**AIM:-** To write a program to implement N-queens problem using backtracking.

**ALGORITHM**:-

1. Place the queens column wise, start from the left most column
2. If all queens are placed.
   1. return true and print the solution matrix.
3. Else
   1. Try all the rows in the current column.
   2. Check if queen can be placed here safely if yes mark the current cell in solution matrix as 1 and try to solve the rest of the problem recursively.
   3. If placing the queen in above step leads to the solution return true.
   4. If placing the queen in above step does not lead to the solution , BACKTRACK, mark the current cell in solution matrix as 0 and return false.
4. If all the rows are tried and nothing worked, return false and print NO SOLUTION.

**CODE:-**

#include<stdio.h>

#include<math.h>

int board[20],count;

int main()

{

int n,i,j;

void queen(int row,int n);

printf(" - N Queens Problem Using Backtracking -");

printf("\n\nEnter number of Queens:");

scanf("%d",&n);

queen(1,n);

return 0;

}

//function for printing the solution

void print(int n)

{

int i,j;

printf("\n\nSolution %d:\n\n",++count);

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

  printf("\t%d",i);

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

{

  printf("\n\n%d",i);

  for(j=1;j<=n;++j) //for nxn board

  {

   if(board[i]==j)

    printf("\tQ"); //queen at i,j position

   else

    printf("\t-"); //empty slot

  }

}

}

/\*funtion to check conflicts

If no conflict for desired postion returns 1 otherwise returns 0\*/

int place(int row,int column)

{

int i;

for(i=1;i<=row-1;++i)

{

  //checking column and digonal conflicts

  if(board[i]==column)

   return 0;

  else

   if(abs(board[i]-column)==abs(i-row))

    return 0;

}

return 1; //no conflicts

}

//function to check for proper positioning of queen

void queen(int row,int n)

{

int column;

for(column=1;column<=n;++column)

{

  if(place(row,column))

  {

   board[row]=column; //no conflicts so place queen

   if(row==n) //dead end

    print(n); //printing the board configuration

   else //try queen with next position

    queen(row+1,n);

  }

}

}

**OUTPUT**:-

