**EXPERIMENT N0-7**

**AIM: Implementation of 8-QUEEN problem using backtracking**

**CODE:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int a[30],count=0;

int place(int pos) {

int i;

for (i=1;i<pos;i++) {

if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))

return 0;

}

return 1;

}

void print\_sol(int n) {

int i,j;

count++;

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

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

for (j=1;j<=n;j++) {

if(a[i]==j)

printf("Q\t"); else

printf("\*\t");

}

printf("\n");

}

}

void queen(int n) {

int k=1;

a[k]=0;

while(k!=0) {

a[k]=a[k]+1;

while((a[k]<=n)&&!place(k))

a[k]++;

if(a[k]<=n) {

if(k==n)

print\_sol(n); else {

k++;

a[k]=0;

}

} else

k--;

}

}

void main() {

int i,n;

printf("Enter the number of Queens\n");

scanf("%d",&n);

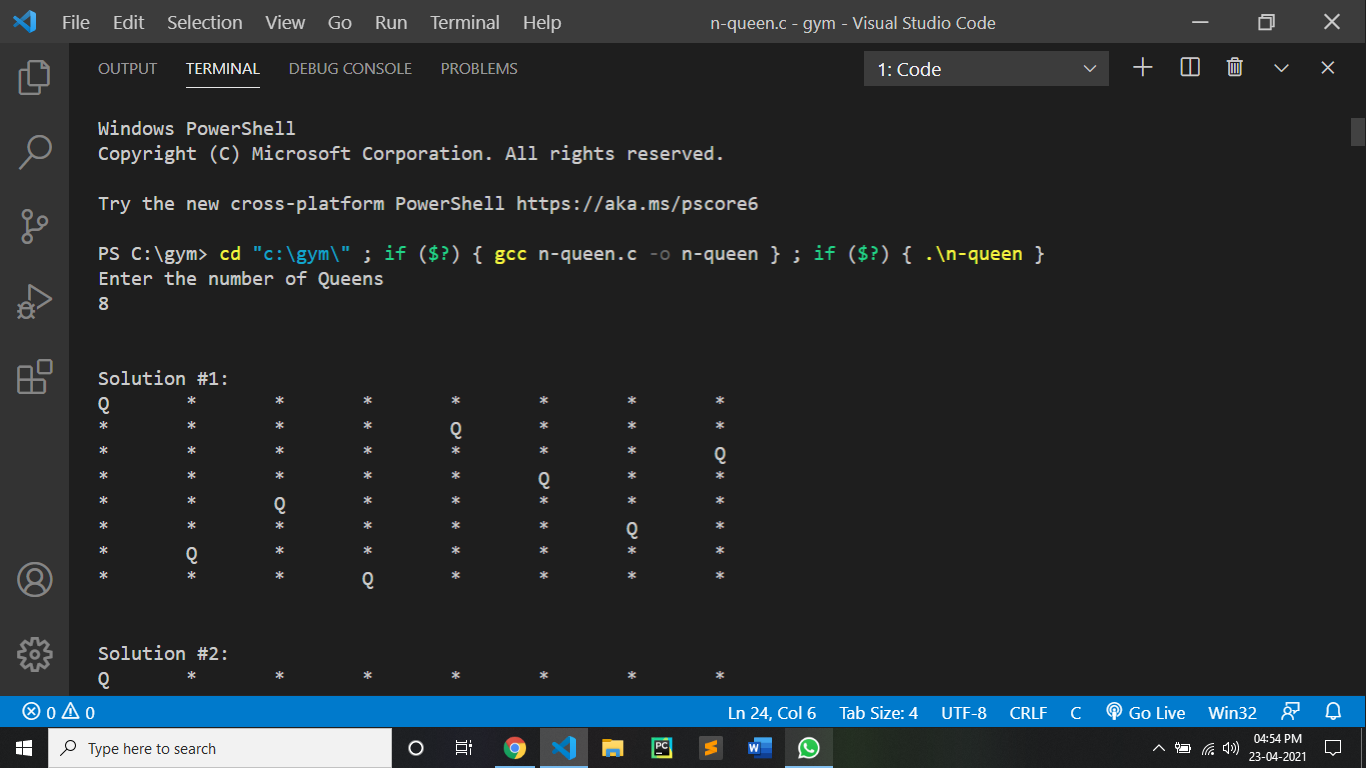
queen(n);

printf("\nTotal solutions=%d",count);

getch();

}

**OUTPUT:**



CONCLUSION:

By performing the n-queen problems we can conclude:

* In backtracking, at each level branching factor decreases by 1 and it creates a new problem of size (n-1). With n choices, it creates n different problems of size (n-1) at level 1.
* Place function determines the position of the queen in O(n) time. This function is called n times.
* Thus, the recurrence of n-Queen problem is defined as,

T(n) = n\*T(n-1)+n^2. Solution to recurrence would be O(n!).