Answer Script

Question No. 01

Convert the following adjacency matrix into an adjacency list and draw the graph. **6**

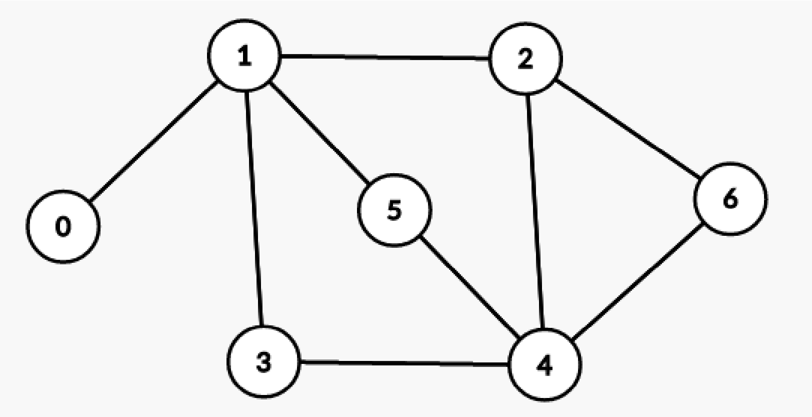
$$\begin{array}{c|ccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
5 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}$$

Answer No. 01

Adjacency Matrix to an Adjacency list:

0: [1]
1: [0, 2, 3, 5]
2: [1, 4, 6]
3: [1, 4]
4: [2, 3, 5, 6]
5: [1, 4]
6: [2, 4]

Graph:



Question No. 02

Write down the difference between BFS and DFS algorithms? (At least three). **6**

Answer No. 02

BFS	DFS
BFS stands for Breadth First Search.	DFS stands for Depth First Search.
BFS uses Queue data structure for finding the shortest path.	DFS uses Stack data structure.
BFS is slower than DFS.	DFS is faster than BFS.

Question No. 03

Write C++ program to solve Perfect Squares by using the Memoization method. **16**

Answer No. 03

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

int solve(int n, vector<int>& memo)
{
        if (n == 0) {
        return 0;
        }
        if (memo[n] != 0) {
        return memo[n];
        }
        int res = n;
        for (int i = 1; i * i <= n; i++) {
        int subproblem = solve(n - i * i, memo);
        res = min(res, subproblem + 1);
        }
        memo[n] = res;
        return res;
}

int numSquares(int n)
{
        vector<int> memo(n + 1);
        return solve(n, memo);
}

int main()
{
        int n;
        cin >> n;
        cout << numSquares(n) << "\n";
        return 0;
}
```

Question No. 04

Write C++ program to solve House Robber II by using the Memoization method. **16**

Answer No. 04

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

int robHelper(vector<int>& nums, int start, int end, vector<int>& memo)
{
        if (start > end) {
        return 0;
        }
        if (memo[start] != -1) {
        return memo[start];
        }
        int prev1 = 0, prev2 = 0, curr = 0;
        for (int i = start; i <= end; i++) {
        curr = max(prev2 + nums[i], prev1);
        prev2 = prev1;
        prev1 = curr;
        }
        memo[start] = curr;
        return curr;
}

int rob(vector<int>& nums)
{
        int n = nums.size();
        if (n == 1) {
        return nums[0];
        }
        vector<int> memo(n, -1);
        int max1 = robHelper(nums, 0, n-2, memo);
        fill(memo.begin(), memo.end(), -1);
        int max2 = robHelper(nums, 1, n-1, memo);
        return max(max1, max2);
}

int main()
{
        int input;
        cout << "Number of inputs: ";
        cin >> input;
        vector<int> a;
        for(int i = 0; i < input; i++) {
        int num;
        cin >> num;
        a.push_back(num);
        }
        cout << rob(a) << "\n";
        return 0;
}
```

Question No. 05

Write C++ program to solve Grid Paths by using both Memoization and Tabulation methods. **16**

Answer No. 05

Memoization method:

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

const int mod = 1e9 + 7;
int n;
vector<string> grid;
vector<vector<int>> memo;

int countPaths(int i, int j) {
   if (i == n-1 && j == n-1) return 1;
```

```
    if (i >= n || j >= n || grid[i][j] == '*') return 0;
    if (memo[i][j] != -1) return memo[i][j];
    int paths = (countPaths(i+1, j) % mod + countPaths(i, j+1) % mod) % mod;
    memo[i][j] = paths;
    return paths;
}

int main() {
    cin >> n;
    grid.resize(n);
    memo.resize(n, vector<int>(n, -1));
    for (int i = 0; i < n; i++) cin >> grid[i];
    int ans = countPaths(0, 0);

    if (ans == 0) {
        cout << -1 << endl;
    } else {
        cout << ans << endl;
    }
    return 0;
}
```

Tabulation method:

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

int main()
{
        long long int n;
        cin>>n;
        string s;
        long long int mod = 1e9+7;

        vector<vector<long long int>> dp(n+1,vector<long long int> (n+1,0));
        dp[0][0] = 1;

        for(long long int i=0;i<n;i++)
        {
        cin>>s;
```

```
    for(long long int j=0;j<n;j++)
    {
    if(s[j] != '*')
    {
        if(i>0)
        {
        dp[i][j] += dp[i-1][j];
        dp[i][j] %= mod;
        }
        if(j>0)
        {
        dp[i][j] += dp[i][j-1];
        dp[i][j] %= mod;
        }
    }
    else
    {
        dp[i][j] = 0;
    }
    }
    }

    if (n > 0) {
    cout << dp[n-1][n-1] % mod;
    }
    else {
    cout << -1;
    }

    return 0;
}
```

Question No. 06

Write C++ program to solve King Escape by using both BFS and DFS. **20**

Answer No. 06

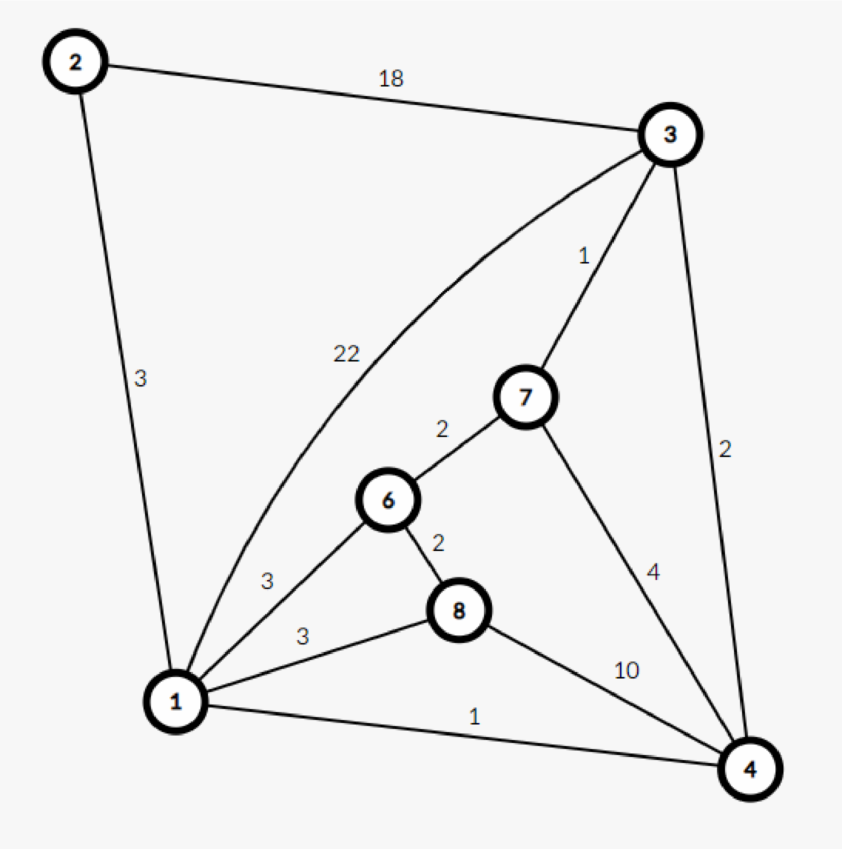
```
#include <bits/stdc++.h>
using namespace std;

const int N = 1e3 + 5;
int n, ax, ay, bx, by, cx, cy;
bool vis[N][N];

bool can_reach(int x, int y)
{
    if (x == ax || y == ay || abs(x - ax) == abs(y - ay)) return false;
    return true;
}
bool dfs(int x, int y)
{
    if (x == cx && y == cy) return true;
    vis[x][y] = true;
    for (int i = -1; i <= 1; i++) {
    for (int j = -1; j <= 1; j++) {
    int nx = x + i, ny = y + j;
    if (nx < 1 || nx > n || ny < 1 || ny > n || vis[nx][ny]) continue;
    if (can_reach(nx, ny)) {
        if (dfs(nx, ny)) return true;
    }
    }
    }
    return false;
}
bool bfs()
{
    queue<pair<int, int>> q;
    q.push(make_pair(bx, by));
    vis[bx][by] = true;
    while (!q.empty()) {
    pair<int, int> p = q.front();
    q.pop();
    int x = p.first, y = p.second;
    if (x == cx && y == cy) return true;
    for (int i = -1; i <= 1; i++) {
    for (int j = -1; j <= 1; j++) {
        int nx = x + i, ny = y + j;
        if (nx < 1 || nx > n || ny < 1 || ny > n || vis[nx][ny]) continue;
        if (can_reach(nx, ny)) {
        q.push(make_pair(nx, ny));
        vis[nx][ny] = true;
        }
    }
    }
    }
    return false;
}
int main()
{
    cin >> n >> ax >> ay >> bx >> by >> cx >> cy;
    if (dfs(bx, by) || bfs()) {
    cout << "YES\n";
    }
    else {
    cout << "NO\n";
    }
    return 0;
}
```

Question No. 07

Write the shortest distance from node 2 to every other node using the Dijkstra algorithm for the following graph .You must write all the steps. **10**



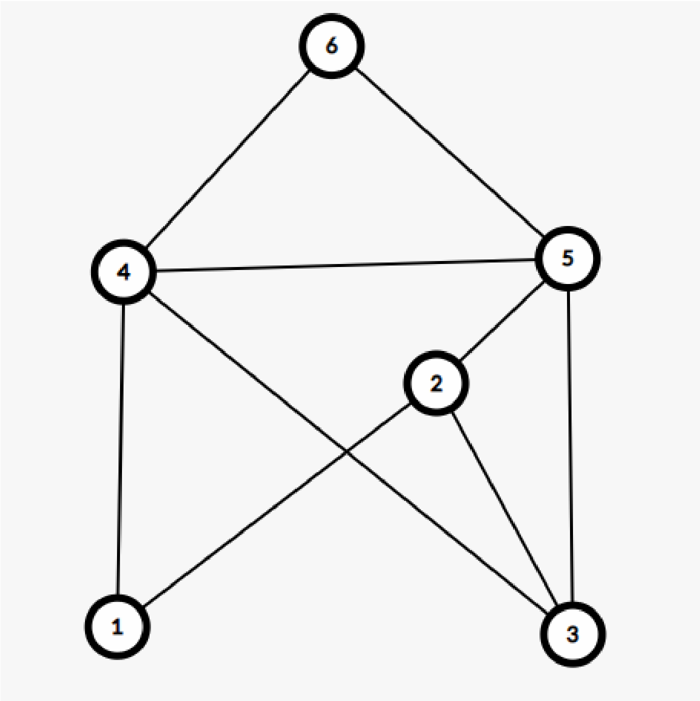
Answer No. 07

S.N	1	2	3	4	6	7	8
—	inf.	0	inf.	inf.	inf.	inf.	inf.
2	3	**0**	18	inf.	inf.	inf.	inf.
1	**3**	**0**	18	4	6	inf.	6
4	**3**	**0**	6	**4**	6	8	6
3	**3**	**0**	**6**	**4**	6	7	6
6	**3**	**0**	**6**	**4**	**6**	7	6

8	**3**	**0**	**6**	**4**	**6**	7	**6**
7	**3**	**0**	**6**	**4**	**6**	**7**	**6**

Question No. 08

Perform BFS Traversal on the following graph and write the traversal output. Choose node 2 as the source. You must write all the steps.
10



Answer No. 08

BFS Traversal:

Steps	**Queue**	**Visited Array**	**Output**
1	[2]	[2]	[ ]
2	[1, 3, 5]	[2, 1, 3, 5]	[2]

3	[3, 5, 4]	[2, 1, 3, 5, 4]	[2, 1]
4	[5, 4]	[2, 1, 3, 5, 4]	[2, 1, 3]
5	[4, 6]	[2, 1, 3, 5, 4, 6]	[2, 1, 3, 5]
6	[6]	[2, 1, 3, 5, 4, 6]	[2, 1, 3, 5, 4]
7	[ ]	[2, 1, 3, 5, 4, 6]	**[2, 1, 3, 5, 4, 6]**

BFS Traversal Output: [2, 1, 3, 5, 4, 6]