Answer To The Question No:1

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
const int N=1e7;
int n,m;
vector<pair<int,int>>adj_list[N];
int dist[N];
int32_t main()
{

    FastIO();
    cin>>n>>m;
    for(int i=0;i<m;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        adj_list[u].push_back({v,w});
        dist[i]=INT_MAX;
    }
    dist[2]=0;
    bool hasCycle=false;
    for(int i=1;i<=n;i++)
    {
        for(int node=1;node<=n;node++)
        {
            for(auto adj_entries:adj_list[node])
            {
                int v=adj_entries.first;
                int w=adj_entries.second;
                if(dist[node]+w<dist[v])
                {
                    dist[v]=dist[node]+w;
                    if(i==n)
                    {
                        hasCycle=true;
                    }
                }
            }
        }
    }
    if(hasCycle)
    {
        cout<<"YES\n";
    }
    else
    {
        cout<<"NO\n";
    }
}
```

Answer To The Question No:2

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
const int N=1e7;
int n,m;
vector<pair<int,int>>adj_list[N];
int parent[N];
int dist[N];
int32_t main()
{

    FastIO();
    cin>>n>>m;
    for(int i=0;i<m;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        adj_list[u].push_back({v,w});
        dist[i]=INT_MAX;
    }
    dist[2]=0;
    bool hasCycle=false;
    int lastUpdatedNode=-1;
    for(int i=1;i<=n;i++)
    {
        for(int node=1;node<=n;node++)
        {
            for(auto adj_entries:adj_list[node])
            {
                int v=adj_entries.first;
                int w=adj_entries.second;
                if(dist[node]+w<dist[v])
                {
                    dist[v]=dist[node]+w;
                    parent[v]=node;
                    if(i==n)
                    {
                        hasCycle=true;
                        lastUpdatedNode=v;
                    }
                }
            }
        }
    }
    if(hasCycle)
    {
        cout<<"YES\n";
        int selectedNode=lastUpdatedNode;
        for(int i=1;i<=n;i++)
        {
            selectedNode=parent[selectedNode];
        }
        vector<int>result;
        int firstNode=selectedNode;
        result.push_back(firstNode);
        while(1)
        {
            selectedNode=parent[selectedNode];
            result.push_back(selectedNode);
            if(selectedNode==firstNode)
            {
                break;
            }
        }
        for(auto i=result.rbegin();i!=result.rend();i++)
        {
            cout<<*i<<" ";
        }
        cout<<"\n";
    }
    else
    {
        cout<<"NO\n";
    }
}
```

Answer To The Question No:3

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
const int N=1e4;
int dist[N][N];
int32_t main()
{

    FastIO();
    int n,m,q;
    cin>>n>>m>>q;
    for(int i=0;i<=n;i++)
    {
        for(int j=0;j<=n;j++)
        {
            dist[i][j]=1e10;
        }
    }
    for(int i=0;i<m;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        dist[u][v]=min(dist[u][v],w);
        dist[v][u]=min(dist[v][u],w);
    }
    for(int i=0;i<=n;i++)
    {
        dist[i][i]=0;
    }
    for(int k=1;k<=n;k++)
    {
        for(int u=1;u<=n;u++)
        {
            for(int v=1;v<=n;v++)
            {
                dist[u][v]=min(dist[u][v],dist[u][k]+dist[k][v]);
            }
        }
    }
    while(q--)
    {
        int x,y;
        cin>>x>>y;
        if(dist[x][y]==1e10)
        {
            cout<<"-1\n";
        }
        else
        {
            cout<<dist[x][y]<<"\n";
        }
    }
}
```

Answer To The Question No:4

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

```
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
const int N=1e5;
int dist[N];
int visited[N];
int parent[N];
vector<pair<int,int>>adj_list[N];
void INIT()
{
    for(int i=0;i<N;i++)
    {
        dist[i]=1e7;
    }
}
void Dijakstra(int start)
{
    dist[start]=0;
    priority_queue<pair<int,int>>pq;
    pq.push({0,start});
    while(pq.empty()==false)
    {
        auto king=pq.top();
        pq.pop();
        visited[king.second]=1;
        for(auto child:adj_list[king.second])
        {
            int u=king.second;
            int w=child.first;
            int v=child.second;
            if(visited[v]==0)
            {
                if(dist[u]+w<dist[v])
                {
                    dist[v]=dist[u]+w;
                    parent[v]=u;
                    pq.push({-1*dist[v],v});
                }
            }
        }
    }
}
int32_t main()
{

    FastIO();
    INIT();
    int n,m;
    cin>>n>>m;
    for(int i=0;i<n;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        adj_list[u].push_back({w,v});
        adj_list[v].push_back({w,u});
    }
    int start=1;
    Dijakstra(1);
    if(visited[n]==0)
    {
        cout<<"-1"<<"\n";
        return 0;
    }
    int currentNode=n;
    vector<int>result;
    while(currentNode!=start)
    {
        result.push_back(currentNode);
        currentNode=parent[currentNode];
    }
    result.push_back(1);
    for(auto i=result.rbegin();i!=result.rend();i++)
    {
        cout<<*i<<" ";
    }
    cout<<"\n";
}
```

Answer To The Question No:5

```
#include <bits/stdc++.h>

using namespace std;
const int N = 1000;
int visited[N][N];
int maze[N][N];
int n, m;
int dx[] = {0, 0, -1, 1};
int dy[] = {1, -1, 0, 0};

bool isSafe(int x, int y)
{
    if (x>=0 and x < n and y >= 0 and y < m)
    {
        return true;
    }
    return false;
}

int  BFS(pair<int, int> src)
{
    int len=1;
    queue<pair<int, int>> q;
    q.push(src);
    while (!q.empty())
    {
        int x = q.front().first;
        int y = q.front().second;
        visited[x][y]=1;
        q.pop();
        for (int i = 0; i < 4; i++)
        {
            int new_x = x + dx[i];
            int new_y = y + dy[i];
            if (isSafe(new_x, new_y) && visited[new_x][new_y] == 0 &&
maze[new_x][new_y] != -1)
            {
                q.push({new_x, new_y});
                visited[new_x][new_y] = 1;
                len++;
            }
        }
    }
    return len;
}

pair<int, int> Give_unvisited()
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {

            if (visited[i][j] == 0 && maze[i][j] == 0)
            {
                return {i, j};
            }
        }
    }

    return {-1, -1};
}
int main()
{

    cin >> n >> m;
    pair<int, int> src, dst;
    for (int i = 0; i < n; i++)
    {
        string input;
        cin >> input;
        for (int j = 0; j < m; j++)
        {
            if (input[j] == '#')
            {
                maze[i][j] = -1;
            }
        }
    }
    int result = 0;
    int length=0;
    while (1)
    {
        if (Give_unvisited() == pair<int, int>(-1, -1))
        {
            break;
        }
        else
        {
            length=max(length,BFS(Give_unvisited()));
            result++;
        }
    }
    cout <<"Rooms: "<<result << "\n"<<"Longest Room lenght:
"<<length<<"\n";
}
```

Answer To The Question No:6

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
int freq[26];
bool solve(string s)
{
    for(int i=0;i<s.size();i++)
    {
        freq[s[i]-'a']++;
    }
    int oddCounter=0;
    for(int i=0;i<26;i++)
    {
        if(freq[i]%2)
        {
            oddCounter++;
        }
    }
    if(oddCounter<=1)
    {
        return true;
    }
    return false;
}
int32_t main()
{

    FastIO();
    string s;
    cin>>s;
    if(solve(s))
    {
        cout<<"YES\n";
    }
    else
    {
        cout<<"NO\n";
    }

}
```

Answer To The Question No:7

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
void FastIO()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
}
int solve(int n)
{
    if(n==0)
    {
        return 0;
    }
    return n%10+solve(n/10);
}
int32_t main()
{

    FastIO();
    int n;
    cin>>n;
    cout<<solve(n)<<"\n";

}
```