

## LAB PROGRAM 11

**AIM: Implement "N-Queens Problem" using Backtracking.**

### SOURCE 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;
}

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)
        {
            if(board[i]==j)
                printf("\tQ");
            else
                printf("\t-");
        }
    }
}

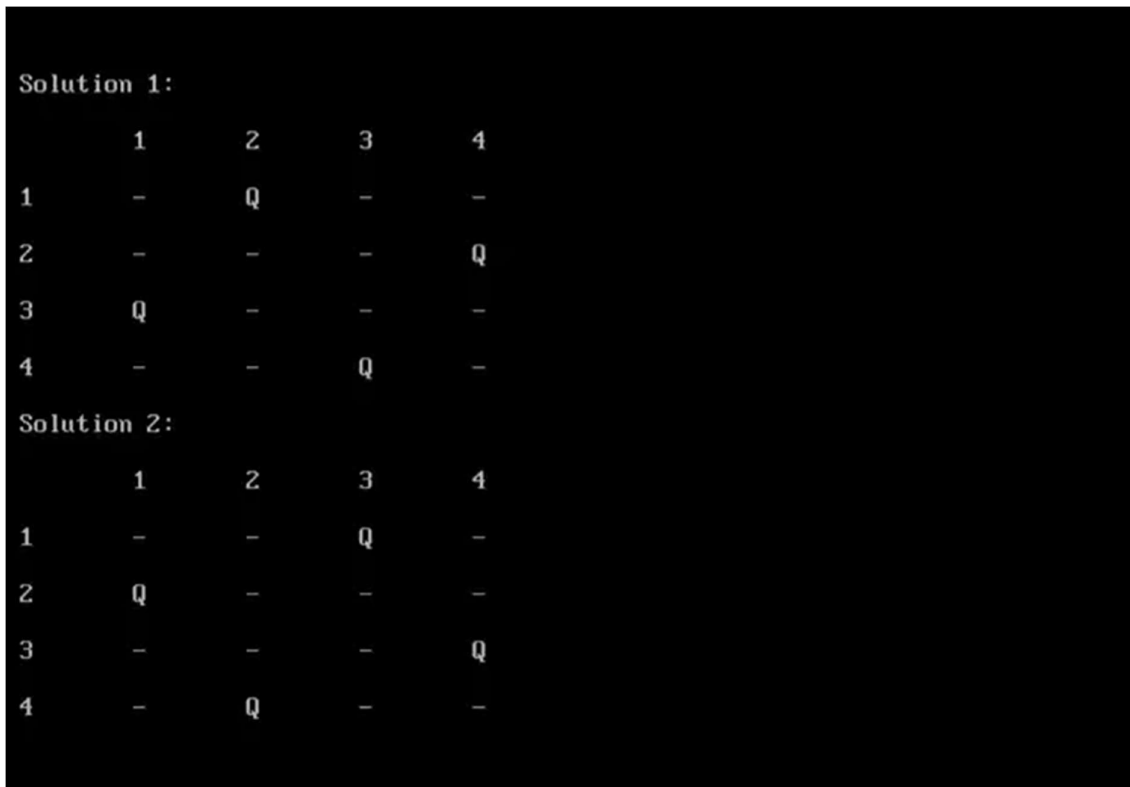
int place(int row,int column)
{
    int i;
    for(i=1;i<=row-1;++i)
    {
        if(board[i]==column)
            return 0;
    }
}
```

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

return 1;
}

void queen(int row,int n)
{
    int column;
    for(column=1;column<=n;++column)
    {
        if(place(row,column))
        {
            board[row]=column;
            if(row==n)
                print(n);
            else
                queen(row+1,n);
        }
    }
}
```

## OUTPUT SCREENSHOT



```
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      -      -
```