

# **CSE 2003: Lab Assignment #11**

Due on Thursday, April 13, 2017

*Prof. Shaik Naseera 2:00pm*

**Jacob John**

## Contents

Problem 1	3
-----------	---

## Problem 1

Write a program to generate solution for N queens problem using Backtracking approach using C

Listing 1: N queens problem using Backtracking approach using C

```
#include<stdio.h>
#include<stdlib.h>

int board[20],count;

5
int main()
{
    int n,i,j;
    void queen(int row,int n);
10
    printf(" - N Queens Problem Using Backtracking -");
    printf("\n\nEnter number of Queens:");
    scanf("%d",&n);
    queen(1,n);
15
    return 0;
}

//function for printing the solution
void print(int n)
20
{
    int i,j;
    printf("\n\nSolution %d:\n\n",++count);

    for(i=1;i<=n;++i)
25
        printf("\t%d",i);

    for(i=1;i<=n;++i)
    {
        printf("\n\n%d",i);
30
        for(j=1;j<=n;++j) //for nxn board
        {
            if(board[i]==j)
                printf("\tQ"); //queen at i,j position
            else
35
                printf("\t-"); //empty slot
        }
    }
}

40
/*funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0*/
int place(int row,int column)
{
    int i;
45
    for(i=1;i<=row-1;++i)
    {
        //checking column and digonal conflicts
```

```

    if (board[i]==column)
        return 0;
50  else
    if (abs (board[i]-column)==abs (i-row))
        return 0;
    }

55  return 1; //no conflicts
}

//function to check for proper positioning of queen
void queen(int row,int n)
60  {
    int column;
    for (column=1;column<=n;++column)
    {
        if (place (row,column))
65  {
            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
70  queen (row+1,n);
        }
    }
}

```

## Output:

```

- N Queens Problem Using Backtracking -
Enter number of Queens:4

Solution 1:
  1   2   3   4
1  -   Q   -   -
2  -   -   -   Q
3  Q   -   -   -
4  -   -   Q   -

Solution 2:
  1   2   3   4
1  -   -   Q   -
2  Q   -   -   -
3  -   -   -   Q
4  -   Q   -   -

-Jacobs-MacBook-Pro:Downloads jacobjohn$ ./a.out
- N Queens Problem Using Backtracking -
Enter number of Queens:1

Solution 1:
  1
1  Q
-Jacobs-MacBook-Pro:Downloads jacobjohn$ clear
-Jacobs-MacBook-Pro:Downloads jacobjohn$ ./a.out
- N Queens Problem Using Backtracking -
Enter number of Queens:4

```

```

n_queen_backtracking.c  knapsack_greedy.c  +
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int board[20],count;
5
6  int main()
7  {
8  int n,i,j;
9  void queen(int row,int n);
10
11  printf(" - N Queens Problem Using Backtracking -");
12  printf("\n\nEnter number of Queens:");
13  scanf("%d",&n);
14  queen(1,n);
15  return 0;
16  }
17
18  //function for printing the solution
19  void print(int n)
20  {
21  int i,j;
22  printf("\n\nSolution %d:\n\n",++count);
23
24  for(i=1;i<=n;i++)
25  printf("\t%d",i);
26
27  for(i=1;i<=n;i++)
28  {
29  printf("\n\n%d",i);
30  for(j=1;j<=n;j++) //for nxn board
31  {
32  if(board[i]==j)
33  printf("\tQ"); //queen at i,j position
34  else
35  printf("\t-"); //empty slot
36  }
37  }
38
39
40  //function to check conflicts
41  if no conflict for desired position returns 1 otherwise returns 0
42  int place(int row,int column)
43  {
44  int i;
45  for(i=1;i<=row-1;i++)
46  {
47  if(abs(board[i]-column)==abs(i-row))
48  return 0;
49  }
50  return 1;
51  }
52
53  void queen(int row,int n)
54  {
55  int column;
56  for (column=1;column<=n;++column)
57  {
58  if (place (row,column))
59  {
60  board[row]=column;
61  if (row==n)
62  print (n);
63  else
64  queen (row+1,n);
65  }
66  }
67  }

```