下先手就是必败的。

令 $dp[i][j][0/1]$ 表示当前局面为 (i,j) 时，先手是 H/V 是否有必胜策略。边界为 $dp[1][1][0]=dp[1][1][1]=0$。

于是我们可以记忆化 dfs。

也可以打表，不难发现规律。

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

int n,m,dp[505][505][2];//HV
char s[100];

int dfs(int n,int m,int turn)
{
    if(dp[n][m][turn]!=-1) return dp[n][m][turn];
    int ret=0;
    if(turn==1)
        for(int i=1;i<n;i++) ret|=!(dfs(i,m,turn^1)||dfs(n-i,m,turn^1));
    else
        for(int i=1;i<m;i++) ret|=!(dfs(n,i,turn^1)||dfs(n,m-i,turn^1));
    return dp[n][m][turn]=ret;
}

int main()
{
    int _;
    scanf("%d",&_);
    memset(dp,-1,sizeof(dp));
    dp[1][1][0]=dp[1][1][1]=0;
    while(_--)
    {
        scanf("%d%d%s",&n,&m,s);
        if(s[0]=='H')
```

```
            if(dfs(n,m,0)) printf("Harry can win\n");
            else printf("Harry cannot win\n");
        else
            if(dfs(n,m,1)) printf("Vicky can win\n");
            else printf("Vicky cannot win\n");
    }
    return 0;
}
```

F. Quick Brown Fox

签到题。

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

int cnt[26],tot;
char s[105];

int main()
{
    int _;
    scanf("%d",&_);
    while(_--)
    {
        tot=0;
        scanf(" %[^\n]",s);
        memset(cnt,0,sizeof(cnt));
        for(int i=0;s[i];i++)
            if(s[i]>='a'&&s[i]<='z') cnt[s[i]-'a']++;
            else if(s[i]>='A'&&s[i]<='Z') cnt[s[i]-'A']++;
        for(int i=0;i<26;i++) tot+=cnt[i]>0;
        if(tot==26) printf("pangram\n");
        else
        {
            printf("missing ");
            for(int i=0;i<26;i++)
                if(!cnt[i]) printf("%c",i+'a');
            printf("\n");
        }
    }
    return 0;
}
```

G. Safe Passage

经典问题。

假设 $a[]$ 已经排完序了，表示移动一次的代价。我们可以利用 $a[1]$ 和 $a[2]$ 来尽量使得代价总和最小。

设 $dp[i]$ 表示前 i 个人都已经到目的地时的最小代价。

此时有两种决策可以得到当前状态：

1. 让 $a[1]$ 回来，再让 $a[1]$ 和 $a[i]$ 一起去，代价为 $dp[i-1]+a[1]+a[i]$。
2. 让 $a[1]$ 回来，再让 $a[i-1]$ 和 $a[i]$ 一起去，再让 $a[2]$ 回来，再让 $a[1]$ 和 $a[2]$ 一起去，代价为 $dp[i-2]+a[1]+a[i]+2*a[2]$ 。

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

int n,a[16],dp[16];

int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",a+i);
    sort(a+1,a+1+n);
    dp[1]=a[1],dp[2]=a[2];
    for(int i=3;i<=n;i++)
        dp[i]=min(dp[i-1]+a[1]+a[i],dp[i-2]+a[1]+a[i]+2*a[2]);
    printf("%d",dp[n]);
    return 0;
}
```

H. Secret Message

签到题，模拟即可。

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

int n,k;
char s[10005],a[105][105];

void solve()
{
    scanf("%s",s);
    n=strlen(s);
    memset(a,'*',sizeof(a));
    for(k=1;k*k<n;k++);
    for(int i=0;i<n;i++) a[i/k][i%k]=s[i];
    for(int j=0;j<k;j++)
        for(int i=k-1;i>=0;i--)
            if(a[i][j]!='*') printf("%c",a[i][j]);
    printf("\n");
}

int main()
{
    int _;
    scanf("%d",&_);
    while(_--) solve();
    return 0;
}
```

I. Simon Says

签到题，判断一行的前缀是否包含 `Simon says`。

注意输出时应该带空格，之前校内Domjudge的比较器把空格吃了导致不带空格的也过了，丢到计蒜客上就发现跑不过了。(~~说话带空格~~)

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

const char t[]="Simon says ";
char s[105];

int main()
{
    int _;
    scanf("%d",&_);
    while(_--)
    {
        int flag=1,n;
        scanf(" %[^\n]",s);
        n=strlen(s);
        if(n>11)
            for(int i=0;i<11;i++) flag&=t[i]==s[i];
        else
            flag=0;
        if(flag) printf(" %s\n",s+11);
    }
    return 0;
}
```

J. Torn To Pieces

给一无向图和起点终点，保证起点到终点的简单路径最多只有一条，求路径。

注意读入比较恶心，真实的点数可能大于 n ，然后直接 $\mathrm{dfs}/\mathrm{bfs}$ 即可。

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

int n,s,t,tot,vis[33*33];
char ss[1005],tt[1005];
map<string,int> id;
string nm[33*33];
vector<int> e[33*33],p;

void dfs(int now,int fa)
{
    vis[now]=1;
    p.push_back(now);
    if(now==t)
    {
        for(auto it:p) printf("%s ",nm[it].c_str());
        exit(0);
    }
```

```
    for(auto to:e[now])
        if(to!=fa&&!vis[to]) dfs(to,now);
    p.pop_back();
}

int main()
{
    scanf("%d",&n);
    while(n--)
    {
        int l,cur=0,now,to;
        scanf(" %[^\n]",ss);
        l=strlen(ss);
        sscanf(ss+cur,"%s",tt);cur+=strlen(tt);
        if(id.count(string(tt))==0)
        {
            id[string(tt)]=++tot;
            nm[tot]=string(tt);
        }
        now=id[string(tt)];
        while(cur<l)
        {
            sscanf(ss+cur,"%s",tt);cur+=strlen(tt)+1;
            if(id.count(string(tt))==0)
            {
                id[string(tt)]=++tot;
                nm[tot]=string(tt);
            }
            to=id[string(tt)];
            e[to].push_back(now),e[now].push_back(to);
        }
    }
    scanf("%s%s",ss,tt);
    if(id.count(string(ss))==0)
    {
        id[string(ss)]=++tot;
        nm[tot]=string(ss);
    }
    if(id.count(string(tt))==0)
    {
        id[string(tt)]=++tot;
        nm[tot]=string(tt);
    }
    s=id[string(ss)],t=id[string(tt)];
    dfs(s,s);
    printf("no route found");
    return 0;
}
```

K. UnDetected

问题可以简化成，k 最大能取到多少，使得前 k 个圆不能使左边界和右边界联通。

并查集维护圆与圆、圆与左右边界的联通性即可。

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

int fa[205],n,x[205],y[205],r[205];

int _find(int x){return fa[x]==x?x:fa[x]=_find(fa[x]);}
void _union(int x,int y){fa[_find(x)]=fa[_find(y)];}

int abs2(int x1,int y1,int x2,int y2)
{
    return (x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
}

int main()
{
    scanf("%d",&n);
    for(int i=0;i<=n+1;i++) fa[i]=i;
    for(int i=1;i<=n;i++)
    {
        scanf("%d%d%d",x+i,y+i,r+i);
        if(x[i]<r[i]) _union(i,0);
        if(x[i]>200-r[i]) _union(i,n+1);
        for(int j=1;j<i;j++)
            if(abs2(x[i],y[i],x[j],y[j])<(r[i]+r[j])*(r[i]+r[j]))
_union(i,j);
        if(_find(0)==_find(n+1))
        {
            printf("%d\n",i-1);
            break;
        }
    }
    return 0;
}
```