Course Code	BCSL404	CIE Marks	50
Number of Contact Hours/Week	0:0:2	SEE Marks	50
Total Number of Lab Contact Hours	28	Exam Hours	03

PROGRAM 01:

Design and implement C/C++ Program to find Minimum Cost Spanning Key of a given connected undirected graph using Kruskal's Algorithm.

```
#include<stdio.h>
#define INFI 99
#define MAX 10
int a[MAX][MAX],b[MAX][MAX],n,cost=0;
void findmin(int * v1,int *v2)//finding the edge having minimum weight.
{
int edge=INFI,i,j;
for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
if(a[i][j]>0 && a[i][j]<edge)
{
edge=a[i][j];
*v1=i;
*v2=j;
}

}
void update(int root[],int v1,int v2)
{
int temp,i;
temp=root[v2];
for(i=1;i<=n;i++)
{
if(root[i]==temp)
root[i]=root[v1];
```

```
}
}
void kruskal()
{
int i ,v1,v2,root[MAX],edge,count=0;
for(i=1;i<=n;i++)
root[i]=i;
i=0;
while(i!=n-1)
{
findmin(&v1,&v2);
edge=a[v1][v2];
a[v1][v2]=a[v2][v1]=0;//do not select the same edge on next time.
if(root[v1]!=root[v2])
{
printf("(%d,%d)\n",v1,v2);
update(root,v1,v2);
cost+=edge;
i++;
}
}
}
int main()
{
int i,j;
printf("\n Enter the number of vertices : ");

scanf("%d",&n);
printf("\n Enter the weighted graph : \n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\n Edges of spanning tree are:\n");
kruskal();
printf("\n Minimum cost=%d:",cost);
```

```
return(0);
}
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices : 4

 Enter the weighted graph :
0 2 99 2
2 0 3 1
99 3 0 5
2 1 5 0

 Edges of spanning tree are:
(2,4)
(1,2)
(2,3)

 Minimum cost=6:aiml-admin@aimladmin-HP:~/ada$
```

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices : 4

 Enter the weighted graph :
0 1 99 2
1 0 6 2
99 6 0 5
2 2 5 0

 Edges of spanning tree are:
(1,2)
(1,4)
(3,4)

 Minimum cost=8:aiml-admin@aimladmin-HP:~/ada$
```

2. Design and implement C/C++ Program to find Minimum Cost Spanning Key of a given connected undirected graph using Prim's Algorithm.

```
#include<stdio.h>
#define INFI 99
int edges[10][3],n,wt[10][10];;
void prims();
void main()
{
int i,j;
printf("\n Enter the number of vertices: ");
scanf("%d",&n);
printf("\n Enter the cost matrix? \n");
for( i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&wt[i][j]);
prims();
}
void prims()
{
int u[10],lowcost[10],visited[10];
int min,mincost=0,i,j,v;
//mark nodes as unvisited
visited[1]=1;
//find low cost edge
for(i=2;i<=n;i++)
{
visited[i]=0;
u[i]=1;
lowcost[i]=wt[1][i];
}
for(i=1;i<=n-1;i++)
{ min=lowcost[2];
v=2;
for(j=3;j<=n;j++)
```

```
{
if(lowcost[j]<min)
{
min=lowcost[j];
v=j;
}
}
//save edge
edges[i][1]=u[v];
edges[i][2]=v;
mincost+=lowcost[v];
visited[v]=1;
lowcost[v]=INFI;
for(j=2;j<=n;j++)
if(wt[v][j]<lowcost[j] && !visited[j])
{
lowcost[j]=wt[v][j];
u[j]=v;
}
}
printf("\n\n The edges of this minimum cost spanning tree are\n");
for(i=1;i<=n-1;i++)
printf(" (%d,%d)\n",edges[i][1],edges[i][2]);
printf("\n Minimum cost Spanning Tree is:%d",mincost);
}
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices: 4

 Enter the cost matrix?
0 2 99 2
2 0 5 3
99 5 0 4
2 3 4 0


 The edges of this minimum cost spanning tree are
 (1,2)
 (1,4)
 (4,3)

 Minimum cost Spanning Tree is:8aiml-admin@aimladmin-HP:~/ada$
```

3. a) Design and implement C/C++ Program to solve All-Pairs Shortest Paths Problem using Floyd's algorithm.

b) Design and implement C/C++ Program to find the transitive closure using Warshall's algorithm

```
#include<stdio.h>
#include <stdlib.h>
#define MAX 10
#define min(c,d) (c<d?c:d)
int dist[MAX][MAX],n;
void floyd()
{
 int i,j,k;

for(k=1;k<=n;k++)  // record the lengths of shortest path
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);
}

void main()
{
   int i, j;
printf("Enter the number of vertices :\n");
 scanf("%d",&n);
 printf("Enter the distance matrix\n");//read distance matrix
 for(i=1;i<=n;i++)
 for( j=1;j<=n;j++)
 scanf("%d",&dist[i][j]);

 floyd();
 printf("\nAll pairs shortest path matrix is :\n");
 for( i=1;i<=n;i++)
 {
 for( j=1;j<=n;j++)
```

```
 {
    printf("%d\t",dist[i][j]);
}
printf("\n");
 }
 }
```

b.

Warshall's Algorithm

```
#include<stdio.h>
#define MAX 10
int D[MAX][MAX],n;
void warshall()
{
int i,j,k;
for(k=1;k<=n;k++)
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
D[i][j] = D[i][j] || ( D[i][k] && D[k][j] );
}
void main()
{
int i, j;
printf("Enter the number of vertices :\n");
scanf("%d",&n);
printf("Enter the adjacency matrix\n");
for(i=1;i<=n;i++)
for( j=1;j<=n;j++)
scanf("%d",&D[i][j]);
warshall();
printf("Trasitive closure of digraph is :\n");
for( i=1;i<=n;i++)
{
for( j=1;j<=n;j++)
printf("%d\t",D[i][j]);
printf("\n") ;
```

```
}
}
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ gcc -fopenmp program3a.c
aiml-admin@aimladmin-HP:~/ada$ ./a.out
Enter the number of nodes: 4
Enter the cost adjacency matrix:
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0

Total Threads Used are: 12
All-Pairs Shortest Paths is as follows:
0       10      3       4
2       0       5       6
7       7       0       1
6       16      9       0

The time taken to perform Floyd's Algorithm is: 0.035606
aiml-admin@aimladmin-HP:~/ada$
```

```
aiml-admin@aimladmin-HP:~$ gcc w.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the number of vertices :
4
Enter the adjacency matrix
0 1 1 0
0 0 0 0
1 0 0 1
1 0 0 0
Trasitive closure of digraph is :
1       1       1       1
0       0       0       0
1       1       1       1
1       1       1       1
aiml-admin@aimladmin-HP:~$
```

4. Design and implement C/C++ Program to find From a given vertex in a weighted connected graph to other vertices using Dijkstra's algorithm.

```
#include<stdio.h>
#define INFINITY 99
void dijkstra(int);
void printpath(int);
int MinVertex();
int dist[10],p[10],visit[10]; //p->penultimate vertex(previous vertex)
int wt[10][10], n, edge;
int main()
{
int i,j,s;
printf("\n Enter the number of vertices in a graph : ");
scanf("%d",&n);
for( i=1;i<=n;i++)
{
dist[i]=0; p[i]=0; visit[i]=0;
}
printf("\n Enter the weight matrix? \n");
for( i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&wt[i][j]);
printf("\n enter the source vertex: ");
scanf("%d",&s);
printf("\n\n Shortest paths from vertex %d are\n\n",s);
dijkstra(s);
printpath(s);
return 0;
}
void dijkstra(int s)
{
int i,j,step,u;
for(i=1;i<=n;i++)

{
```

```
dist[i] = wt[s][i];
if(dist[i] ==INFINITY)
p[i]=0;
else
p[i]=s;

}
visit[s]=1;
dist[s]=0;
for(step=2;step<=n;step++)
{
u=MinVertex(); //choose the minimum distance vetrex from source
visit[u]=1;
for(j=1;j<=n;j++) //update distances of fringe vertices from source
if( ((dist[u]+wt[u][j]) < dist[j]) && !visit[j])
{
dist[j]=dist[u]+wt[u][j];
p[j]=u;
}

}
}
int MinVertex()
{
int min=INFINITY;
int u,i;
for(i=1;i<=n;i++)
{
if((dist[i]<min) && (visit[i]==0))
{
min=dist[i];
u=i;
}
}
return u;
```

```
}
void printpath(int s)
{
int i,t;
for(i=1;i<=n;i++)
if(visit[i]==1 && i!=s)
{
printf("vertext->%d length:%d path: ",i,dist[i]);
t=p[i];
printf("%d",i);
while(t!=s)
{
printf("<--%d",t);
t=p[t];
}
if(t==s)
printf("<--%d\n",s);

}

}
```

output:

```
aiml-admin@aimladmin-HP:~/ada$ gcc p5.c
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices in a graph : 4

 Enter the weight matrix?
0 2 5 1
2 0 1 2
5 1 0 3
1 2 3 0

 enter the source vertex: 2


 Shortest paths from vertex 2 are

vertext->1 length:2    path: 1<--2
vertext->3 length:1    path: 3<--2
vertext->4 length:2    path: 4<--2
aiml-admin@aimladmin-HP:~/ada$
```

5. Design and implement C/C++ Program to obtain the Topological ordering of vertices in a given digraph.

```
#include<stdio.h>
#define MAX 10
void top(int ad[MAX][MAX],int n)
{
int f,count=0,flag=1,i,j,k;
int torder[100],in=1;
while(flag)
{
count++;
for(i=1;i<=n;i++)
{
f=0;
for(j=1;j<=n;j++)
{
if(ad[j][i]!=0 || torder[j]==i)
{
f=1;
break;
}
}
if(f!=1)
{
torder[in++]=i;
for(k=1;k<=n;k++)
ad[i][k]=0;
}
}
if(count==n || in>n)
flag=0;
}
if(in<=n)
printf("\n No topological order");
else
```

```
{
printf("\n Topological sequence is... \n");
for(i=1;i<=n;i++)
printf("%d\t",torder[i]);
}
}
void main()
{
int ad[MAX][MAX],n,i,j;
printf("\n Enter the number of vertices: ");
scanf("%d",&n);
printf("\n Enter the matrix of the digraph :\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&ad[i][j]);
top(ad,n);
}
```

output:

```
aiml-admin@aimladmin-HP:~$ ./a.out

 Enter the number of vertices: 3

 Enter the matrix of the digraph :
0 1 0
0 0 1
1 0 0

 No topological orderaiml-admin@aimladmin-HP:~$ ./a.out

 Enter the number of vertices: 4

 Enter the matrix of the digraph :
0 0 0 1
1 0 0 1
0 1 0 1
0 0 0 0

 Topological sequence is...
3       2       1       4       aiml-admin@aimladmin-HP:~$
```

6. Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic Programming method.

```
#include <stdio.h>
int my_max(int a, int b)
{
return (a > b) ? a : b;
}
int val[20],wt[20],n,c,v[20][20];
//build a matrix(v[i][j]) with the weight bounds as the columns and the number of items as the rows.
int knap()
{ int i,j;
for(i=0;i<=n;i++)
for(j=0;j<=c;j++)
if(i==0||j==0)
v[i][j] = 0;
else
if(wt[i]>j)
v[i][j] = v[i-1][j];
else
v[i][j] =my_max(v[i-1][j],(v[i-1][j-wt[i]]+val[i]));
return v[n][c];
}
int main()
{ int opt,i,j;
printf("\nEnter the no of items in Knapsack : ");
scanf("%d",&n);
printf("\nEnter values (profit) of %d elements : ",n);
for(i=1;i<=n;i++)
scanf("%d",&val[i]);
printf("\nEnter weight of %d elements : ",n);
for(i=1;i<=n;i++)
scanf("%d",&wt[i]);
printf("\n Enter the capacity of Knapsack : ");
scanf("%d",&c);
```

```
opt = knap();
printf("\n\nCapacity");
for(j=0;j<=c;j++)
printf("%4d",j);
printf("\n");
for(i=0;i<=n;i++)
{
printf("\nItem-%2d:",i);
for(j=0;j<=c;j++)
printf("%4d",v[i][j]);
}
printf("\n\nOptimal solution is : %d",opt);
printf("\n\n The selected items are : ");
while(n>0)//best subset with weight at most knapsack size
{
if(v[n][c] != v[n-1][c])
{
printf("%d\t",n);
c = c - wt[n];
}n --;
}
return(0);
}
```

output:

```
aiml-admin@aimladmin-HP:~$ ./a.out

Enter the no of items in Knapsack : 4

Enter values (profit) of 4 elements : 12 10 20 15

Enter weight of  4 elements : 2 1 3 2

 Enter the capacity of Knapsack : 5


Capacity   0   1   2   3   4   5

Item- 0:   0   0   0   0   0   0
Item- 1:   0   0  12  12  12  12
Item- 2:   0  10  12  22  22  22
Item- 3:   0  10  12  22  30  32
Item- 4:   0  10  15  25  30  37

Optimal solution is : 37

 The selected items are : 4     2       1       aiml-admin@aimladmin-HP:~$
```

7. . Design and implement C/C++ Program to solve discrete Knapsack and continuous Knapsack problem using greedy approximation method

```
#include <stdio.h>
int main() {
int i, j = 0, max_qty, m, n;
float sum = 0, max;
int array[2][20];
printf("Enter no of items: ");
scanf("%d", &n);
printf("Enter the weights of each items: ");
for (i = 0; i < n; i++) {
scanf("%d", &array[0][i]);
}
printf("Enter the values of each items: ");
for (i = 0; i < n; i++) {
scanf("%d", &array[1][i]);
}
printf("Enter maximum volume of knapsack: ");
scanf("%d", &max_qty);
m = max_qty;
while (m > 0) {
max = 0;
for (i = 0; i < n; i++) {
if (((float)array[1][i]) / ((float)array[0][i]) > max) {
max = ((float)array[1][i]) / ((float)array[0][i]);
j = i;
}
}
if (array[0][j] > m) {
printf("Quantity of item number: %d added is %d\n", (j + 1), m);
sum += m * max;
m = -1;
} else {
printf("Quantity of item number: %d added is %d\n", (j + 1), array[0][j]);
```

```
m -= array[0][j];
sum += (float)array[1][j];
array[1][j] = 0;
}
}
printf("The total profit is %.2f\n", sum);
return 0;
}
```

output:

8. Design and implement C/C++ Program to find a subset of a given set S = {sl, s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d.

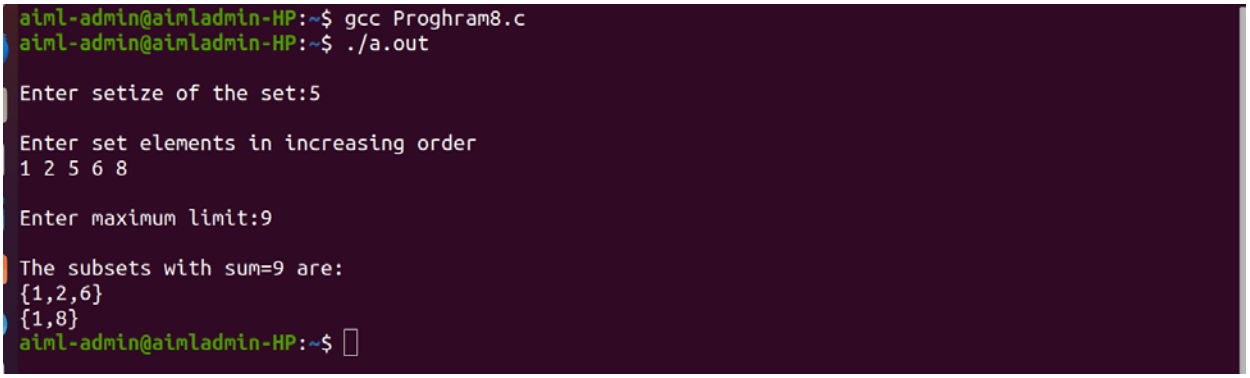
```
#include<stdio.h>
int set[10], x[10], d, n;
void sumofsub(int,int);
int main()
{
int sum=0,i;
printf("\nEnter setize of the set:");
scanf("%d",&n);
printf("\nEnter set elements in increasing order\n");
for(i=1;i<=n;i++)
scanf("%d",&set[i]);
printf("\nEnter maximum limit:");
scanf("%d",&d);
printf("\nThe subsets with sum=%d are:\n",d);
for( i=1;i<=n;i++)
sum=sum+set[i];
if( sum<d || set[1]>d )
printf("\n No setubsetet posetsetible");
else
sumofsub(0,1);

return 0;
}
void sumofsub(int s, int k)
{
int i;
x[k]=1; //generate left child
if(s+set[k]==d)
{
printf("{");
for(i=1;i<=n;i++)
if(x[i]==1) //subset found
```

```
printf("%d,",set[i]);
printf("\b}\n");
}
else

{
if(s+set[k]<d && k+1<=n)//with s[k]
{
sumofsub(s+set[k],k+1); //Generate right child
x[k+1]=0;
}
if(s+set[k+1]<=d && k+1<=n)//without s[k]
{
x[k]=0;
sumofsub(s,k+1);
x[k+1]=0;
}
}
}
```

output:



9. Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time Complexity. Run the program for varied values of n>5000 and record the time to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int n;
int *array;
void input()
{
printf("Enter the total numbers: ");
scanf("%d", &n);
array = (int *)malloc(n * sizeof(int));
for (int i = 0; i < n; i++) {
array[i] = rand() % 1000; // Generates random numbers 0-999
}
printf("Unsorted array\n");
for (int i = 0; i < n; i++) {
printf("%d ", array[i]);

}
printf("\n");
}
void selectionSort()
{
for (int step = 0; step < n - 1; step++) {
int min_idx = step;
for (int i = step + 1; i < n; i++) {
if (array[i] < array[min_idx]) {
min_idx = i;
}
}
int temp = array[step];
```

```
array[step] = array[min_idx];
array[min_idx] = temp;
}
}
int main()
{
input();
clock_t start = clock();
selectionSort();
clock_t end = clock();
double duration = ((double)(end - start)) / CLOCKS_PER_SEC * 1000000000;
printf("\nTime for sorting is %.2f nano seconds\n", duration);
printf("Sorted Array in Ascending Order:\n");
for (int i = 0; i < n; i++) {
printf("%d ", array[i]);
}
printf("\n");
free(array);
return 0;
}
```

output:

```
aiml-admin@aimladmin-HP:~$ gcc sel.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the total numbers: 10
Unsorted array
383 886 777 915 793 335 386 492 649 421

Time for sorting is 4000.00 nano seconds
Sorted Array in Ascending Order:
335 383 386 421 492 649 777 793 886 915
aiml-admin@aimladmin-HP:~$
```

10. Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time Complexity. Run the program for varied values of n>5000 and record the time to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include <stdio.h>
#include<stdlib.h>
#include <time.h>
#include<sys/time.h>
void swap(int *a,int *b) {
int temp;
temp=*a;
*a=*b;
*b=temp;
}
int partition(int a[],int left,int right) {
int p,i,j;
p=a[left];
i=left+1;
j=right;
while(1) {
while(a[i]<p && i<=right) {
i++;//find elt>p
//delay(100); }
while(a[j]>p)//find elt<p {
j--;
//delay(100);
}
if(i>j)
break;
swap(&a[i],&a[j]);
i++;
j--;

}
swap(&a[left],&a[j]);//swap pivot and a[j]
```

```
return j;
}
void quicksort(int a[],int left,int right)
{
int s;
if(left<right)
{ s=partition(a,left,right);
quicksort(a,left,s-1);
quicksort(a,s+1,right);
}
}
void input(int a[],int n)
{
int i;
for(i=0;i<n;i++)
{
a[i]=rand()%100;
}
}

void main()
{
int n,a[100],i;
struct timeval tv1, tv2;
printf("Enter the number of elements:\n");
scanf("%d",&n);
printf("randomly generated elements are:\n");
input(a,n);
printf("Array is\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
clock_t start=clock();
quicksort(a,0,n-1);
clock_t end=clock();
```

```
printf("\nSorted array is\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
double  elapsedTime= (double)(end - start) / CLOCKS_PER_SEC * 1000000000;
printf ("Total time = %f nanoseconds\n",elapsedTime);

}
```

output:

```
aiml-admin@aimladmin-HP:~$ gcc quick.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the number of elements:
10
randomly generated elements are:
Array is
83      86      77      15      93      35      86      92      49      21
Sorted array is
15      21      35      49      77      83      86      86      92      93      Total time = 3000.00
0000 nanoseconds
aiml-admin@aimladmin-HP:~$
```

11. Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time Complexity. Run the program for varied values of n>5000 and record the time to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

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

#define MAX 10000

void merge(int array[], int low, int mid, int high) {
int i = low;
int j = mid + 1;
int k = low;
int resarray[MAX];

while (i <= mid && j <= high) {
if (array[i] < array[j]) {
resarray[k++] = array[i++];
} else {
resarray[k++] = array[j++];
}
}

while (i <= mid) {
resarray[k++] = array[i++];
}

while (j <= high) {
resarray[k++] = array[j++];
}

for (int m = low; m <= high; m++) {
array[m] = resarray[m];
}
}

void sort(int array[], int low, int high) {
if (low < high) {
int mid = (low + high) / 2;
sort(array, low, mid);
sort(array, mid + 1, high);
merge(array, low, mid, high);
}
}

int main() {
int array[MAX];
int i;
```

```
printf("Enter the array size: ");
int n;
scanf("%d", &n);

srand(time(NULL));
for (i = 0; i < n; i++) {
array[i] = rand() % 20;
}

printf("Array before sorting:\n");
for (i = 0; i < n; i++) {
printf("%d ", array[i]);
}
printf("\n");

clock_t start = clock();
sort(array, 0, n - 1);
clock_t end = clock();

double elapsedTime = (double)(end - start) / CLOCKS_PER_SEC * 1000000000;
printf("Time taken to sort array is: %.0f nanoseconds\n", elapsedTime);

printf("Sorted array:\n");
for (i = 0; i < n; i++) {
printf("%d ", array[i]);
}
printf("\n");

return 0;
}
```

output:

```
aiml-admin@aimladmin-HP:~$ gcc mergesortnew.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the array size: 10
Array before sorting:
0 10 14 3 1 10 5 5 9 5
Time taken to sort array is: 42000 nanoseconds
Sorted array:
0 1 3 5 5 5 9 10 10 14
aiml-admin@aimladmin-HP:~$
```

12. Design and implement C/C++ Program for N Queen's problem using Back Tracking.

```
#include<stdio.h>
#include<stdlib.h>
void Nqueen(int);
int place(int);
int col[30],count=0;
void main()
{
int i,n;
printf("\n Enter the number of queens: ");
scanf("%d",&n);
Nqueen(n);
printf("\n Total no.of solutions are %d", count);
}
int place(int r)
{
int i;
for(i=1;i<r;i++)//i-->row r-->current row
{
if((col[i]==col[r]) ||(abs(i-r)==abs(col[i]-col[r])))
//if q is already placed in column col[k] OR (k,col[k]) cell and
//one of the previously placed queen are in same diagonal
return 0;
}
return 1;
}
void Nqueen(int n)
{
int r=1,i,j;//r-->row
col[r]=0;
while(r!=0)
{
col[r]=col[r]+1;
while((col[r]<=n) && !place(r))
col[r]=col[r]+1;
```

```
if(col[r]<=n) //if 1
{
if(r==n)
{
count++;
printf("\nSolution # %d\n",count);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(j==col[i])
printf("Q ");
else
printf("* ");
}
printf("\n\n");
}
}
else
{
r++;
col[r]=0;

}
}//if 1
else
r --; //backtracking
}//while
}
```

output:

```
aiml-admin@aimladmin-HP:~$ gcc Program12.c
aiml-admin@aimladmin-HP:~$ ./a.out

 Enter the number of queens: 4

Solution # 1
* Q * *

* * * Q

Q * * *

* * Q *


Solution # 2
* * Q *

Q * * *

* * * Q

* Q * *
```