1. Progam1

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 10
char stack [SIZE];
int top ;
void push (char symbol)
{
stack[++top] = symbol;
}
char pop ()
{
return (stack[top--]);
}
int priority (char op)
{
switch (op)
{
case '#': return 0;
case '(':return 1;
case '+':
case '-':return 2;
case '*':
case '/':return 3;
}
}
void infixToPostfix (char *infix, char* postfix)
{
char symbol, brace;
int i = 0, k = 0;
```

```
push ('#');
while ( ( symbol = infix[i++] ) != '\0')
{
if ( isalnum (symbol) )
 postfix [k++] = symbol;
 else if (symbol == '(')
 push (symbol);
 else if (symbol == ')')
{
 while (stack[top] != '(')
 postfix [k++] = pop ();
 brace = pop ();
 }
else
{
 while (priority(stack[top]) >= priority(symbol))
 postfix [k++] = pop ();
 push(symbol);
 }
}
while (stack[top] != '#')
 postfix [k++] = pop ();
postfix[k] = '\0';
}
void main()
{
char infix [50], postfix [50];
top = -1;
printf ("\n Input the Infix Expression: ");
scanf ("%s", infix);
infixToPostfix (infix, postfix);
printf ("\n Postfix Expression: %s", postfix);
}
```

2.program 2

```
#include<stdio.h>
#define MAX 20
typedef struct stack
{
int data[MAX];
int top;
}stack;

void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *,int);
int evaluate(char x,int op1,int op2);
int main()
{
stack s; char x;
int op1,op2,val;
init(&s);
printf("Enter the expression(eg: 59+3*)\nSingle digit operand
and operators
only:");
while((x=getchar())!='\n')
{
if(isdigit(x))
push(&s,x-48); //x-48 for removing the effect of ASCI
else
{
op2=pop(&s);
op1=pop(&s);
val=evaluate(x,op1,op2);
push(&s,val);
}
}
val=pop(&s);
printf("\nValue of expression=%d",val);
return 0;
}
int evaluate(char x,int op1,int op2)
{
if(x=='+')
return(op1+op2);
if(x=='-')
return(op1-op2);
if(x=='*')
return(op1*op2);
if(x=='/')
return(op1/op2);
if(x=='%')
return(op1%op2);
}
void init(stack *s)
{
s->top=-1;
}
int empty(stack *s)
{
if(s->top==-1)
return(1);
return(0);
}
int full(stack *s)
{
if(s->top==MAX-1)
```

```
return(1);
return(0);
}
void push(stack *s,int x)
{
s->top=s->top+1;
s->data[s->top]=x;
}
int pop(stack *s)
{
int x;
x=s->data[s->top];
s->top=s->top-1;
return(x);
}
```

2. QUEUE

```
#include <stdio.h>
#include <stdlib.h>
# define MAX 5
int Q [max];
int front, rear ;
void enqueue (int item)
{
if ( rear == MAX-1 )
{
 printf ("\n ** Can't insert as it leads Overflow ** \n");
 return;
}
if ( front == -1 )
 front ++;
Q [ ++rear ] = item;
if ( rear == MAX-1 )
 printf ("\n ** Full **\n");
}
int dequeue ( )
{
 int item;
 if ( front == -1 )
 {
printf (" ** Can't remove as it leads 'Underflow **\n");
return;
 }
 item = Q [ front ];
 if ( front == rear )
 {
front = rear = -1;
printf (" ** Empty **\n");
```

```
 }
 else
front ++;
 printf("\n Deleted element is: %d \n", item) ;
}
void display ( )
{
 int temp;
 if ( front == -1 )
 {
printf ("\n **Empty..\n");
return;
 }
 for ( temp = front ; temp <= rear ; temp++ )
printf (" %d ", Q [temp] );
}
void main()
{
 int choice, item;
 front = rear = -1; /* Initialize queue to empty */
 while(1)
 {
 printf("\n Queue Implementation");
 printf("\n 1. Insert.");
 printf("\n 2. Delete.");
 printf("\n 3. Display.");
 printf("\n 4. Exit.");
 printf ("\n Enter your choice: ");
 scanf ("%d", &choice);
 switch(choice)
 {
case 1:
printf("\n Enter the element to be inserted: ");
scanf("%d", &item);
enqueue(item);
break;
case 2:
dequeue();
break;
case 3:
printf("\n The Contents of the Queue are: \n");
display();
break;
default:
exit (0);
 }
 }
}
```

4.linked list

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
{
int data;
struct node *next;
}*start=NULL,*q,*t;
int main()
{
int ch;
void insert();
void display();
void delete_beg();
void delete_end();
int delete_pos();
while(1)
{
printf("\n\n---- Singly Linked List(SLL) Menu ");
printf("\n1.Insert\n2.Display\n3.Delete\n4.Exit\n\n");
printf("Enter your choice(1-4):");
scanf("%d",&ch);
switch(ch)
{
case 1:insert();
break;
case 2: display(); break;
case 3: printf("\n---- Delete Menu ");
printf("\n1.Delete from beginning\n2.Delete from
end\n3.Delete from specified position\n4.Exit");
printf("\n\nEnter your choice(1-4):"); scanf("%d",&ch);
switch(ch)
{
case 1: delete_beg(); break;
case 2: delete_end(); break;
case 3: delete_pos(); break;
case 4: exit(0);
default: printf("Wrong Choice!!");
}
break; case 4: exit(0);
default: printf("Wrong Choice!!");
}
}
return 0;
}
void insert()
{
int num;
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
t->data=num;
if(start==NULL)
{
t->next=NULL; start=t;
}
else
{
t->next=start; start=t;
}
}
void display()
{
if(start==NULL)
```

```
{
printf("List is empty!!");
}
else
{
q=start;
printf("The linked list is:\n");
while(q!=NULL)
{
printf("%d->",q->data);
q=q->next;
}
}
}
void delete_beg()
{
if(start==NULL)
{
printf("The list is empty!!");
}
else
{
q=start;
start=start->next;
printf("Deleted element is %d",q->data);
free(q);
}
}
void delete_end()
{
if(start==NULL)
{
printf("The list is empty!!");
}
else
{
q=start;
while(q->next->next!=NULL)
q=q->next;
t=q->next;
q->next=NULL;
printf("Deleted element is %d",t->data);
free(t);
}
}
int delete_pos()
{
int pos,i;
if(start==NULL)
{
printf("List is empty!!"); return 0;
}
printf("Enter position to delete:");
scanf("%d",&pos);
q=start; for(i=1;i<pos-1;i++)
{
if(q->next==NULL)
{
printf("There are less elements!!");
return 0;
}
q=q->next;
}
t=q->next;
q->next=t->next;
printf("Deleted element is %d",t->data);
```

```
free(t);
return 0;
}
```

5. liner and binary search

```
#include <stdio.h>
int linearSearch (int*, int, int);
int binarySearch (int*, int, int, int);
int main()
{
int choice, arr [10], n, key, index;
printf ("Enter the size of the array : ");
scanf ("%d", &n);
printf ("Selct from Menu : \n 1. Linear Search. \n 2. Binary
Search. ");
printf ("Enter Your Choice : ");
scanf ("%d", &choice);
if (choice == 1)
printf ("Enter the array Elements: \n");
else
printf (" Enter sorted elements for Binary Search **)", n);
for (int i=0; i<n; i++)
scanf ("%d", &arr[i]);
printf ("Enter the Key Element: ");
scanf ("%d", &key);
switch (choice)
{
case 1: index = linearSearch(arr, n, key);
break;
case 2: index = binarySearch (arr, 0, n-1, key);
break;
default : printf ("***** Wrong Choice Entered ******: \n");
break;
 }
if (index == -1)
printf ("\n Element Not found. " );
else
printf ("\n Element found at location %d \n", index);
}
int linearSearch (int *a, int n, int key)
{
int i = 0;
while( i<n )
{
if (a[i]==key)
return i+1;
i++;
}
return -1;
}
int binarySearch (int a[], int left, int right, int key)
{
if (left > right)
return -1;
int mid = ( left + right ) / 2;
if (a[mid] == key)
return mid + 1;
else if (a[mid] < key)
binarySearch (a, left, mid-1, key);
else
binarySearch (a, mid+1, right, key);
}
```

Program 6

```
#include<stdio.h>
#include<stdlib.h>
void display(int a[],int n);
void bubble_sort(int a[],int n);
void selection_sort(int a[],int n);
int main()
{
int n=10,choice,i;
int arr[10];
char ch[20];
printf("Enter no. of elements u want to sort : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter %d Element : ",i+1);
scanf("%d",&arr[i]);
}
printf("Please select any option Given Below for Sorting :
\n");
while(1)
{
printf("\n1. Bubble Sort\n2. Selection Sort\n3. Exit the
Program.\n");
printf("\nEnter your Choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1: bubble_sort(arr,n);
break;
case 2: selection_sort(arr,n);
break;
case 4: return 0;
default: printf("\nPlease Select only 1-3 option \n");
return 0;
}
}
}
void display(int arr[10],int n)
{
int i;
for(i=0;i<10;i++)
{
printf(" %d ",arr[i]);
}
}
void bubble_sort(int arr[],int n)
{
int i,j,temp; for(i=0;i<n;i++)
{
for(j=0;j<n-i-1;j++)
{
if(arr[j]>arr[j+1])
{
temp=arr[j]; arr[j]=arr[j+1]; arr[j+1]=temp;
}
}
}
printf("After Bubble sort Elements are : ");
display(arr,n);
}
void selection_sort(int arr[],int n)
{
int i,j,temp; for(i=0;i<n-1;i++)
{
```

```
for(j=i+1;j<n;j++)
{
if(arr[i]>arr[j])
{
temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;
}
}
}
printf("After Selection sort Elements are : ");
display(arr,n);
}
```

7. Krushkal

```
#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);
void readCostMatrix()
{
printf("\n Enter the number of nodes:");
scanf("%d", &n);
printf("\n Enter the cost 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;
```

```
}
}
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;
}
int main()
{
printf ("\n Kruskal's algorithm to find Minimum Cost
spanning Tree ");
readCostMatrix();
printf ("\n The edges of Minimum Cost Spanning Tree
are\n\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 ("\n %d edge (%d, %d) =%d\n", ne++, a, b, min);
mincost += min;
}
cost [a][b] = cost[b][a] = 999;
}
printf ("\n\t Minimum cost = %d\n", mincost);
}
```

8.dikshreetra

```
#include<stdio.h>
#include<conio.h>
#define infinity 999
void dij(int n,int v,int cost[10][10],int dist[100])
{
int i,u,count,w,flag[10],min;
for(i=1;i<=n;i++)
flag[i]=0,dist[i]=cost[v][i];
count=2;
while(count<=n)
{
min=99;
for(w=1;w<=n;w++)
if(dist[w]<min && !flag[w])
min=dist[w],u=w;
flag[u]=1;
count++;
for(w=1;w<=n;w++)
if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
dist[w]=dist[u]+cost[u][w];
}
}
void main()
{
int n,v,i,j,cost[10][10],dist[10];
clrscr();
printf("\n Enter the number of nodes:");
scanf("%d",&n);
printf("\n Enter the cost 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]=infinity;
}
printf("\n Enter the source matrix:");
scanf("%d",&v);
dij(n,v,cost,dist);
printf("\n Shortest path:\n");
for(i=1;i<=n;i++)
if(i!=v)
printf("%d->%d,cost=%d\n",v,i,dist[i]);
getch();
}
```