Connection Among Cities

```
#include<iostream>
#include<vector>
#include<map>
#include<queue>
using namespace std;
int main(){
   int totalTest, testCases;
   cin>>totalTest;//test cases
   for(int testCases = 0; testCases< totalTest; testCases++){
      int n, m , p, s, t, i, j, k;
      cin >>n;//number of cities
      vector<pair<int, int>>graph[n];
      map<string, int>city;
      string str;
      for(i = 0; i<n; i++){
         cin >> str;// name of city
         city.insert({str, i});
         cin>>m;  // number of neighbors of Name str
         for(j = 0; j<m; j++){
            int v, d;     // v= nr= index of a city connected to str
            cin>>v>>d;    //  d= cost of transportation
            v--;        // reduce index by 1
            graph[i].push_back({v, d});
         }    }
      cin>>p; //no of paths to find
      while(p--){
         cin>>str; //start city
         s = city[str];
         cin>>str;  //end city
         t = city[str];
         int dist[n]; // for storing the distance
         fill(dist, dist+n, INT_MAX); // set value to all array space
         dist[s] = 0; //distance of traveling starting from source set to 0
         bool visited[n]; // set if the vertex is visited
         fill(visited, visited+n, 0);// fill with 0
         priority_queue<pair<int, int>> q; //store the distance and index of vertex
         q.push({-dist[s], s});// distance, index
         while(!q.empty()){
                  pair<int, int>p = q.top();
                  int x = -p.first,y = p.second;//x= distance from that index, y = city index
                  q.pop();
                  visited[y] = 1;
                  if(y == t) break;// destination
```

```
                for(j = 0;j<graph[y].size();j++){
                    int k = graph[y][j].first;
                    if(visited[k]) continue;
                    int cost = x+graph[y][j].second;
                    if(cost<dist[k]){
                       dist[k] = cost;
                       q.push({-cost, k});
                    }}}
                cout<<dist[t]<<endl;
            }
        }
}
```

Palindrome

```
#include <iostream>
using namespace std;
void solve();
int main(){
   int t;
   cin >> t;
   while (t--){
      solve();
   }
   return 0;
}
void solve(){
   int a;
   cin >> a;
   int arr[a];
   for (int i = 0; i < a; i++){
      cin >> arr[i];
    }
   bool flag = false;
   for (int i = 0; i < a; i++)   {
      for (int j = i + 2; j < a; j++)   {
         if (arr[i] == arr[j]) {
            flag = true;
         }
      }
}
   if (a==1 || flag)cout << "YES" << endl;
      else cout << "NO" << endl;
}
```

Mice Maze problem

```
#include<iostream>
#include<vector>
#include<queue>
using namespace std;
typedef pair < int, int > ii;
#define INF 100000000
vector < ii > g[101];
int N, E, T, M;
int dist[101];
int dijkstra(int src ){
   priority_queue < ii > pq;
   for( int i = 1; i <= N; i++ ) dist[i] = INF;
   pq.push( ii(src, 0));
   dist[src] =0;
   int from, to, t, cost;
   while( !pq.empty() ){
      from = pq.top().first;
      cost = pq.top().second;
      pq.pop();
      for( int i = 0; i < (int) g[from].size(); i++ ){
         to = g[from][i].first;
         t = g[from][i].second;

         if( dist[from] + t < dist[to]){
            dist[to] = dist[from] + t;
            pq.push(ii(to, dist[to]));
         }
      }
   }
   int mices = 0;
   for( int i = 1; i <= N; i++ ){
      if( dist[i] <= T )
         mices += 1;
   }
   return mices;
}
```

```
int main(){
   int test, from, to, t;
   cin>>test;
   while(test--){
      cin>>N>>E>>T;
      cin>>M;
      for( int i = 0; i < M; i++ ){
         cin>>from>>to>>t;
         g[to].push_back( ii(from, t) );
      }

   cout<<dijkstra(E)<<endl<<endl;
   for( int i = 1; i <= N; i++ ) g[i].clear();
   }
         return 0;
}
```

The Number Pattern (Cheeky Cheeky)

```
#include<iostream>
#include<vector>
using namespace std;
typedef long long ll;
vector<ll>pi;

void preFunc(string s){
   ll len=s.size();
   pi.resize(len+5);
   pi[0]=0;
   for(ll i=1;i<len;i++){
      ll j=pi[i-1];
      while(j>0&&s[i]!=s[j])j=pi[j-1];
      if(s[i]==s[j])j++;
      pi[i]=j;
   }
}
int main(){
   ll t;
   cin>>t;
   while(t--){
      string s;
      cin>>s;
      ll len=s.size();
      reverse(s.begin(),s.end());
      preFunc(s);
      ll idx;
      for(ll i=len-1;i>=0;i--){
         if(pi[i]*2==(i+1)){
            idx=pi[i]-1;
            break;
         }
      }
string ans="";
      for(ll i=0;i<=idx;i++)ans+=s[i];
      len=ans.size();
      ll rep=0;
      if(len<8){
         rep=(8/len);
         if(8%len!=0)rep++;
      }
      for(ll i=0;i<rep;i++){
         ans+=ans;
      }
      reverse(ans.begin(),ans.end());
      for(ll i=0;i<8;i++)cout<<ans[i];
      cout<<"...\n";
   }
   return 0;
}
```