Index

Problem	Remarks
1. Implementation Of Quicksort	
2. Implementation Of Insertion Sort	
3. Implementation Of Merge Sort	
4. Implementation Of Fibonacci Series	
5. Implementation Of Binary Search	
6. Fractional Knapsack using Dynamic Programming	
7. cxMinimum cost spanning tree using Kruskal's Algorithm	
8. Implementation Of Euclidean's Algorithm	
9. Implementation Of Matrix Chain Multiplication	
10. Implementation Of Chinese Remainder Theorem	
11. Solving N-Queen Problem using Backtracking	

Quicksort

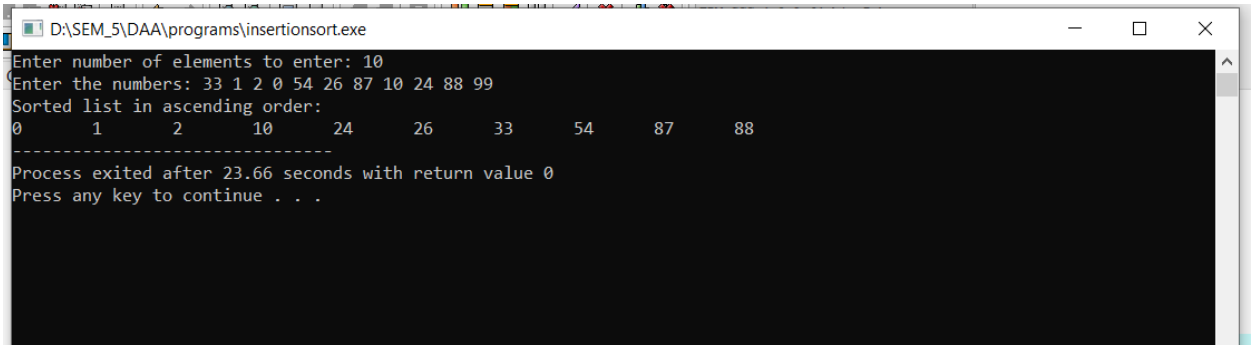
```
#include<stdio.h>
void quicksort(int number[25],int first, int last)
{
int i, j, pivot, temp;
if(first<last)
{
pivot=first;
i=first;
j=last;
while(i<j)
{
while(number[i]<=number[pivot]&&i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j)
{
temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}
}
int main(){
int i, count, number[25];
printf("Enter the number of elements: ");
scanf("%d",&count);
printf("Enter %d elements: ", count);
for(i=0;i<count;i++)
scanf("%d",&number[i]);
quicksort(number,0,count-1);
printf("Sorted elements: ");
for(i=0;i<count;i++)
printf(" %d",number[i]);
return 0;
}
```

D:\SEM_5\DAA\programs\quicksort.exe

```
Enter the number of elements: 7
Enter 7 elements: 3
4
5
6
1
2
0
Sorted elements:  0 1 2 3 4 5 6
--------------------------------
Process exited after 8.543 seconds with return value 0
Press any key to continue . . .
```

Insertion Sort

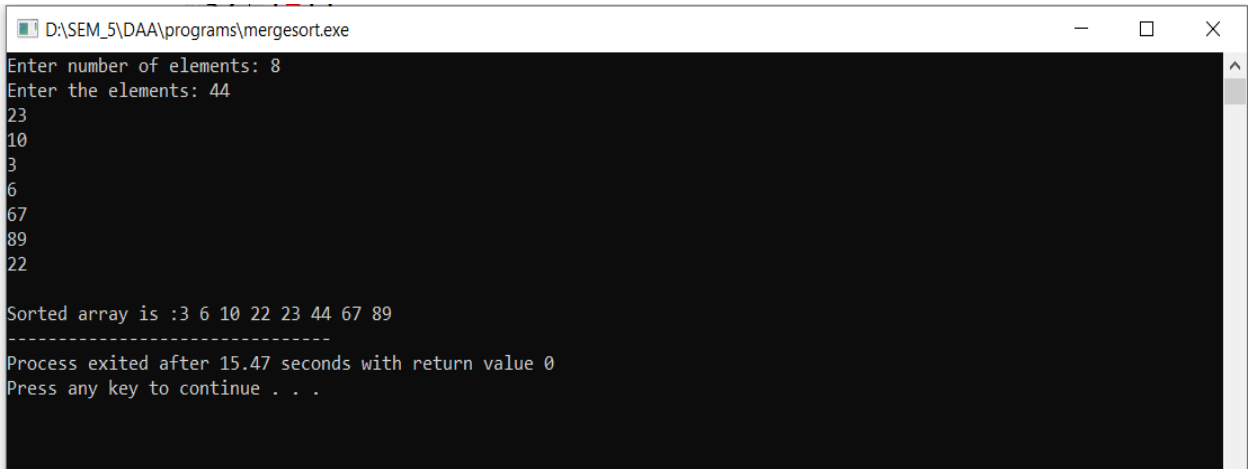
```
#include<stdio.h>
int main()
{
int a[100],i,j,n,s;
printf("Enter number of elements to enter: ");
scanf("%d",&n);
printf("Enter the numbers: ",n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for (i=1;i<n;i++)
{ s=a[i];
}
j=i-1;
while(j>=0 && a[j]>s)
{
a[j+1]=a[j];
j=j-1;
a[j+1]=s;
}
}
printf("Sorted list in ascending order:\n");
for(i=0;i<n;i++)
{
printf("%d\t", a[i] );
}
```



Merge Sort

```
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int
i2,int j2);
int main()
{
int a[30],n,i;
printf("Enter number of elements:");
scanf("%d",&n);
printf("Enter the elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
mergesort(a,0,n-1);
printf("\nSorted array is :");
for(i=0;i<n;i++)
printf("%d ",a[i]);
return 0;
}
void mergesort(int a[],int i,int j)
{
int mid;
if(i<j)
{
mid=(i+j)/2;
mergesort(a,i,mid);
mergesort(a,mid+1,j);
merge(a,i,mid,mid+1,j); }
}
void merge(int a[],int i1,int j1,int
i2,int j2)
{
int temp[50];
int i,j,k;
i=i1;
j=i2;
k=0;
while(i<=j1 && j<=j2)
{
if(a[i]<a[j])
temp[k++]=a[i++];
else
temp[k++]=a[j++];
}
while(i<=j1)
temp[k++]=a[i++];
while(j<=j2)
temp[k++]=a[j++];
for(i=i1,j=0;i<=j2;i++,j++)
```

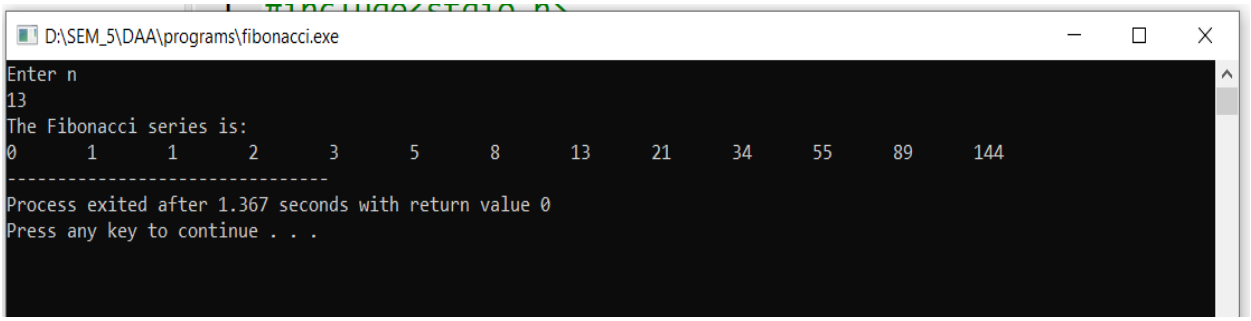
```
a[i]=temp[j];
}
```


D:\SEM_5\DAA\programs\mergesort.exe
Enter number of elements: 8
Enter the elements: 44
23
10
3
6
67
89
22
Sorted array is :3 6 10 22 23 44 67 89

Process exited after 15.47 seconds with return value 0
Press any key to continue . . .

Fibonacci

```
#include<stdio.h>
#include<conio.h>
int main()
{
	int a=0,b=1;
	int c,n,i;
	printf("Enter n\n");
	scanf("%d",&n);
	printf("The Fibonacci series is: \n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a);
		c = a+b;
		a=b;
		b=c;
	}
}
```



Binary Search

```
#include <stdio.h>

int main()

{

int c, first, last, middle, n, search,
array[100];

printf("Enter number of elements to
be entered: \n");

scanf("%d", &n);

printf("Enter %d numbers in sorted
order: \n", n);

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

scanf("%d", &array[c]);

printf("Enter value to search: \n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last)

{

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search)

{

printf("%d found at location %d.\n",
search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found! %d isn't present in
the list.\n", search);

return 0;

}
```

D:\SEM_5\DAA\programs\binarysearch.exe

```
Enter number of elements to be entered:
10
Enter 10 numbers in sorted order:
12 17 29 33 46 58 65 80 95 98
Enter value to search:
65
65 found at location 7.

--------------------------------
Process exited after 23.05 seconds with return value 0
Press any key to continue . . .
```

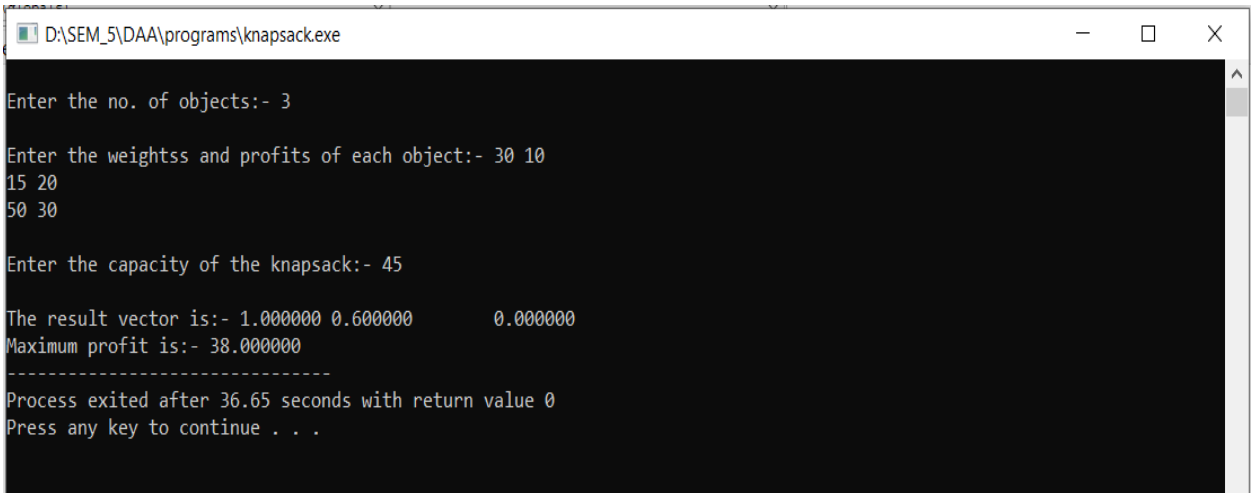
D:\SEM_5\DAA\programs\binarysearch.exe

```
Enter number of elements to be entered:
10
Enter 10 numbers in sorted order:
12
33
56
76
78
79
85
82
101
111
Enter value to search:
86
Not found! 86 isn't present in the list.

--------------------------------
Process exited after 19.52 seconds with return value 0
Press any key to continue . . .
```

Fractional Knapsack

```
# include<stdio.h>

void knapsack(int n, float weight[],
float profit[], float capacity)

{

float x[20], tp = 0;

int i, j, u;

u = capacity;

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

x[i] = 0.0;

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

{

if (weight[i] > u)

break;

else {

x[i] = 1.0;

tp = tp + profit[i];

u = u - weight[i];

}

}

if (i < n)

x[i] = u / weight[i];

tp = tp + (x[i] * profit[i]);

printf("\nThe result vector is:- ");

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

printf("%f\t", x[i]);

printf("\nMaximum profit is:- %f",
tp);

}

int main()

{

float weight[20], profit[20], capacity;

int num, i, j;

float ratio[20], temp;

printf("\nEnter the no. of objects:- ");

scanf("%d", &num);

printf("\nEnter the weightss and
profits of each object:- ");

for (i = 0; i < num; i++) {

scanf("%f %f", &weight[i],
&profit[i]);

}

printf("\nEnter the capacity of the
knapsack:- ");

scanf("%f", &capacity);

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

{

ratio[i] = profit[i] / weight[i];

}

for (i = 0; i < num; i++)
```

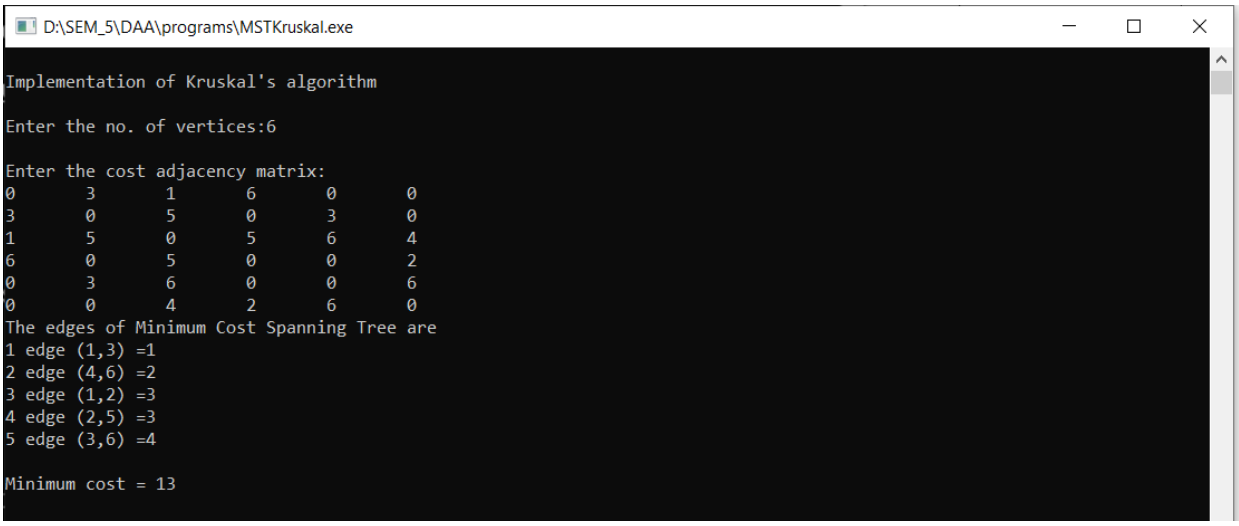
```
{
for (j = i + 1; j < num; j++)
{
if (ratio[i] < ratio[j])
{
temp = ratio[j];
ratio[j] = ratio[i];
ratio[i] = temp;
temp = weight[j];
weight[j] = weight[i];
weight[i] = temp;
temp = profit[j];
profit[j] = profit[i];
profit[i] = temp;
}}}
knapsack(num, weight, profit, capacity);
return(0);
}
```



Kruskal’s Algorithm

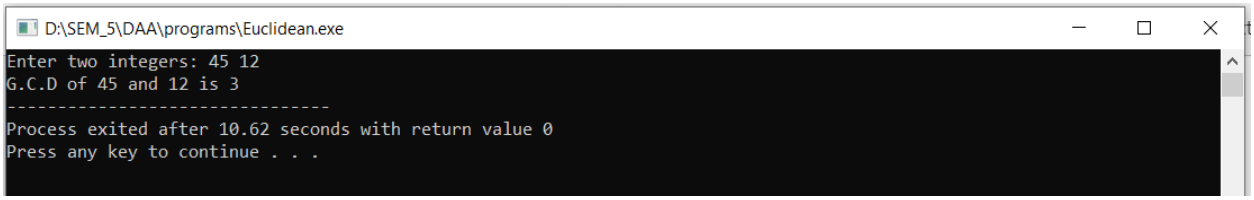
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int
min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
int main()
{

printf("\nImplementation of Kruskal's
algorithm\n");
printf("\nEnter the no. of vertices:");
scanf("%d",&n);
printf("\nEnter the cost adjacency
matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
if(cost[i][j]==0)
cost[i][j]=999;
}
}
printf("The edges of Minimum Cost
Spanning Tree are\n");
while(ne < n)
{
for(i=1,min=999;i<=n;i++)
{
for(j=1;j <= n;j++)
{
if(cost[i][j] < min)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
}
}
u=find(u);
v=find(v);
if(uni(u,v))
{
printf("%d edge (%d,%d)
=%d\n",ne++,a,b,min);
```

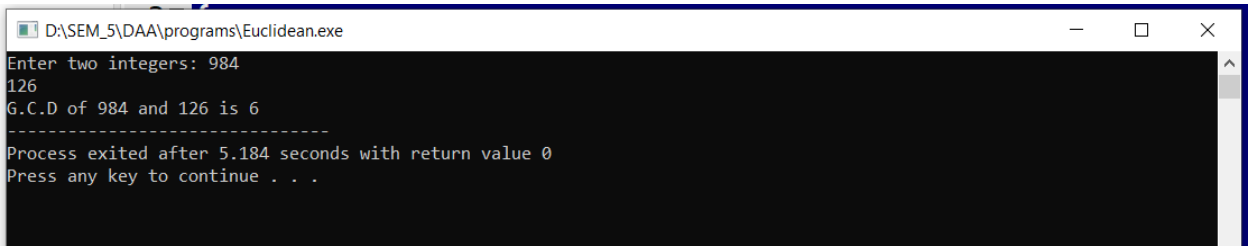
```
mincost +=min;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimum cost =
%d\n",mincost);
getch();
}
int find(int i)
{
while(parent[i])
i=parent[i];
return i;
}
int uni(int i,int j)
{
if(i!=j)
{
parent[j]=i;
return 1;
}
return 0;
}
```



Euclidean Algorithm

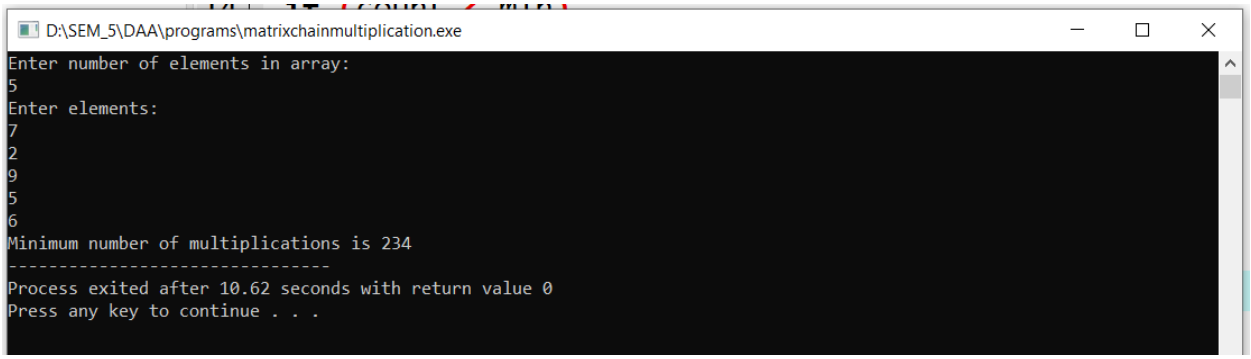
```
#include <stdio.h>
int main()
{
int n1, n2, i, gcd;
printf("Enter two integers: ");
scanf("%d %d", &n1, &n2);
for(i=1; i <= n1 && i <= n2; ++i)
{
if(n1%i==0 && n2%i==0)
gcd = i;
}
printf("G.C.D of %d and %d is %d", n1, n2, gcd);
return 0;
}
```





Matrix Chain Multiplication

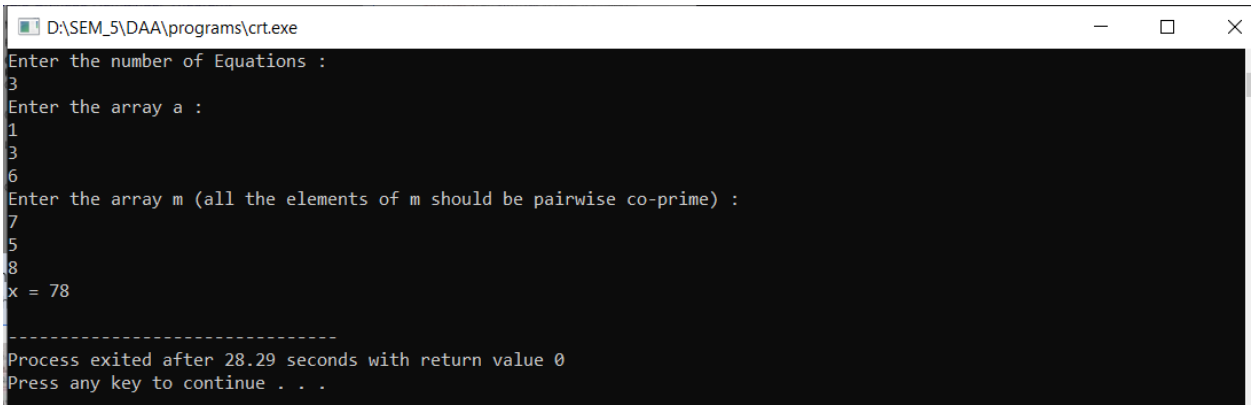
```
#include <limits.h>
#include <stdio.h>
int MatrixChainOrder(int p[], int i, int j)
{
if (i == j)
return 0;
int k;
int min = INT_MAX;
int count;
for (k = i; k < j; k++)
{ count = MatrixChainOrder(p, i, k)
+ MatrixChainOrder(p, k + 1, j)
+ p[i - 1] * p[k] * p[j];
if (count < min)
min = count;
}
return min; }
int main()
{
int i,n,arr[100];
printf("Enter number of elements in array: \n");
scanf("%d",&n);
printf("Enter elements:\n");
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
printf("Minimum number of multiplications is %d ",
MatrixChainOrder(arr, 1, n - 1));
getchar();
return 0;
}
```



Chinese Remainder Theorem

```
#include<stdio.h>
#include<stdlib.h>
long long int MMI_BF(long long int e,long long int mod)
{
long long int i;
for(i=1;i<mod;i++)
if((e*i)%mod==1)
return i;
}
int main()
{
long long int i,n,*a,*b,*m,M,*Marray,answer;
printf("Enter the number of Equations : \n");
scanf("%lld",&n);
a=(long long int*)malloc(sizeof(long long int)*n);
m=(long long int*)malloc(sizeof(long long int)*n);
Marray=(long long int*)malloc(sizeof(long long int)*n);
printf("Enter the array a :\n");
for(i=0;i<n;i++)
scanf("%lld",&a[i]);
printf("Enter the array m (all the elements of m should be pairwise co-prime) :\n");
for(i=0;i<n;i++)
scanf("%lld",&m[i]);
M=1;
```

```
for(i=0;i<n;i++)
M*=m[i];
for(i=0;i<n;i++)
Marray[i]=M/m[i];
answer=0;
for(i=0;i<n;i++)
answer = (answer + ((a[i] * Marray[i])%M *
MMI_BF(Marray[i],m[i]))%M)%M;
printf("x = %lld\n",answer);
return 0;
}
```



N-Queen Problem

```
#define N 4
#include <stdbool.h>
#include <stdio.h>
void printSolution(int board[N][N])
{
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
printf(" %d ", board[i][j]);
printf("\n");
}
}
bool isSafe(int board[N][N], int row,
int col)
{
int i, j;
for (i = 0; i < col; i++)
if (board[row][i])
return false;
for (i = row, j = col; i >= 0 && j >= 0;
i--, j--)
if (board[i][j])
return false;
for (i = row, j = col; j >= 0 && i < N;
i++, j--)
if (board[i][j])
return false;
return true; }
bool solveNQUtil(int board[N][N], int
col)
{
if (col >= N)
return true;
for (int i = 0; i < N; i++)
{
if (isSafe(board, i, col))
{
board[i][col] = 1;
if (solveNQUtil(board, col + 1))
return true;
board[i][col] = 0;
}
}
return false;
}
bool solveNQ()
{
int board[N][N] = { { 0, 0, 0, 0 },{ 0,
0, 0, 0 }, { 0, 0, 0, 0 },{ 0, 0, 0, 0 } };
```

```
if (solveNQUtil(board, 0) == false)
{
printf("Solution does not exist");
return false;
}
printSolution(board);
return true;
}
int main()
{
solveNQ();
return 0;
}
```