9BACKTRACKING

1. PERMUTATION OF A STRING ‘ABC’ WHICH DOES NOT CONTAIN ‘AB’

```
#include<bits/stdc++.h>
using namespace std;
bool isSafe(string s,int l,int i,int r)
{
    if(l!=0 && s[l-1]=='A' && s[i]=='B')
    {
        return false;
    }
    if(r==l+1 && s[l]=='B' && s[i]=='A')
    {
        return false;
    }
    return true;
}
void permute(string s,int l,int r)
{
    if(l==r)
    {
        cout<<s<<" ";
        return;
    }
    else
    {
        for(int i=l;i<=r;i++)
        {
            if(isSafe(s,l,i,r))
            {
                swap(s[i],s[l]);
                permute(s,l+1,r);
```

```
                swap(s[i],s[l]);
            }
        }
    }
}
int main()
{
    string str="ABC";
    permute(str,0,str.length()-1);
    return 0;
}
```

2. RAT IN A MAZE

```
#include<bits/stdc++.h>
using namespace std;
#define N 4
bool solveMazeRec(int maze[N][N], int x, int y, int sol[N][N]);

void printsol(int sol[N][N])
{
    for(int i=0;i<N;i++)
    {
        for(int j=0;j<N;j++)
        {
            cout<<sol[i][j]<<" ";
        }
        cout<<endl;
    }
}
bool solveMaze(int maze[N][N])
{
    int sol[N][N] = { { 0, 0, 0, 0 },
                      { 0, 0, 0, 0 },
                      { 0, 0, 0, 0 },
```

```
                    { 0, 0, 0, 0 } };

    if(solveMazeRec(maze,0,0,sol)==false)
    {
        cout<<"No solution";
        return false;
    }


        printsol(sol);
        return true;

}
bool isSafe(int maze[N][N],int i,int j)
{
    if(i<N && j<N && maze[i][j]==1)
    {
        return true;
    }
    return false;
}
bool solveMazeRec(int maze[N][N],int i,int j,int sol[N][N])
{
    if(i==N-1 && j==N-1 && maze[i][j]==1)
    {
        sol[i][j]=1;
        return true;
    }
    if(isSafe(maze,i,j)==true)
    {
        sol[i][j]=1;
        if(solveMazeRec(maze,i+1,j,sol)==true)
        {
            return true;
        }
```

```
        else if(solveMazeRec(maze,i,j+1,sol)==true)
        {
            return true;
        }
        sol[i][j]=0;
    }
}
int main()
{
    int maze[N][N] = { { 1, 0, 0, 0 },
                       { 1, 1, 0, 1 },
                       { 0, 1, 0, 0 },
                       { 1, 1, 1, 1 } };

    solveMaze(maze);
    return 0;
}
```

3. N QUEENS PROBLEM

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

#define N 4

int board[N][N];

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            printf(" %d ", board[i][j]);
        printf("\n");
    }
}
```

```
bool isSafe(int row, int col)
{

    for (int i = 0; i < col; i++)
        if (board[row][i])
            return false;

    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;

    for (int i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;

    return true;
}

bool solveRec(int col)
{
    if (col == N)
        return true;

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

        if (isSafe(i, col)) {
            board[i][col] = 1;

            if (solveRec(col + 1))
                return true;

            board[i][col] = 0;
        }
    }
```

```
    return false;
}

bool solve()
{
    if (solveRec(0) == false) {
        printf("Solution does not exist");
        return false;
    }

    printSolution(board);
    return true;
}

int main() {

        solve();
    return 0;

}
```

4. **SUDOKU**

```
#include<bits/stdc++.h>
using namespace std;
#define N 9
bool isSafe(int board[N][N],int row,int col,int num)
{
    for(int i=0;i<N;i++)
    {
        if(board[row][i]==num)
        {
            return false;
        }
```

```
    }
    for(int i=0;i<N;i++)
    {
        if(board[i][col]==num)
        {
            return false;
        }
    }
    int s = (int)sqrt(N);
    int rs = row - row%s;
    int cs = col - col%s;
    for(int i=rs;i<rs+s;i++)
    {
        for(int j=cs;j<cs+s;j++)
        {
            if(board[i][j]==num)
            {
                return false;
            }
        }
    }
    return true;
}
void printGrid(int grid[N][N])
{
    for(int i=0;i<N;i++)
    {
        for(int j=0;j<N;j++)
        {
            cout<<grid[i][j]<<" ";
        }
        cout<<endl;
    }
}
bool solve(int board[N][N],int n)
```

```
{
    int row=-1;
    int col=-1;
    bool isEmpty = true;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(board[i][j]==0)
            {
                row=i;
                col=j;
                isEmpty=false;
                break;
            }
        }
        if(!isEmpty)
        {
            break;
        }
    }
    if(isEmpty)
    {
        return true;
    }
    for(int num=1;num<=n;num++)
    {
        if(isSafe(board,row,col,num))
        {
            board[row][col]=num;
            if(solve(board,n))
            {
                return true;
            }
            else
```

```
            {
                board[row][col]=0;
            }
        }
    }
    return false;
}
int main()
{
        int grid[N][N] = { { 3, 0, 6, 5, 0, 8, 4, 0, 0 },
                                        { 5, 2, 0, 0, 0, 0, 0, 0, 0 },
                                        { 0, 8, 7, 0, 0, 0, 0, 3, 1 },
                                        { 0, 0, 3, 0, 1, 0, 0, 8, 0 },
                                        { 9, 0, 0, 8, 6, 3, 0, 0, 5 },
                                        { 0, 5, 0, 0, 9, 0, 6, 0, 0 },
                                        { 1, 3, 0, 0, 0, 0, 2, 5, 0 },
                                        { 0, 0, 0, 0, 0, 0, 0, 7, 4 },
                                        { 0, 0, 5, 2, 0, 6, 3, 0, 0 } };
        if (solve(grid,N) == true)
                printGrid(grid);
        else
                cout << "No solution exists";

        return 0;
}
```

5. LARGEST NUMBER IN K SWAPS

```
class Solution
{
    public:
    //Function to find the largest number after k swaps.
    void findmax(string str,string &ans,int k,int pos)
    {
        if(k==0)
```

```
    {
        return;
    }
    char maxm = str[pos];
    for(int i=pos+1;i<str.length();i++)
    {
        if(maxm<str[i])
        {
            maxm=str[i];
        }
    }
        if(maxm!=str[pos])
        {
            k--;
        }
        for(int i=pos;i<str.length();i++)
        {
            if(str[i]==maxm)
            {
                swap(str[i],str[pos]);
                ans = max(ans,str);
                findmax(str,ans,k,pos+1);
                swap(str[i],str[pos]);
            }
        }

    return;
}
string findMaximumNum(string str, int k)
{
   // code here.
   if(k==0)
   {
       return str;
   }
```

```
        string ans="";
        findmax(str,ans,k,0);
        return ans;
    }
};
```

6. BLACK AND WHITE

```
long long numOfWays(int N, int M)
{
    // write code here
    int sub_res = 1;
    int total = M*N;
    long long res=0;
    for(int i=0;i<N;i++)
    {
        for(int j=0;j<M;j++)
        {
            if(i-2>=0 && j+1<M)
            {
                sub_res++;
            }
            if(i-1>=0 && j+2<M)
            {
                sub_res++;
            }
            if(i+1<N && j+2<M)
            {
                sub_res++;
            }
            if(i+2<N && j+1<M)
            {
                sub_res++;
            }
            if(i+1<N && j-2>=0)
```

```
                {
                    sub_res++;
                }
                if(i+2<N && j-1>=0)
                {
                    sub_res++;
                }
                if(i-1>=0 && j-2>=0)
                {
                    sub_res++;
                }
                if(i-2>=0 && j-1>=0)
                {
                    sub_res++;
                }
                res = (res+total-sub_res)%1000000007;
                sub_res=1;
            }
        }
        return res;
    }
```

7.