FIRST SEMESTER MCA (2020 SCHEME)

practical examination June 2021

20MCA135 DATA STRUCTURES LAB

Date : 30.06.2021 Time : 1.00 PM - 3 PM

Submitted by,

Thasni PK

ICE20MCA-2040

Batch-3

1 Qn:- Write a program to impliment Linked List ?

Program

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<limits.h>
#define CAPACITY 10000
  struct stack
  {
     int data;
     struct stack *next;
  }
```

```
}
*top;
// Stack Size
int size = 0;
void push (int element);
int pop ();
void main ()
{
 int choice, data;
 clrscr ();
 Printf (".....\n");
 Printf (" LINKED STACK IMPLIMENTATION PROGRAM
                                        \n");

 printf (".....\n");
 printf (" 1. push \n");
 Printf (" 2. pop \n");
 printf ( " 3. Size \n");
 printf ( " 4. exit \n");
 printf (".....\n");
 while (1)
 {
  printf (" Enter your choice:");
  Scanf (" %d, &choice);
  switch ( choice )
  {
```

```
case 1 :
  printf(" Enter data to push into stack : ");
  scanf ("%d ", &data );
  push (data);
  break ;
case 2 :
    data = pop( );
    if ( data != INT_MIN)
    printf (" Data => %d \n", data);
    break ;
case 3:
    printf (" Stack size : %d \n", size);
    break ;
case 4:
    printf("Exiting from app \n");
    exit (0) ;
    break ;
default :
   printf(" Invalid choice please try agais\n");
 }
  printf(" \n \n");
   getch();
 }
 }
```

```
void push (int element)
{
Struct Stack *New Node;
if (size >= capacity)
{
printf("Stack overflow, cant add more elements
                                to stack \n");
return;
}
new Node = (struct stack *) malloc (sizeof (
                        Struct stack));
New Node -> data = element;
new Node -> next = top;
top = new Node;
size++;
printf("Data pushed to stack \n");
}
int pop ()
{
int data = 0;
Struct Stack * top Node;
if (size <= 0 || !top)
```

```
{
printf("Stack is empty \n");
return INT_MIN;
}
topNode = top;
data = top -> data;
top = top -> next;
free(topNode);
size--;
return data;
}
```

Expected output

```
.....
STACK IMPLIMENTATION PROGRAM
.....
1. Push
2. POP
3. Size
4. Exit

Enter your choice 1
Enter data to push into stack : 10
data pushed to stack
Enter your choice : 1
enter data to push into stack : 20
data pushed to stack
```

```
enter your choice : 1
enter data to push into stack : 30
Data pushed to stack
enter your choice : 2
data => 30
```

2 Qn write a program to impliment kruskal algorithm?

Program

```
#include <stdio.h>
# include < conio.h>
# define MAX 30
typedef struct edge
{
    int u, v, w ;
}edge ;
typedef struct edge_list
{
  edge data [MAX];
  int n;
}
 edge_list ;
 edge_list elist ;
int graph [MAX][MAX], n;
```

```
edge_list spanlist;
void kruskal Algo();
int find(int belongs[], int vertex no);
void apply Union(int belongs[], int c1, int c2);
void sort();
void print();
void kruskal Algo()
{
  int belongs[MAX], i, j, cno1, cno2;
  elist.no = 0;
  Printf("Elements of the graph are \n");
  for(i=1; i<n; i++)
  for(j=0; j<i; j++)
  {
    if(Graph[i][j] != 0)
    {
      elist.data[elist.n].u = i;
      elist.data[elist.n].v = j;
      elist.data[elist.n].w = graph[i][j];
      elist.n++;
    }
}
}
Sort();
for(i=0; i<n; i++)
 belongs[i] = i;
```

```
spanlist.n = 0;
for(i=0;i<elist.n;i++)
{
  cno1 = find(belongs, elist.data[i].v);
  if(cno1 != cno2)
  {
   spanlist.data[spanlist.n] = elist.data[i];
   spanlist.n = spanlist.n+1;
   applyUnion(belongs, cno1, cno2);
  }
 }
}
int find(int belongs[] int vertexno)
 {
  return(belongs[vertexno]);
 }
  void applyUnion(int belongs[], int c1, int c2)
  {
   int i;
   for(i=0;i<n;i++)
   if(belongs[i]==c2)
    belongs[i] = c1;
  }
   Void sort()
```

```
{
int i,j;
edge temp;
for (i=1;i<elist.n;i++)
for (j=0;j<elist.n;j++)
if (elist.data[j].w > elist.data[j+1].w)
{
temp = elist.data[j]
elist.data[j]= elist.data[j+1];
elist.data[j+1]= temp;
}}
void print()
{
int i, cost=0;
for (i=0;i<spanlist.n;i++)
{printf("\n%d %d : %d", spanlist.data
[i].u, spanlist.data[i].v, spanlist.data[i].w);
cost = cost + spanlist.data[i].w;
}
printf("\n spanning tree cost : %d",cost);
}
void main()
{
```

```
int i, j, total_cost;
clrscr();
n=6;
graph[0][0]=0;
graph[0][1]=4;
graph[0][2]=4;
graph[0][3]=0;
graph[0][4]=0;
graph[0][5]=0;
graph[0][6]=0;
graph[1][0]=4;
graph[1][1]=0;
graph[1][2]=8;
graph[1][3]=0;
graph[1][4]=0;
graph[1][5]=0;
graph[1][6]=0;
graph[2][0]=4;
graph[2][1]=8;
graph[2][2]=0;
graph[2][3]=3;
graph[2][4]=4;
graph[2][5]=0;
graph[2][6]=0;
```

```
graph[3][0] =0;
graph [3][1] =0;
graph [3][2] =3;
graph [3][3] =0;
graph [3][4] =3;
graph [3][5] =0;
graph [3][6] =0;

graph [4][0] =0;
graph [4][1] =0;
graph [4][2] =4;
graph[4][3] =3;
graph [4][4] =0;
graph [4][5] =0;
graph [4][6] =0;
graph[5][0] =0;
graph [5][1] =0;
graph[5][2] =2;
graph [5][3] =0;
graph [5][4] =3;
graph [5][5] =0;
graph [5][6] =0;
  kruskal Algo ();
  print ();
}  getch();
```

expected output

elements of the graph are

2-1 : 2

5-2 : 2

3-2 : 3

4-3 : 3

1-0 : 4

spanning tree cost : 14