5) SUDOKU SOLVER:

METHOD:

How to Solve Sudoku using Backtracking | Recursion

1) **The first thing in this question is that, for placing any number into the grid, it must satisfy the condition of sudoku i.e. there should not be same element in the same row and column and also in the same subgrid.**
2) **So we need to have a function for checking this property.**

Function code:

```
bool isValid(int grid[N][N],int i,int j,int element){
      for(int row=0;row<9;row++){
         if(grid[i][row]==element){
            return false;
         }
      }
      for(int col=0;col<9;col++){
         if(grid[col][j]==element){
            return false;
         }
      }
      int rowLimit=(i/3)*3;
      int colLimit=(j/3)*3;
      for(int row=rowLimit;row<rowLimit+3;row++){
         for(int col=colLimit;col<colLimit+3;col++){
            if(grid[row][col]==element){
               return false;
            }
         }
      }
      return true;
   }
```

3) **Now the important thing in this question is recursive function.The recursive function will have 3 arguments , one is the grid and the other two are the coordinates of the box the recursion is currently on.**
4) **So if i and j both becomes equal to 9 then it means that previously all the conditions for all the boxes must have been satisfied and that is why the recursion reached here.Otherwise it would have backtracked earlier.**
5) **If j exceeds 9 this means that the a particular row has been completed and we have to move on to the next row, so we do i = i + 1 and j = 0**
6) **Else we check whether the position is zero or not.if zero then we check all the possible values from 1 to 9. If we particular value satisfies then recursively we go on next position and after the call we have to make that position again zero so that while backtracking no values of previous functions remains.**

```
void fun(int grid[N][N],int i,int j){
      if(i>=9 && j>=9){
         for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
```

```
                    answer[i][j]=grid[i][j];
                }
            }
            return;
        }
        else if(j>=9){
            fun(grid,i+1,0);
            return;
        }
        else{
            if(grid[i][j]!=0){
                fun(grid,i,j+1);
                return;
            }
            else{
                for(int it=1;it<=9;it++){
                    if(isValid(grid,i,j,it)){
                        grid[i][j]=it;
                        fun(grid,i,j+1);
                        grid[i][j]=0;
                    }
                }
                return;
            }
        }
    }
```

COMPLETE CODE:

```
class Solution
{
    public:
    //Function to find a solved Sudoku.
    int answer[N][N];
    bool isValid(int grid[N][N],int i,int j,int element){
        for(int row=0;row<9;row++){
            if(grid[i][row]==element){
                return false;
            }
        }
        for(int col=0;col<9;col++){
            if(grid[col][j]==element){
                return false;
            }
        }
        int rowLimit=(i/3)*3;
        int colLimit=(j/3)*3;
        for(int row=rowLimit;row<rowLimit+3;row++){
```

```
        for(int col=colLimit;col<colLimit+3;col++){
            if(grid[row][col]==element){
                return false;
            }
        }
    }
    return true;
}
void fun(int grid[N][N],int i,int j){
    if(i>=9 && j>=9){
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                answer[i][j]=grid[i][j];
            }
        }
        return;
    }
    else if(j>=9){
        fun(grid,i+1,0);
        return;
    }
    else{
        if(grid[i][j]!=0){
            fun(grid,i,j+1);
            return;
        }
        else{
            for(int it=1;it<=9;it++){
                if(isValid(grid,i,j,it)){
                    grid[i][j]=it;
                    fun(grid,i,j+1);
                    grid[i][j]=0;
                }
            }
            return;
        }
    }
}
bool SolveSudoku(int grid[N][N]){
    fun(grid,0,0);
    for(int i=0;i<N;i++){
        for(int j=0;j<N;j++){
            grid[i][j]=answer[i][j];
        }
    }
    return true;
}

//Function to print grids of the Sudoku.
void printGrid (int grid[N][N]){
```

```
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                cout<<grid[i][j]<<" ";
            }
        }
    }
};
```