**AIM:** Implement “N-Queens Problem” using Backtracking.

Note: In the record book students should

- Handwrite the Algorithm,

- Handwrite the Program

- Pasting of the printout of the Output or handwrite the Output

**ALGORITHM**: nqueens(n)

//places n queens on a nXn matrix such that no two queens are placed along same row or same column

//or same diagonal using backtracking method

//Input: n-number of queens

//Output: k-the queen k

// x[k]-the position of queen k

k🡨1

x[k]🡨0

**while** k!=0 **do**

x[k]🡨x[k]+1

**while** place(x,k)!=1 **and** x[k]<=n **do**

x[k]🡨x[k]+1

**end while**

**if** x[k]<=n

**if** k=n

**for** k🡨1 to n **do**

write k,x[k]

**end for**

**else**

k🡨k+1

x[k]🡨0

**end if**

**else**

k🡨k-1

**end if**

**end while**

**ALGORITHM**: place(x[k],k)

//places n queens on a nXn matrix such that no two queens are placed along same row or same column

//or same diagonal using backtracking method

//Input: k-the queen k

// x[k]- position of queen k

//Output: returns 0 if any two queens are placed along same diagonal or same column otherwise 1 is returned

**for** i🡨1 to k-1 **do**

**if** i-x[i]=k-x[k] **or** i+x[i]=k+x[k] **or** x[i]=x[k]

**return** 0

**end if**

**end for**

return 1

#include<stdio.h>

#include<conio.h>

int canplace(int r,int c[50])

{

int i;

for(i=0;i<r;i++)

{

if(c[i]==c[r] || abs(c[i]-c[r])==abs(i-r))

return 0;

}

return 1;

}

void display(int c[50],int n)

{

int i,j;

char cb[10][10];

for(i=0;i<n;i++)

for(j=0;j<n;j++)

cb[i][j]='-';

for(i=0;i<n;i++)

cb[i][c[i]]='q';

printf("-----------------------------\n");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%c\t ",cb[i][j]);

}

printf("\n");

}

}

void nqueen(int n)

{

int r,c[50];

c[0]=-1;

r=0;

while(r>=0)

{

c[r]++;

while(c[r]<n && !canplace(r,c))

c[r]++;

if(c[r]<n)

{

if(r==n-1)

{

display(c,n);

printf("\n");

}

else

{

r++;

c[r]=-1;

}

}

else

r--;

}

}

void main()

{

int n;

clrscr();

printf("enter the number of queens\n");

scanf("%d",&n);

nqueen(n);

getch();

}

**Output:**

Enter the number of Queens

4

Solution is 1

Queen Position

1 2

2 4

3 1

4 3

Solution is 2

Queen Position

1 3

2 1

3 4

4 2

**AIM**: Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution.

**ALGORITHM:** subset(s[1….n],d)

// To find subsets of a given set of n positive integers whose sum is equal to a given positive integer *d*

//Input: An array s[1….n] of sorted elements

// d-required sum

//Output: subsets of given set s[1….n] whose elements sum is equal to *d*

x[k]🡨1

if m+s[k]=d

write ‘subset solution is’, count🡨count+1

for i🡨0 to k do

if x[i]=1

write s[i]

end if

end for

else if m+s[k]+s[k+1]<=d

subset(m+s[k],k+1,sum-s[k])

end if

if m+sum-s[k]>=d and m+s[k+1]<=d

x[k]🡨0

subset(m,k+1,sum-s[k])

end if

**Program:**

#include<stdio.h>

#include<conio.h>

intcount,w[10],d,x[10];

void subset(intcs, int k, int r)

{

int i;

x[k]=1;

if(cs+w[k]==d)

{

printf("\nSubset solution = %d\n", ++count);

for(i=0;i<=k;i++)

{

if(x[i]==1)

printf("%d", w[i]);

}

}

else

if(cs+w[k]+w[k+1]<=d)

subset(cs+w[k], k+1, r-w[k]);

if((cs+r-w[k]>=d) && (cs+w[k+1])<=d)

{

x[k]=0;

subset(cs,k+1,r-w[k]);

}

}

void main()

{

int sum=0,i,n;

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

scanf("%d", &n);

printf("Enter the elements in ascending order\n");

for(i=0;i<n;i++)

scanf("%d", &w[i]);

printf("Enter the required sum\n");

scanf("%d", &d);

for(i=0;i<n;i++)

sum+=w[i];

if(sum<d)

{

printf("No solution exists\n");

return;

}

printf("The solution is\n");

count=0;

subset(0,0,sum);

getch();

}

**Output:**

Enter the number of elements

5

Enter the elements in ascending order

1

2

5

6

8

Enter the required sum

9

The solution is

Subset solution = 1

1 2 6

Subset solution = 2

1 8