Name of the Problem: BFS

Code:

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+10;
vector<int>v[N];
int vis[N],level[N];
void bfs(int node)
{
    queue<int>q;
    q.push(node);
    vis[node]=1;
    level[node]=1;
    cout<<"BFS :- ";
    while(!q.empty())
    {
        int fr=q.front();
        cout<<fr<<" ";
        q.pop();
        for(auto x:v[fr])
        {
            if(!vis[x])
            {
                vis[x]=1;
               level[x]=level[fr]+1;
                q.push(x);
            }
        }
    }
    cout<<endl;
}
int main()
{
    cout<<"Enter the number of edge:
";
    int n;
    cin>>n;
    for(int i=0; i<n; i++)
    {
        int x,y;
        cin>>x>>y;
        v[x].push_back(y);
        v[y].push_back(x);
    }
    cout<<"\nAdjacency List: \n";
    for(int i=1; i<=n; i++)
    {
        cout<<i;
        for(auto x:v[i])cout<<"-
>"<<x;
        cout<<endl;
    }
    cout<<endl;
    bfs(1);
    cout<<endl;
    cout<<"Distance from source
node: \n";
    for(int i=1; i<=n; i++)
    {
        cout<<"\t\t\t"<<i<<"-
>"<<level[i]<<endl;
    }
}
```

Output:

```
"D:\CSE\Algorithm_Lab\A_N_S_30_383\bfs traverse and level or cost.exe"
Enter the number of edge: 5
1 2
2 3
2 4
4 5
4 6

Adjacency List:
1->2
2->1->3->4
3->2
4->2->5->6
5->4

BFS :- 1 2 3 4 5 6

Distance from source node:
                        1->1
                        2->2
                        3->3
                        4->3
                        5->4
```

Name of the Problem: 3) Subset Sum Problem Using BitMasking

Code:

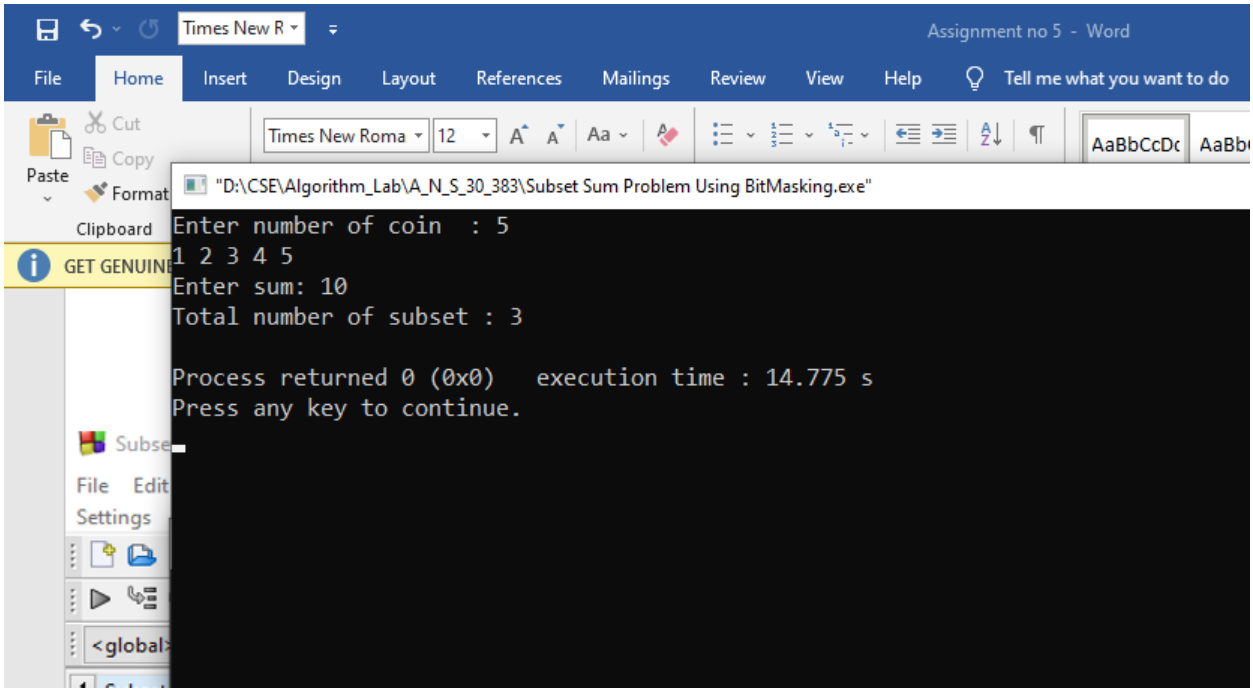
```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cout<<"Enter number of coin
: ";
    cin>>n;
    int ara[n];
    for(int i=0; i<n;
i++)cin>>ara[i];
    cout<<"Enter sum: ";
    int sum,count=0;
    cin>>sum;
```

```
for(int i=1; i<(1<<n); i++)
    {
        int x = i;
        int subsetsum = 0;
        for(int j=0; j<n; j++)
        {
            if( (x&(1<<j))!=0)

subsetsum+=ara[j];
        }
        if(sum==subsetsum)
        {
            count++;
        }
    }
    cout<<"Total number of subset
: "<<count<<endl;

    return 0;
}
```

OUTPUT:



Name of the Problem: 4) Subset Sum Problem Using BitMask & Meet in the middle technique.

Code:

```
#include<bits/stdc++.h>
using namespace std;
int SetOfElement[100];
void
culculation_sum_of_subset(
int n,int c,vector<int>&v)
{
    for(int i=0; i<(1<<n);
i++)
    {
        int sum=0;
        for(int j=0; j<n;
j++)
        {
           if(i&(1<<j))
            {

sum=sum+SetOfElement[j+c];
            }
        }
      v.push_back(sum);
    }
}
int main()
{
    vector<int>v1,v2;
    int n,count=0;
    cout<<"Enter the
number of elements: ";
    cin>>n;
    for(int i=0; i<n; i++)
    {
        cin>>SetOfElement[i];
    }
    cout<<"Enter wanting values: ";
    int values;
    cin>>values;
    culculation_sum_of_subset(n/2,0,v1);
    culculation_sum_of_subset((n+1)/2,(n/2),v2)
    sort(v1.begin(),v1.end());
    int sz=v1.size();
    for(int i=0; i<v2.size(); i++)
    {
        int seSetOfElementch=0;
        if(v2[i]<=values)
        {
            seSetOfElementch=values-v2[i];
            auto
it=lower_bound(v1.begin(),v1.end(),seSetOfEleme
ntch)-v1.begin();
            if(it==sz)
            {
                it--;
            }
            if(seSetOfElementch==v1[it])
            {
                count++;
            }
        }
    }
    cout<<"Total set of subSet :
"<<count<<endl;
}
```