```
////////////////////////////////////// ANARC

#include <iostream>
#include <bits/stdc++.h>
#include <stack>
#include <list>
#include <set>
#include <cmath>
#include <stdio.h>

using namespace std;

#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",&x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);

int main(int argc, char const *argv[])
{
        int cn = 1;
        while(1){
                int n;
                sd(n);
                if(n==0)
                        break;
                int mat[n][n];

                for(int i=0; i<n; ++i)
                        for(int j=0; j<n; ++j)
                                sd(mat[i][j]);

                int val1, val2, val3;
                val3=val2=val1=0;

                for(int i=0; i<n; ++i){
                        for(int j=0; j<n; ++j){
                                val1+=mat[i][j];
                        }
                }

                int max_diff = 0;
                for(int i=0; i<n; ++i){
                        int temp1=0;
                        int temp2=0;
                        for(int j=0; j<n; ++j)
                                if(i!=j)
                                        temp1+=mat[i][j];
                        for(int j=0; j<n; ++j)
                                if(i!=j)
                                        temp2+=mat[j][i];
                        // printf("%d %d\n", temp1, temp2);
                        int k=temp2-temp1;
                        if(k>0) //k*=-1;
                                max_diff+=k;
                }
                printf("%d. %d %d\n", cn, val1, max_diff);
                cn++;
        }
        return 0;
}

////////////////////////////////////////////////////// COLOR
#include <iostream>
// #include <bits/stdc++.h>
#include <stack>
```

```
#include <list>
#include <set>
#include <cmath>
#include <stdio.h>

using namespace std;

#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",&x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);

/*
        Using pair in C++ :
        ???
*/

int main(int argc, char const *argv[]){
        int n, m;
        sd(n); sd(m);
        int color[n+1];
        list<int> graph[n+1];
        int max_col=0;
        for(int i=1;i<=n;++i){
                sd(color[i]);
                if(color[i]>max_col)
                        max_col=color[i];
        }
        list<int> v_k[max_col+1];

        // To generate the graph
        for(int i=1;i<=m;++i){
                int n1, n2;
                sd(n1); sd(n2);
                graph[n1].push_back(n2);
                graph[n2].push_back(n1);
        }

        // To generate the V(k)
        for(int i=1;i<=n;++i){
                v_k[color[i]].push_back(i);
        }

        int ans = 0;
        int cor_color = 0;

        for(int i=0; i<=max_col; ++i){
                set<int> q_k;
                for(list<int>::iterator node=v_k[i].begin(); node!= v_k[i].end(); ++node){
                        for(list<int>::iterator neigh_nodes=graph[*node].begin(); neigh_nod
es!= graph[*node].end(); ++neigh_nodes){
                                if(color[*neigh_nodes]!=i)
                                        q_k.insert(color[*neigh_nodes]);
                        }
                }
                int count = q_k.size();
                if(count > ans){
                        ans = count;
                        cor_color = i;
                }
        }

        printf("%d\n", cor_color);
        return 0;
}
```

```
////////////////////////////////////////////// DIJKSTRA

#include <iostream>
#include <stdio.h>
#include <bits/stdc++.h>
#include <stdlib.h>
#include <cmath>
#include <queue>

using namespace std;

#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);

#define pii pair<int,int>

int dijkstra(vector< pii > graph[], int n, int source, int dest){
        int dist[n+1];
        for(int i=1 ;i<=n; ++i)
                dist[i] = INT_MAX;
        bool visited[n+1];
        for(int i=1 ;i<=n; ++i)
                visited[i] = false;

        priority_queue<pii> q;

        dist[source]    = 0;
        q.push(pii(0,source));
        while(!q.empty()){
                int node = q.top().second;
                q.pop();
                if(visited[node])
                        // break;
                        continue;
                for(vector< pii >::iterator i=graph[node].begin(); i!=graph[node].end(); ++
i){
                        pii temp = *i;
                        int u = temp.first;
                        int u_weight = temp.second;

                        if(!visited[u] && u_weight+dist[node] < dist[u]){
                                dist[u] = u_weight+dist[node];
                                q.push(pii(-1*dist[u], u));
                        }
                }
                visited[node] = true;
        }
        return dist[dest];
}

int main(int argc, char const *argv[])
{
        int t; sd(t);
        while(t--){
                int v,k; sd(v); sd(k);
                vector< pii > graph[v+1];
                for(int i=1 ;i<=k; ++i){
                        int n1, n2, w;
                        sd(n1); sd(n2); sd(w);
                        graph[n1].push_back(pii(n2,w));
                }
                int A,B; sd(A); sd(B);
```

```
                int minDist = dijkstra(graph, v, A, B);
                if(minDist==INT_MAX)
                        printf("NO\n");
                else
                        printf("%d\n", minDist);
        }
        return 0;
}

////////////////////////////////////////////////// MAKEMAZE
#include <iostream>
#include <list>
#include <queue>
#include <stdio.h>
#include <bits/stdc++.h>
#include <string.h>

#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",&x);
#define sc(x)   scanf("%c",&x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);
#define pc(x)   printf("%c",x);


using namespace std;

bool bfs(string mat[], int m, int n, pair<int,int> start, pair<int,int> end){
        queue< pair<int,int> > q;
        bool visit[n][m];
        for(int i=0; i<n; ++i)
                for(int j=0; j<m; ++j)
                        visit[i][j] = false;
        q.push(start);
        visit[start.first][start.second] = true;
        while(!q.empty()){
                pair<int, int> temp = q.front();
                // printf(" first : %d second : %d\n", temp.first, temp.second);
                q.pop();

                int i,j;
                i = temp.first-1;
                j = temp.second;
                if(i>=0 && j>=0 && i<n && j<m && !visit[i][j] && mat[i][j]=='.'){
                        visit[i][j] = true;
                        pair<int,int> repo;
                        repo.first = i;
                        repo.second = j;
                        q.push(repo);
                        if(repo==end)
                                return true;
                }

                i = temp.first+1;
                j = temp.second;
                if(i>=0 && j>=0 && i<n && j<m && !visit[i][j] && mat[i][j]=='.'){
                        visit[i][j] = true;
                        pair<int,int> repo;
                        repo.first = i;
                        repo.second = j;
                        q.push(repo);
                        if(repo==end)
                                return true;
                }
```

```
                i = temp.first;
                j = temp.second-1;
                if(i>=0 && j>=0 && i<n && j<m && !visit[i][j]  && mat[i][j]=='.'){
                        visit[i][j] = true;
                        pair<int,int> repo;
                        repo.first = i;
                        repo.second = j;
                        q.push(repo);
                        if(repo==end)
                                return true;
                }

                i = temp.first;
                j = temp.second+1;
                if(i>=0 && j>=0 && i<n && j<m && !visit[i][j]  && mat[i][j]=='.'){
                        visit[i][j] = true;
                        pair<int,int> repo;
                        repo.first = i;
                        repo.second = j;
                        q.push(repo);
                        if(repo==end)
                                return true;
                }


        }
        return false;
}


int main(int argc, char const *argv[]) {
        int t,n,m;
        sd(t);
        while(t--){
                sd(n);
                sd(m);

                string mat[n];
                for(int i=0; i<n; ++i)
                        cin>>mat[i];

                bool extra = false;
                for(int i=0; i<n; ++i)
                        for(int j=0; j<m; ++j)
                                if(mat[i][j]!='.' && mat[i][j]!='#'){
                                        extra = true; break;}
                if(extra){
                        printf("invalid\n");
                        continue;
                }

                pair<int,int> start, end;
                int count=0;

                int pos=0;
                for(int i=0; i<n; ++i){
                        if(mat[i][0]=='.'){
                                if(pos==0){
                                        start.first = i;
                                        start.second = 0;
                                        pos++;
                                }
                                else if(pos==1){
                                        end.first = i;
                                        end.second = 0;
                                        pos++;
                                }
                                count++;
```

```
                }
            }

            if(m!=1){
                for(int i=0; i<n; ++i){
                    if(mat[i][m-1]=='.'){
                        if(pos==0){
                            start.first = i;
                            start.second = m-1;
                            pos++;
                        }
                        else if(pos==1){
                            end.first = i;
                            end.second = m-1;
                            pos++;
                        }
                        count++;
                    }
                }
            }

            for(int i=1; i<m-1; ++i){
                if(mat[0][i]=='.'){
                    if(pos==0){
                        start.first = 0;
                        start.second = i;
                        pos++;
                    }
                    else if(pos==1){
                        end.first = 0;
                        end.second = i;
                        pos++;
                    }

                    count++;
                }
            }
            if(n!=1){
                for(int i=1; i<m-1; ++i){
                    if(mat[n-1][i]=='.'){
                        if(pos==0){
                            start.first = n-1;
                            start.second = i;
                            pos++;
                        }
                        else if(pos==1){
                            end.first = n-1;
                            end.second = i;
                            pos++;
                        }
                        count++;
                    }
                }
            }

            // printf("start : %d %d\n", start.first, start.second);
            // printf("end : %d %d\n", end.first, end.second);
            if(count>2){
                printf("%s\n", "invalid");
            }
            else{
                bool res = bfs(mat, m, n, start, end);
                if(res)
                    printf("%s\n","valid" );
                else
                    printf("invalid\n");
            }
        }
```

```
        return 0;
}

////////////////////////////////////////////////// PPATH

#include <iostream>
#include <stdio.h>
#include <queue>
#include <cmath>

#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",&x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);

using namespace std;

bool isPrime[10000], visited[10000];
int level[10000];



bool checkPrime(int n){
        for(int i=2; i<=sqrt(n); ++i){
                if(n%i==0)
                        return false;
        }
        return true;
}


void initPrime(){
        for(int i=1000; i<10000; ++i){
                isPrime[i] = checkPrime(i);
        }

}

void initVisitLevel(){
        for(int i=0; i<10000; ++i){
                visited[i]=false;
                level[i]=0;
        }
}

int bfs(int start, int end){
        initVisitLevel();
        queue<int> q;
        visited[start] = true;
        level[start] = 0;
        q.push(start);

        while(!q.empty()){

                int temp = q.front();
                q.pop();

                for(int pos = 0 ; pos<4; ++pos){
                        int ones, two, hund, thou;

                        ones = temp%10;
                        two      = (temp%100)/10;
                        hund = (temp%1000)/100;
                        thou = (temp%10000)/1000;
```

```
                        for(int val = 0; val<=9 ; ++val){
                                int genNum = 0;
                                if(pos==0){
                                        genNum = thou*1000+hund*100+two*10+((ones+val)%10);
                                }
                                else if(pos==1){
                                        genNum = thou*1000+hund*100+((two+val)%10)*10+ones;
                                }
                                else if(pos==2){
                                        genNum = thou*1000+((hund+val)%10)*100+two*10+ones;
                                }
                                else if(pos==3){
                                        int jki  = (thou+val)%10;
                                        if(jki!=0)
                                                genNum = ((thou+val)%10)*1000+hund*100+two*
10+ones;
                                }

                                if(!visited[genNum] && isPrime[genNum] && genNum!=temp){
                                        visited[genNum]=true;
                                        level[genNum]=level[temp] + 1;
                                        q.push(genNum);
                                        if(genNum==end)
                                                return level[end];
                                }

                        }

                }

        }

        return level[end];
}

int main(int argc, char const *argv[]) {
        initPrime();
        int T;
        int n1, n2;
        sd(T);

        while(T--){
                sd(n1);
                sd(n2);
                if(n1==n2)
                        printf("0\n");
                else{
                        int lev = bfs(n1,n2);
                        if(lev!=0)
                                printf("%d\n", lev);
                        else
                                printf("Impossible\n");
                }

        }
        return 0;
}

////////////////////////////////////////////// ALLIZZWELL

#include <iostream>
#include <bits/stdc++.h>
#include <stack>
#include <list>
#include <cmath>
#include <stdio.h>

using namespace std;
```

```
#define sd(x)   scanf("%d",&x);
#define sll(x)  scanf("%lld",&x);
#define ss(x)   scanf("%s",x);
#define ll long long
#define pll(x)  printf("%lld",x);
#define pd(x)   printf("%d",x);
#define ps(x)   printf("%s",x);

#define pii pair<int,int>

char ch[] = "ALLIZZWELL";

string inp[101];
bool vis[101][101];

bool dfsr(int n, int m, int starti, int startj, int point){
        vis[starti][startj] = true;
        bool asd;
        if(point>9){
                        return true;
        }

                int tempr = starti-1;
                int tempc = startj-1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti-1;
                tempc = startj;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti-1;
                tempc = startj+1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti;
                tempc = startj-1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti;
                tempc = startj+1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
```

```
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti+1;
                tempc = startj-1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti+1;
                tempc = startj;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                tempr = starti+1;
                tempc = startj+1;
                if(tempr>=0 && tempr<n && tempc>=0 && tempc<m && inp[tempr][tempc]==ch[poin
t] && !vis[tempr][tempc]){
                        vis[tempr][tempc] = true;
                        asd =dfsr(n, m, tempr, tempc, point+1);
                        if(asd) return true;
                        vis[tempr][tempc] = false;
                }

                vis[starti][startj] = false;
                return false;
}

int main(int argc, char const *argv[])
{
        int t; sd(t);
        while(t--){
                int n,m; sd(n); sd(m);
                for(int i=0;i<n;++i){
                        cin>>inp[i];
                }
                bool res = false;
                bool ans = false;
                for(int i=0; i<n; ++i)
                        for(int j=0; j<m; ++j)
                                        vis[i][j] = false;

                for(int i=0; i<n; ++i){
                        for(int j=0; j<m; ++j){
                                if(inp[i][j]==ch[0]){
                                        // printf("%d %d\n",i,j );
                                        ans = dfsr(n,m,i,j,1);
                                        res = res | ans;
                                        if(res) break;
                                }
                        if(res) break;
                        }
                }
                if(res)
                        printf("%s\n", "YES");
                else
```

```
                        printf("%s\n", "NO");
        }

        return 0;
}

////////////////////////////////////////////////////// PARADOX

#include <bits/stdc++.h>

typedef long long ll;

#define MOD       1000000007
#define scll(t)      scanf("%lld",&t)
#define sc(t)       scanf("%d",&t)
#define max(a,b)      (a>=b?a:b)
#define min(a,b)      (a<b?a:b)
#define gc           getchar_unlocked
#define mp           make_pair
#define pb           push_back
#define freinp       freopen("in.txt","r",stdin)
#define freout       freopen("out.txt","w",stdout)

using namespace std;


vector<int >incoming[109];
int vis[109],truth[109],outgoing[109],claim[109],flag=0;
stack<int> st;

void dfs()
{
    int tval,next_node;
    while(!st.empty())
    {
        int cur_node = st.top();
        st.pop();
        if(vis[cur_node])
            continue;

        vis[cur_node] = 1;

        if(truth[cur_node] == 1)
            tval = claim[cur_node];
        else tval = not(claim[cur_node]);

        next_node = outgoing[cur_node];

        if(!vis[next_node])
        {
            truth[next_node] = tval;
            st.push(next_node);
        }
        else if(truth[next_node] != tval)
        {
            flag = 0;
            return;
        }

        //for incoming edges

        for(int i=0;i<incoming[cur_node].size();i++)
        {
            int inc_node = incoming[cur_node][i];
            if(truth[cur_node] == 1)
                tval = claim[inc_node];
            else tval = not(claim[inc_node]);
```

```
            if(!vis[inc_node])
            {
                st.push(inc_node);
                truth[inc_node] = tval;
            }
            else if(truth[inc_node] != tval)
            {
                flag = 0;
                return;
            }
        }

    }
}

int main()
{

  int t,n,i,x;
 string s;
 sc(n);
 while(n!=0)
 {
        while(!st.empty())
            st.pop();

        flag = 1;

   for(i=0;i<=n;i++)
  {
            incoming[i].clear();
            vis[i] = 0;
            truth[i] = 0;
            claim[i] = 0;
  }
  for(i=1;i<=n;i++)
  {
            sc(x);
            cin>>s;
            if(s=="true")
                claim[i] = 1;
            outgoing[i] = x;
            incoming[x].pb(i);
            st.push(i);
  }
        dfs();

       if(flag)
            printf("NOT PARADOX\n");
        else printf("PARADOX\n");

        sc(n);
 }
}
```