BACKTRACKING :-

1) Sum of subsets :-

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

using namespace std;

void solve(vector<int> a, int b, vector<int> & temp,
           <vector<vector<int>>&vl, int k) {

    if (b==0){
        sort (temp.begin(), temp.end());
        vl.push_back(temp);
        return;
    }

    if(k==a.size()) return;

    if (a[k] <= b) {
        temp.push_back(a[k]);
        solve(a, b-a[k], temp, vl, k+1);
        temp.pop_back();
    }

    solve(a, b, temp, vl, k+1);

}
```

```
vector<vector<int>> SumOfSubset (vector<int>& a, int b){
        vector< int> temp;
        sort(a.begin(), a.end());
        vector<vector<int>> v1;
        solve (a,b, temp, v1, 0);
        return v1;
}
int main(){
    int n; cin>>n;
    int w; cin>>w; vector<int> arr(n);
    for(int i=0; i<n; i++) cin>>arr[i];
    vector<vector<int>> v1 = sumOfSubsets (arr, w);
    for(auto it:v1){
        for(auto j:it) cout<<j<<" ";
        cout<<endl;
    }
}
```

2) NQueens Problem:-

```
#include<bits/stdc++.h>
using namespace std;
void store(vector<vector<int>>& arr, vector<vector<int>>& v1,int n)
{
        vector<int> temp;
        for(int i=1; i<=n; i++){
            for(int j=1; j<=n; j++) {
                if (arr[i][j]==1) temp.push_back(j);
            }
        }
        v1.push_back(temp);
}
bool isSafe(vector<vector<int>>& arr, int row, int col, int n)
{
        for(int i=1; i<col; i++) {
            if(arr[row][i]==1) return false;
        }
        int i=row, j=col;
        while(i>1 and j>1){
            j--; i--;
            if(arr[i][j]==1) return false;
        }
```

```
    i=row; j=col;
    while(i<n and j>=1){
        j--; i++;
        if(arr[i][j]==1) return false;
    }
    return true;
}

void solve(int col, int n, vector<vector<int>>& arr,
           vector<vector<int>>& vl)
{
    if(col==n+1){
        store(arr, vl, n);
        return;
    }
    for(int i=1; i<=n; i++){
        if(isSafe(arr, i, col, n)){
            arr[i][col]=1;
            solve(col+1, n, arr, vl);
            arr[i][col]=0;
        }
    }
}
```

```
vector<vector<int>> nQueen(int n){
    vector<vector<int>> arr(n+1, vector<int>(n+1, 0)), vl;
    solve(1, n, arr, vl);
    return vl;
}

int main(){
    int n; cin>>n;
    vector<vector<int>> vl= nQueen(n);
    for(auto it: vl){
        for(auto x: it) cout<<x<<" ";
        cout<<endl;
    }
}
```

3) HamiltonCycle:-

```
#include<bits/stdc++.h>
using namespace std;
void print-permutation(int a[], int n, vector<vector<int>>&v)
{
    int cost=0;
    for(int i=0; i<=n; i++){
        cout<<a[i]<<" ";
        if(i<n) cost+=v[a[i]][a[i+1]];
    }
    cost+=v[a[n]][a[1]];
    cout<<" "<<"->"<<cost;
    cout<<endl;
}
bool isSafe(int a[], int key, vector<vector<int>>& V, int n)
{
    if(k==n){
        for(int i=1; i<k; i++){
            if(a[i]==key or v[a[k-1]][key]==0 or
                v[key][a[1]]==0)
                return 0;
        }
    }
    else{
        for(int i=1; i<k; i++){
            if(a[i]==key or v[a[k-1]][key]==0) return 0;
        }
    }
    return true;
}
void per(int a[], int k, int n, vector<vector<int>>& v)
{
    if(k==n+1){
        print-permutation(a,n,v);
        return;
    }
    for(int i=1; i<=n; i++){
        if(isSafe(a,i,k,v,n)){
            a[k]=i;
            per(a,k+1,n,v);
        }
    }
}
```

```
int main(){

    int n; cin>>n;
    int a[n];

    vector<vector<int>> v(n+1, vector<int>(n+1));

    for(int i=1; i<=n; i++){

        for(int j=0 ; j<=n; j++)  cin>>v[i][j];

    }

    per(a,1,n,v);

}
```

4) Knapsack:-

```
void print-permutation(int a[], int n, int w, int wt[], int val[])

{    int sum=0;

     for(int i=0; i<n; i++){

          if(a[i]==1) sum+=wt[i];

     }

     if(sum==w){

          for(int i=0; i<n; i++){

               if(a[i]==1) cout<<val[i]<<" ";

          }

          cout<<endl;

     }

}

void per(int a[], int k, int n, int w, int wt[], int val[])

{    if(k==n){

          print-permutation(a,n,w,wt,val);

          return;

     }

     for(int i=0; i<2; i++){

          a[k]=i;

          per(a,k+1,n,w,wt,val);

     }

}
```