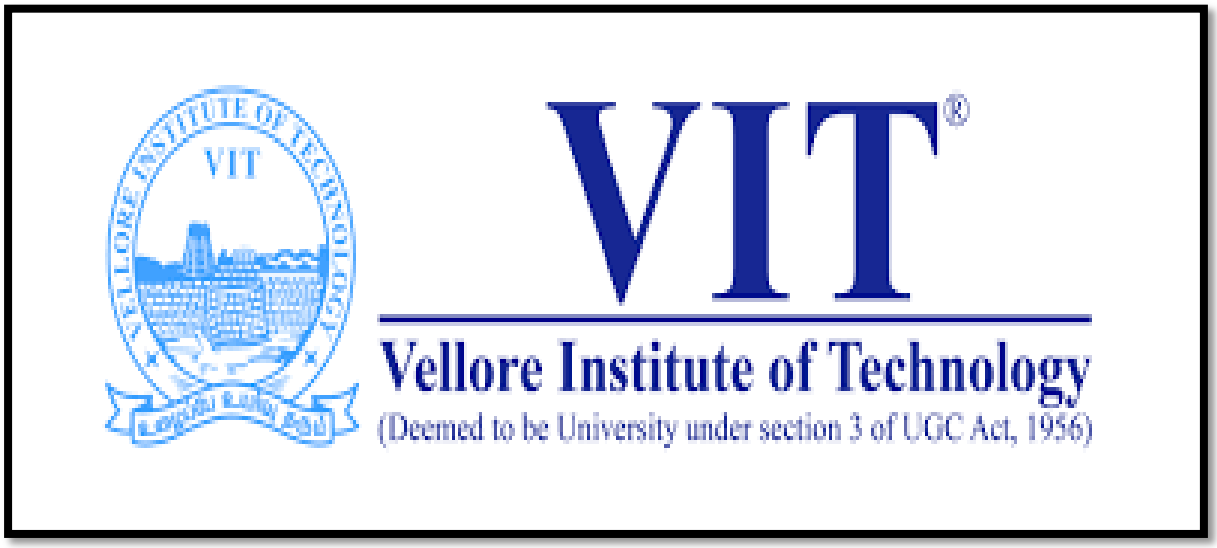

NAME : NITHISH.G

REG NO : 19BCS0012

SUBJECT : DATA STRUCTURE

COURSE : CSC2001

CODE

SLOT : ETH+TE1

① Program Input:-

```
#include<stdio.h>
#include<stdlib.h>
#include<
struct node
{ int data;
  struct node * prev;
  struct node * next;
}*p, *head, *old, *n, *s;
  *p;
void create(); void i;
void insertingatbegin();
void insertmid();
void deletefirst();
void deletemid();
void display();

int main()
{
    int i, D;
do{ printf("\n1.For create list");
   printf("\n2.7 - insert at begin");
```

```
printf("\n 3- insert at middle");
printf("\n 4- delete at first");
printf("\n 5- deletion at middle");
printf("\n 6- display the elements");
printf("\n 7 - to exit");
printf("\n In Enter your choice:");
scanf("%d", &ch);
switch(ch)
{ case 1:
        create();
        break;
  case 2:
        insertbegin();
        break;
  case 3:
        insertmid();
        break;
  case 4:
        deletefirst();
        break;
  case 5:
        deletemid();
        break;
```

```
case 6:
        display();
        break;
case 7:
        exit();
        break;
    default:
        printf("Enter correct option");
} } while(ch!=7)
getch(); }

void create()
{ int i, n;
printf("\n Enter the no. of nodes");
scanf("%d", &n);
printf("\n Enter values");
head = (struct node*) malloc (sizeof (struct node));
scanf("%d", &head->data);
  head->next = Null;
  head->prev = Null;
  old = head;
for (i=0; i<n; i++)
{ n = (struct node*) malloc (sizeof (struct node));
scanf("%d", &n->info);
```

```
n->next = NULL;
n->prev = Null;
old->next=n;
n-> prev=old;
old = old->next;}}
void insat begin()
{ int d ;
   printf("Enter element : ");
   Scanf ("%d", &d);
   n= (struct node*) malloc (size of (struct node
n-> data =0;
n-> next =head;
head -> prev =n;
h-> prev = NULL;
   head =n;
}
void insertmid()
   int i, pos, ele;
   p= old = head;
   printf("\n Enter Position of element:");
   scanf ("%d", &pos);
   Printf ("\n enter the element :");
```

```
Scanf("%d", &ele);
n = (struct node *) malloc (sizeof (struct
                              node));
n->info = ele;
for(i=1; i<=pos; i++)
{ old = p;
  p = p->next; }
old->next = n;
n->prev = old;
n->next = p;
p->prev = n;
}

void deletefirst()
{ struct node *del;
  if (head == NULL)
{ printf("\n nothing to delete");
} else
    { del = head; head = head->next;
    head->prev = NULL;
     free(del); }}
void deletemid()
{ int pos;
Printf("\n Enter the position:");
  Scanf("%d", &pos);
   struct node *current;
```

```
int i;
  current = head;
for (i=1 ;i<pos && current != NULL ;i++)
 { current = current ->next;
 } if (pos = = -1)
 { void deletefirst() }
else
 { current -> Prev -> next = current -> next;
  current -> next -> prev = current -> prev
    free (current);
Printf ("\n the element deleted");
 } }
Void display()
 { struct node *temp;
  int p=1;
  if (head = NULL)
 { printf ("list is empty"); }
else {
      temp = head;
  Printf ("\n the elements are:");
  while (temp != NULL)
  { printf ("%d", temp -> data);
    temp = temp -> next; } } }
```

① Convert the following infix expresion into the Postfix expression.

a) (A-B)*(C+D)

⇒ [AB-CD+*]

⇒ (AB-)*(CD+)

⇒ [AB-CD+*]

b) (A+B)/(C+D)-(D*E)

AB+ CD+ DE*

AB+CD+/-DE*

⇒ [AB+CD+/DE*-]

c) (A+B)*(C*D-E)*F/G

AB+*CD*E-*F/G

⇒ AB*CD*E-*F*G/

d) ((A+B)*C-(D-E)*(F+G))

AB+C*-DE-^FG+

(AB+C*)-(DE-FG+^)

[AB+C*DE-FG+^-]

(2) Prefix:

(A-B) * (C-D)

= (-AB) * (+CD)

⇒ *-AB+C

(b) (A+B)/(C+D)-(D*E)

⇒ (+AB)/(+CD)-(*DE)

⇒ /+AB+CD-(*DE)

⇒ -/AB+CD*DE

(c) (A+B)*(C*D-E)*F/G

⇒ (+AB)*(*(D-E)*F/G

⇒ *+AB*-*(DE)FG

(d) ((A+B)*(C-(D-E))^(F+G)

Prefix:-

⇒ (*(+AB)*(*(-DE)^(+FG)))

⇒ (*(+AB)*(-^(-DE)(+FG))

⇒ -*+AB(^-DE+FG

(3) Convert the following infix expression into postfix expression using stack.

a) A-(B/C+(D%E*F)/G)*H

Input string	output stack	Operator Stack
A-(B/C+(D%E*F)/G)*H	A	
A-(B/C+(D%E*F)/G)*H	A	-(
A-(B/C+(D%E*F)/G)*H	AB	-(
A-(B/C+(D%E*F)/G)*H	AB	-(/
A-(B/C+(D%E*F)/G)*H	ABC	-(/
"	ABC/	-(/+
"	ABC/	-(/+(
"	ABC/D	-(/+(
"	ABC/D	-(/+(%
A-(B/C+(D%E*F)/G*H	ABC/DE	-(/+(%
	ABC/DE%	-(/+(*
	ABC/DE%F	-(/+(*
	ABC/DE%F*	-(/+
	ABC/DE%F*	-(/+
	ABC/DE%F*G	-(/+
	ABC/DE%F*G/+	-*
	ABC/DE%F*G/+H	-*
	ABC/DE%F*G/+H	-**

A-(B/C+(D%E*F)/G*H ABC/DE%F*G/+H**

(b) (A-2*(B+C)/D*E)+F

Input String	output stack	operator stack
(A-2*(B+C)/D*E)+F		(
A-2*(B+C)/D*E)+F	A	-(
-2*(B+C)/D*E)+F	A 2	-(
2*(B+C)/D*E)+F	A 2	-(*
*(B+C)/D*E)+F	A 2	-(*(
(B+C)/D*E)+F	A 2 B	-(*(
B+C)/D*E)+F	A 2 B	-(*(+
+C)/D*E)+F	A 2 BC	-(*(+
C)/D*E)+F	A 2 BC+	-(*
)/D*E)+F	A 2 BC+*	-(
/D*E)+F	A 2 BC+*D	-(/
D*E)+F	A 2 BC+*D/	-(*
*E)+F	A 2 BC+*D/E	-(*
E)+F	A 2 BC+*D/E*	-
)+F	A 2 BC+*D/E*-	+
+F	A 2 BC+*D/E*-F	+
F	A 2 BC+*D/E*-F+	