VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belgaum, Karnataka

RAJARAJESWARI COLLEGE OF ENGINEERING, BENGALURU

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

DATA STRUCTURES LABORATORY (22MCA13)

Prepared BY:

Mrs.Priyanka V Gudada

Asst.prof, MCA Dept.,RRCE.

Data structure Lab Programs

1 Implement a Program in C for converting an Infix Expression to Postfix Expression.

2 Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

3 Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display

4 Write a C program to simulate the working of a singly linked list providing the following operations: a. Display & Insert b. Delete from the beginning/end c. Delete a given element

5 Write a C program to Implement the following searching techniques a. Linear Search b. Binary Search.

6 Write a C program to implement the following sorting algorithms using user defined functions: a. Bubble sort (Ascending order) b. Selection sort (Descending order).

7 Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm (C programming)

8 From a given vertex in a weighted connected graph, find shortest paths to other vertices Using Dijkstra's algorithm (C programming)

Demonstration Experiments (For CIE) if any

9 Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.

10 Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic
operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

PROGRAM 1

Implement a Program in C for converting an Infix Expression toPostfix Expression.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <string.h>
char infix_string[20], postfix_string[20];
int top;
int stack[20];
int pop();
int precedence(char symbol);
 int isEmpty();
void infix_to_postfix();
int check_space(char symbol);
void push(long int symbol);

int main()
{
int count, length;
char temp;
top = -1;
printf("\nINPUT THE INFIX EXPRESSION : ");
scanf("%s", infix_string);
 infix_to_postfix();
printf("\nEQUIVALENT POSTFIX EXPRESSION :%s\n",postfix_string);
return 0;
}

void infix_to_postfix()
{
unsigned int count, temp = 0;
char next;
char symbol;
for(count = 0; count < strlen(infix_string); count++)
{
symbol = infix_string[count]; // Scanning the input expression
if(!check_space(symbol))
{
switch(symbol)
```

```
{
case '(': push(symbol);
break;
case ')':
while((next = pop()) != '(')       // pop until '(' is encountered
{
postfix_string[temp++] = next;
}
  break;
 case '+':
case '-':
case '*':
case '/':
case '%':
case '^':
while(!isEmpty() && precedence(stack[top]) >=precedence(symbol))
            // Check precedence and push the higher one
  postfix_string[temp++] = pop();
  push(symbol);
  break;
  default: postfix_string[temp++] = symbol;
}
}
}
while(!isEmpty())
{
postfix_string[temp++] = pop();
}
postfix_string[temp] = '\0';
}

int precedence(char symbol)
{
switch(symbol)
{
case '(': return 0;
 case '+':
case '-': return 1;
case '*':
case '/':
case '%': return 2;
case '^': return 3;
default: return 0;
```

```
}
}

int check_space(char symbol)
{
if(symbol == '\t' || symbol == ' ' )
{
return 1;
}
else
{
return 0;
}
}

void push(long int symbol)
{
if(top > 20)
{
printf("Stack Overflow\n");
 exit(1);
}
top = top + 1;
stack[top] = symbol;   // Push the symbol and make it as TOP
}

int isEmpty()
{
if(top == -1)
{
return 1;
}
else
{
return 0;
}
}

int pop()
{
if(isEmpty())
{
printf("Stack is Empty\n");
```

```
 exit(1);
}
return(stack[top--]);  // Pop the symbol and decrement TOP
}
```

OUTPUT :

```
INPUT THE INFIX EXPRESSION: A+B*(C-D)/E

EQUIVALENT POSTFIX EXPRESSION: ABCD-*E/+
```

PROGRAM 2

Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

```
#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);
}
```

OUTPUT :

```
Enter the expression(eg: 59+3*)

Single digit operand and operators only: 45+3*25+/

Value of expression = 3
```

PROGRAM 3

Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display

```
#include<stdio.h>
#include<stdlib.h>
void insert();
void delete1();
void display();
int front = -1, rear = -1 ,maxsize;
int queue[100];
int main ()
{
   int choice;
   printf("\n Enter the size of QUEUE : ");
   scanf("%d",&maxsize);
   printf("\n QUEUE OPERATIONS USING ARRAY");
   printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n 4.Exit");
   while(choice != 4)
   {
      printf("\nEnter your choice : ");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1: insert();
         break;
```

```
            case 2:  delete1();
            break;
            case 3:  display();
            break;
            case 4:  exit(0);
            break;
            default:  printf("\nEnter valid choice??\n");
        }
    }
    return 0;
}

void insert()
{
    int item;
    printf("\nEnter the element\n");
    scanf("\n%d",&item);
    if(rear == maxsize-1)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    if(front == -1 && rear == -1)
    {
        front = 0;
        rear = 0;
    }
    else
    {
        rear = rear+1;
    }
    queue[rear] = item;
    printf("\nValue inserted ");

}

void delete1()
{
    int item;
    if (front == -1 || front > rear)
    {
        printf("\nUNDERFLOW\n");
        return;
        }
    else
    {
        item = queue[front];
        if(front == rear)
        {
            front = -1;
            rear = -1 ;
        }
        else
        {
            front = front + 1;
        }
        printf("\nvalue deleted ");
    }
```

```
}

void display()
{
   int i;
   if(rear == -1)
   {
      printf("\nEmpty queue\n");
   }
   else
   {   printf("\n Elements in the queue are\n");
      for(i=front;i<=rear;i++)
      {
         printf("\n%d",queue[i]);
      }
   }
}
```

<u>OUTPUT</u> :

```
Enter the size of QUEUE : 5

 QUEUE OPERATIONS USING ARRAY
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice : 1
Enter the element
6
Value inserted
Enter your choice : 1
Enter the element
```

```
7
Value inserted
Enter your choice : 1
Enter the element
8
Value inserted
Enter your choice : 1
Enter the element
9
Value inserted
Enter your choice : 3
 Elements in the queue are
6
7
8
9
Enter your choice : 2
value deleted
Enter your choice : 3
 Elements in the queue are
7
8
9
Enter your choice : 2
value deleted
Enter your choice : 3
 Elements in the queue are
8
9
Enter your choice : 4
```

PROGRAM 4

Write a C program to simulate the working of a singly linked list providing the following operations: a. Display & Insert b. Delete from the beginning/end c. Delete a given element

```
#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_beg();
void insert_end();
 int insert_pos();
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:
printf("\n---- Insert MenU---");
printf("\n1.Insert at beginning\n2.Insert at end\n3.Insert at specified
position\n4.Exit");
 printf("\n\nEnter your choice(1-4):");
scanf("%d",&ch);

switch(ch)
{
case 1: insert_beg(); break;
case 2: insert_end(); break;
case 3: insert_pos(); break;
case 4: exit(0);
default: printf("Wrong Choice!!");
}
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;
}
return 0;
}

void insert_beg()
{
int num;
t= (struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
t->data=num;
if(start==NULL)  //If list is empty
{
t->next=NULL;
start=t;
}
else
{
t->next=start;
start=t;
}
}

void insert_end()
```

```
{
int num;
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
t->data=num;
t->next=NULL;
if(start==NULL)  //If list is empty
{
start=t;
}
else
{
q=start;
while(q->next!=NULL)
q=q->next;
q->next=t;
}
}

int insert_pos()
{
int pos,i,num;
if(start==NULL)
{
printf("List is empty!!");
return 0;
}
```

```
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
printf("Enter position to insert:");
scanf("%d",&pos);
t->data=num;
q=start; for(i=1;i<pos-1;i++)
{
if(q->next==NULL)
{
printf("There are less elements!!");
return 0;
}
q=q->next;
}
t->next=q->next;
q->next=t;
return 0;
}
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;
}
```

OUTPUT :

```
---- Singly Linked List(SLL) Menu
1.Insert
2.Display
3.Delete
4.Exit
Enter your choice(1-4): 1
---- Insert Menu
1.Insert at beginning
2.Insert at end
3.Insert at specified position
4.Exit
Enter your choice(1-4):1
Enter data : 10
---- Singly Linked List(SLL) Menu
1.Insert
2.Display
3.Delete
4.Exit
Enter your choice(1-4): 2
The liked list is
10->
---- Singly Linked List(SLL) Menu
1.Insert
2.Display
3.Delete
4.Exit
Enter your choice(1-4): 3
---- Delete Menu
1.Delete from beginning
2.Delete from end
3.Delete from specified position
4.Exit
Enter your choice(1-4):1
Deleted element is 10
```

PROGRAM 5

Write a C program to implement the following searching techniques a. Linear Search b. Binary Search.

```
#include<stdio.h>
void main()
{
int a[25], beg, item, last, n, num, i, ch, mid, f=0;
printf ("menu\n");
printf ("\n 1.linear search");
printf ("\n 2.binary search");
printf ("\n enter the choice");
scanf ("%d", &ch);
if (ch==1)
{
printf ("\n enter the number of elements in the array");
scanf ("%d",&n);
printf ("\n enter the sorted array");
for(i=0;i<n;i++)
scanf ("%d", &a[i]);
printf ("\n enter the item to be searched");
scanf ("%d", &item);
for(i=0; i<n; i++)
{
if(a[i]==item)
{
printf ("\n item found at position %d", i+1);
break;
}
}
if (i==n)
printf ("\n item not found");
}
if (ch ==2)
{
printf ("\n enter the number of elements in the array");
scanf ("%d", &n);
printf ("enter the sorted array");
for(i=0; i<n; i++)
scanf ("%d", &a[i]);
printf ("item to be searched");
scanf ("%d", &item);
last=n-1; mid=(beg +last)/2;
while (beg<=last)
{
if (item == a[mid])
{
printf ("\n item found at position %d", mid+1);
break;
}
else if(a[mid]>item)
last =mid-1;
else
beg=mid+1;
mid =(beg + last)/2;
}
}
```

```
}
```

OUTPUT :

```
MENU
 1.linear search
 2.binary search
 enter the choice 1
 enter the number of elements in the array 5
 enter the sorted array 10 20 30 40 50
 enter the item to be searched  40
 item found at position 4

MENU
 1.linear search
 2.binary search
 enter the choice 2
 enter the number of elements in the array 5
 enter the sorted array 10 20 30 40 50
 enter the item to be searched 30
 item found at position 3
```

PROGRAM 6

Write a C Program to Implement the following sorting algorithm using user defined functions: a. bubble Sort (Ascending Order) b. Selection Sort (Descending Order).

```
#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);
//-----------------Main Function----------------------
int main()
{
int n,choice,i,arr[10];
char ch[20];
printf("Enter no. of elements u want to sort : ");
scanf("%d",&n);
int arr[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. Display Array.\n 4. 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 3: display(arr,n);
break;
case 4: return 0;
default: printf("\nPlease Select only 1-4 option     \n");
return 0;
}
}
}
//     End of main function
//     Display Function
void display(int arr[],int n)
{
Int i;
for(i=0;i<n;i++)
{
printf(" %d ",arr[i]);
}
}
//     Bubble Sort Function
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);
}
//------------------Selection Sort Function---------
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);
}
```

OUTPUT :

```
Enter no. of elements u want to sort : 5
Enter 1 Element : 45
Enter 2 Element : 12
Enter 3 Element : 78
Enter 4 Element : 3
Enter 5 Element : 56
Please select any option Given Below for Sorting :
1. Bubble Sort
2. Selection Sort
3. Display Array.
4. Exit the Program.
Enter your Choice : 1
After Bubble sort Elements are : 3  12  45  56  78

1. Bubble Sort
2. Selection Sort
3. Display Array.
4. Exit the Program.
Enter your Choice : 2
After Selection sort Elements are : 78 56  45 12 3

1. Bubble Sort
```

```
2. Selection Sort
3. Display Array.
4. Exit the Program.
Enter your Choice : 3
 3  12  45  56  78

1. Bubble Sort
2. Selection Sort
3. Display Array.
4. Exit the Program.
Enter your Choice : 4
```

PROGRAM 7

Find Minimum Cost Spanning tree of a given undirected graph using kruskal’s algorithm (C programming).

```
#include<stdio.h>
#include<stdlib.h>
#define VAL 999
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
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("Implementation of Kruskal's algorithm\n");
printf("Enter the no. of vertices:");
```

```
scanf("%d",&n);
printf("Enter the cost 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]=VAL;
}
}
printf("The edges of Minimum Cost Spanning Tree are\n");
while(ne < n)
{
for(i=1,min=VAL;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("%d edge (%d,%d) =%d\n",ne++,a,b,min);
mincost +=min;
}
cost[a][b]=cost[b][a]=999;
}
printf("\n\tMinimum cost = %d\n",mincost);
return 0;
}
```

OUTPUT :

```
Implementation of Kruskal's algorithm
Enter the no. of vertices: 5
Enter the cost adjacency matrix:
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) =2
2 edge (2,3) =3
3 edge (1,4) =6
4 edge (2,5) =5
        Minimum cost = 16
```

PROGRAM 8

From a given vertex in a weighted connected graph find shortest paths to other vertices using Dijkstra’s algorithm(C programming).

```
#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();
}
```

OUTPUT

```
Enter the number of nodes: 5
Enter the cost matrix:
0 10 3 0 0
0 0 1 2 0
0 4 0 8 2
0 0 0 0 7
```

```
0 0 0 9 0
Enter the source node: 1
Shortest path:
1->2, cost=7
1->3, cost=3
1->4, cost=9
1->5, cost=5
```

PROGRAM 9

Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.

```
#include<stdio.h>
#include<stdio.h>
struct node
{
float coeff;
int expo;
struct node* link;
};

struct node * insert(struct node* head, float co, int ex)
{
struct node* temp;
struct node* newp =malloc(sizeof(struct node));
newp->coeff =co;
newp->expo=ex;
newp->link=NULL;

if(head==NULL||ex>head->expo)
{
newp->link= head;
head=newp;
}
else
{
temp=head;
while(temp->link!=NULL && temp->link->expo >=ex)
temp=temp->link;
newp->link=temp->link;
temp->link= newp;
}
return head;
}
struct node* create(struct node * head)
{
int n ,i;
```

```
float coeff;
int expo;
printf("Enter the number of terms:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the coefficient for term %d:",i+1);
scanf("%f",&coeff);
printf("Enter the exponent for term %d:",i+1);
scanf("%d",&expo);
head=insert(head,coeff,expo);
}
return head;
}
void print(struct node* head)
{
if(head==NULL)
printf("No Polynomials");
else{
struct node* temp=head;
while(temp!=NULL)
{
printf("(%.1fx^%d)", temp-> coeff,temp->expo);
temp=temp->link;
if(temp!=NULL)
printf("+");
else
printf("\n");
}
}
}
void polyadd(struct node* head1,struct node* head2)
{
struct node* ptr1=head1;
struct node* ptr2=head2;
struct node* head3=NULL;
while(ptr1!=NULL && ptr2!=NULL)
{
if(ptr1->expo==ptr2->expo)
{
head3=insert(head3,ptr1->coeff+ptr2->coeff,ptr1->expo);
ptr1=ptr1->link;
ptr2=ptr2->link;
}
else if(ptr1->expo > ptr2->expo)
{
head3=insert(head3,ptr1->coeff,ptr1->expo);
ptr1=ptr1->link;
}
else if(ptr1->expo < ptr2->expo)
{
head3=insert(head3,ptr2->coeff,ptr2->expo);
ptr2=ptr2->link;
}
}
while(ptr1!=NULL)
{
head3=insert(head3,ptr1->coeff,ptr1->expo);
ptr1=ptr1->link;
}
while(ptr2!=NULL)
{
head3=insert(head3,ptr2->coeff,ptr2->expo);
ptr2=ptr2->link;
}
printf("Added Polynomial is:");
print(head3);
}

int main()
{

struct node* head1=NULL;
struct node* head2=NULL;
clrscr();
printf("Enter the first Polynomial\n");
head1=create(head1);
printf("Enter the Second Polynomial\n");
head2=create(head2);

polyadd(head1,head2);
return 0;
getch();
}
```

OUTPUT

```
Enter the First Polynomial
Enter the number of terms : 2
Enter the coefficient for term 1 : 2
Enter the Exponent for term 1 : 2
Enter the coefficient for term 2: 3
Enter the Exponent for term 2 : 1
Enter the Second Polynomial
Enter the number of terms : 3
Enter the coefficient for term 1 : 4
Enter the Exponent for term 1 : 2
Enter the coefficient for term 2: 3
Enter the Exponent for term 2 : 1
Enter the coefficient for term 3: 2
Enter the Exponent for term 3 : 0
Added Polynomial is : (6x^2)+(6x^1)+(2x^0)
```

PROGRAM 10

Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float oper(char sym, float op1, float op2)
{
switch(sym)
{
case '+': return op1 + op2;
case '-': return op1 - op2;
case '*': return op1 * op2;
case '/': if(op2== 0)
                                        {
printf("Can't evalueate");
exit(0);
}
return op1 / op2;
case '^':
case '$': return pow(op1,op2);
}
}

void push(float item, int *top, float s[ ])
{
s[++(*top)] = item;
}

float pop(int *top, float s[ ])
{
return s[(*top)--];
}

void main()
{
float s[20], result, op1, op2, x;
```

```
int top = -1, i;
char postfix[20], sym;

printf("Enter valid postfix expression\n");
scanf("%s",postfix);

for(i=0;i<strlen(postfix);i++)
{
sym = postfix[i];

if(isdigit(sym))
          push(sym-'0', &top, s) // character to digit conversion
else if (isalpha(sym)
{
        printf("Enter the value of %c: ", sym);
         scanf("%f",&x);
        push(x,&top,s);
}
else
{
op2 = pop(&top,s);
op1 = pop(&top,s);
result = oper(sym,op1,op2);
push(result,&top,s);
}
}
result = pop(&top,s);
printf("Result =%.4f",result);
}
```

Sample Output 1:

```
Enter valid postfix expression: 941-3*/

 Result = 1.0000
```