|  |  |
| --- | --- |
| **Course Code: CT2352** | **Course Name: Lab-DAA** |

|  |  |
| --- | --- |
| **Name of Students: S Akshansh** | **Semester/ Section: 6 A** |
| **Roll No: 72** | **Enroll No: 19010927** |

**Practical No 9**

|  |
| --- |
| **Aim**: Write a program to show the complete simulation of 8 queen’s problem by satisfying all it constraints using backtracking. |
| **Theory:-**  The eight queens problem is the problem of placing eight queens on an 8×8 chessboard such that none of them attack one another (no two are in the same row, column, or diagonal). More generally, the n queens problem places n queens on an n×n chessboard.  Let’s try an example, with four queens and a small board. We will start by placing the first queen:    In the 4 queens problem, the chessboard is proportionally smaller (4×4). The crosses show where queens won’t be in a safe position.  The second step is to place the second queen in a safe position, and then the third one:    We place the second queen on the first safe square we find. After we place the third queen, there is no place where we can put the fourth queen safely.  At this point, there is no safe square on which we can place the last queen. So, we will change the position of the previous one. This is backtracking!    Backtracking.  As you can see, there is no other square to safely place the third queen, different than the one we tried. Hence, this will require multiple backtracks. We will have to go back *again* and change the position of the second queen we placed.    We placed the third queen. Again, there is no safe place for a fourth queen!  We will have to perform multiple backtracks again to place all queens correctly and finally obtain a solution:    **ALGORITHM:**  1) Start in the leftmost column  2) If all queens are placed  return true  3) Try all rows in the current column.  Do following for every tried row.  a) If the queen can be placed safely in this row  then mark this [row, column] as part of the  solution and recursively check if placing  queen here leads to a solution.  b) If placing the queen in [row, column] leads to  a solution then return true.  c) If placing queen doesn't lead to a solution then  unmark this [row, column] (Backtrack) and go to  step (a) to try other rows.  4) If all rows have been tried and nothing worked,  return false to trigger backtracking. |
| **Code:-**  #include <bits/stdc++.h>  using namespace std;  int board[8][8];  bool isPossible(int n,int row,int col){  for(int i=row-1;i>=0;i--){  if(board[i][col] == 1){  return false;  }  }  for(int i=row-1,j=col-1;i>=0 && j>=0 ; i--,j--){  if(board[i][j] ==1){  return false;  }  }  for(int i=row-1,j=col+1;i>=0 && j<n ; i--,j++){  if(board[i][j] == 1){  return false;  }  }  return true;  }  void nQueenHelper(int n,int row){  if(row==n){  for(int i=0;i<n;i++){  for(int j=0;j<n;j++){  cout << board[i][j] << " ";  }cout<<endl;  }  cout<<endl;  return;  }  for(int j=0;j<n;j++){  if(isPossible(n,row,j)){  board[row][j] = 1;  nQueenHelper(n,row+1);  }  board[row][j] = 0;  }  return;  }  void placeNQueens(int n){  memset(board,0,8\*8\*sizeof(int));  nQueenHelper(n,0);  }  int main(){  int n;  cout<<"Enter No of Queens:- ";  cin>>n;  placeNQueens(n);  return 0;  } |
| **Output:-** |
| **Conclusion:** Thus, I have learnt about 8 queen’s problem. |