SYLLABUS

2019-2022 BATCH

I B.Sc(CS)

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.

DEPARTMENT OF CS, CA & IT
LIST OF PRACTICALS

1. Write a C Program to find the number of subsets of a set contains n elements.
2. Write a C Program to find transitive closure of a relation.
3. Write a C Program to prove

1/ (1*2) + 1/(2*3) ……+1/(n(n+1)) = n/(n+1)

4. Write a C Program to to perform the sum = 1+ (1+2) + (1+2+3) + … + (1+2…+n)
5. Write a C program to print Fibonacci series till Nth term using recursion
6. Write a Cprogram in c to calculate factorial of a number using recursion
7. Write a C Program to find a minimum spanning tree using Prim's algorithm
8. Write a C program to find the shortest path with the lower cost in a graph using Dijkstra's Algorithm
9. Write a C Program to construct the truth table for the following formula.

(i) $P \wedge Q \wedge \neg R$ (ii) $P \wedge \neg Q \wedge R$ (iii) $P \wedge Q \wedge \neg R$

10. Write a C Program to prove De – Morgan's law.

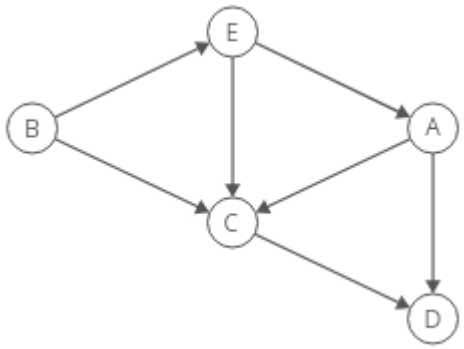
Prg 1://Subset from a given set

```
#include <stdio.h>

/*  Function to generate subset  */
void subset(int arr[], int data[], int start, int end, int index, int r)
{
 int j, i;
 if (index == r) {
      for (j = 0; j < r; j++)
           printf("%d ", data[j]);
       printf("\n");
      return;
  }
      for (i = start; i <= end && end - i + 1 >= r - index; i++)
     {
         data[index] = arr[i];
         subset(arr, data, i+1, end, index+1, r);      }
 }
  /*  End of subset()  */
  /*  Function to print the subset  */
  void printsubset(int arr[], int n, int r)
 {
      int data[2];
      subset(arr, data, 0, n - 1, 0, r);
 }
 /*  End of printsubset()  */

  int main()
  {
     int arr[20], k, n, i;
      printf("Enter the number of input : ");
      scanf("%d", &n);
      printf("\nEnter the integers: \n");
      for (  i = 0; i < n; i++)
      {
              scanf("%d", &arr[i]);
      }
      printf("Enter value of k: ");
     scanf("%d", &k);
     printsubset(arr, n, k);
    return 0;
}
```

Prg 2://Transitive closure



	A	B	C	D	E
A	1	0	1	1	0
B	1	1	1	1	1
C	0	0	1	1	0
D	0	0	0	1	0
E	1	0	1	1	1

```
#include<stdio.h>
#include<conio.h>
#define V 4
void printSolution(int reach[][V]);
 void transitiveClosure(int graph[][V])
{
   int reach[V][V], i, j, k;
       for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
         reach[i][j] = graph[i][j];
    for (k = 0; k < V; k++)
   {
      for (i = 0; i < V; i++)
      {
         for (j = 0; j < V; j++)
         {
//If vertex k is on a path from i to j,
                 // then make sure that the value of reach[i][j] is 1
            reach[i][j] = reach[i][j] || (reach[i][k] && reach[k][j]);
         }
      }
   }
    printSolution(reach);
}
 void printSolution(int reach[][V])
{
```

```
int i,j;
    printf ("Following matrix is transitive closure of the given graph\n");
    for ( i = 0; i < V; i++)
    {
        for (j = 0; j < V; j++)
            printf ("%d ", reach[i][j]);
        printf("\n");
    }
}
int main()
{
        int graph[V][V] = { {1, 1, 0, 1},
                            {0, 1, 1, 0},
                            {0, 0, 1, 1},
                            {0, 0, 0, 1}
                          };
     clrscr();
    // Print the solution
    transitiveClosure(graph);
    getch();
    return 0;
}
```

Prg 3://To Prove the Series is Equal or not

```
#include<stdio.h>
#include<conio.h>
void main()
{
float i,j,sum=0.0,n;
clrscr();
printf("enter the value for n:");
scanf("%f",&n);
for(i=1;i<=n;i++)
{
sum=sum+(1/(i*(i+1)));
printf(“\n Sum=%f”,sum);
}
printf(“\n”);
j=(n/(n+1));
printf(“\n n/(n+1) = %f ”,j);
if(sum==j) {
printf("\n the series is equal");
}
else {
printf("\n the series is not equal");
}
getch();
}
```

Prg 4://Sum of Series

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,j,sum=0;
clrscr();
printf("enter the value for n:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=i;j++)
{
sum=sum+j;
printf(“\t %d”,j);
}
printf(“\n”);
}
printf("\n the series is =%d", sum);
getch();
}
```

Prg 5://Fibonacci Series using Recursion

```
#include <stdio.h>
#include <conio.h>
 int fibonacci(int term);
void main(){
    int terms, counter;
    printf("Enter number of terms in Fibonacci
series: ");
    scanf("%d", &terms);
    printf("Fibonacci series till %d terms\n", terms);
    for(counter = 0; counter < terms; counter++){
        printf("%d\n", fibonacci(counter));
    }
    getch();
}

int fibonacci(int term){
        if(term < 2)
       return term;
```

```
    return fibonacci(term - 1) + fibonacci(term - 2);
}
```

Prg 6://Factorial using Recursion

```
#include <stdio.h>
int factorial(int n);
void main()
{
    int n;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    printf("Factorial of %d = %d", n, factorial(n));
}

int factorial(int n)
{
    if (n >= 1)
        return n* factorial(n-1);
    else
        return 1;
}
```

Prg 7://Minimum cost spanning Tree

```
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10];
void main()
{
clrscr();
printf("\n Enter the number of nodes:");
scanf("%d",&n);
printf("\n Enter the 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;
}
visited[1]=1;
printf("\n");
while(ne<n)
{
for(i=1,min=999;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]<min)
if(visited[i]!=0)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
if(visited[u]==0 || visited[v]==0)
{
printf("\n Edge %d:(%d %d) cost: %d",ne++,a,b,min);
mincost+=min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\n Minimun cost=%d",mincost);
getch();
}
```

Prg 8://Shortest path using Dijikstra’s Algorithm

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
int path[2][2],i,j,min,a[3][3],p,st=1,ed=3,stp,edp,t[3],index;
clrscr();
printf("enter the cost matrix\n");
for(i=1;i<=3;i++)
for(j=1;j<=3;j++)
scanf("%d",&a[i][j]);
printf("enter  number of paths\n");
scanf("%d",&p);
printf("enter possible paths\n");
for(i=1;i<=p;i++)
for(j=1;j<=3;j++)
scanf("%d",&path[i][j]);
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=3;j++)
{
edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else
stp=edp;
}
}
min=t[st];index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];
index=i;
}
}
printf("minimum cost %d",min);
printf("\n minimum cost path ");
for(i=1;i<=3;i++)
{
printf("--> %d",path[index][i]);
if(path[index][i]==ed)
break;
}
getch();
}
```

Prg 9://Constructing a Truth Table

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y,z;
clrscr();
printf("\n X\tY\tZ\tXY+Z");

for(x=0;x<=1;++x)
for(y=0;y<=1;++y)
for(z=0;z<=1;++z)
{
if(x*y+z==2)
printf("\n%d\t%d\t%d\t",x,y,z);
else
printf("\n%d\t%d\t%d\t%d",x,y,z,x*y+z);
}
getch();
}
```

Prg 10://To Prove De- Morgan's Law

```
#include <stdio.h>
#include <math.h>
```

```
int main()
{
int x;
int y;
printf("Enter x: ");
scanf("%d",&x);
printf("Enter y: ");
scanf("%d",&y);
if ((!(x < 3) && !(y >= 7)) && !((x < 3) || (y >= 7)))
{
printf("True!\n\n");
}
else
{
printf("False!\n\n");
}
}
```