Program 3: Design, develop and execute a program in C to convert a given valid parenthesized infix arithmetic expression to postfix expression and then print both the expressions. The expression consists of single character operands and the binary operators +, - , * , /

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int top = -1;
char stack[40];
void push(char x)
{
        stack[++top] = x;
}
int pop( )
{
        return stack[top--];
}

int prior(char x)
{
int p;
if(x=='(' || x=='#')
   p=1;
if(x=='+' || x=='-')
   p=2;
if(x=='*' || x=='/')
   p=3;
if(x=='^' || x=='$')
   p=4;
return p;
}
```

```
void main( )
{
char infix[30], postfix[30];
int i, j=0;
printf("Enter the infix expression:\n");
gets(infix);
push('#');
for(i=0; i<strlen(infix); i++)
{
        if(isalnum(infix[i]))
                postfix[j++] = infix[i];
        else if(infix[i]=='(')
                push(infix[i]);
        else if(infix[i]==')')
        {
        while(stack[top]!='(')
                postfix[j++]=pop( );
                pop( );
          }
        else
         {
        while(prior(stack[top]) == prior(infix[i]))
                postfix[j++]=pop( );
                push(infix[i]);
        }
  }
while(stack[top] != '#')
postfix[j++] = pop( );
postfix[j]='\0';
printf("The postfix expression is:\n");
puts(postfix);
getch();
}
```

Program 4: Design, develop and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are +(add), -(sub), *(mul) and / (divide).

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#define MAXSIZE 30
int s[MAXSIZE];
int top = -1;

int pop()
{
if(top != -1)
        return s[top--];
else
   {
        printf("Stack underflow\n");
        return0;
   }
}

void push(int item)
{
if (top != MAXSIZE-1)
        s[++top] = item;
else
        printf("\nStak Overflow\n");
}

int op(int op1,int op2,char symbol)
{
switch(symbol)
   {
case '+': return op1 + op2;
case '-': return op1 - op2;
case '*': return op1 * op2;
case '/': return op1 / op2;
}
```

```
int isdig(char symbol1)
{
return (symbol1>='0'&& symbol1<='9');
}

void  main()
{
char symbol, postfix[30];
int a,b,res,i;
printf("Enter postfix expression\n");
scanf("%s", postfix);
for(i=0;i<strlen(postfix);i++)
   {
symbol=postfix[i];
if(isdig(symbol))
push(symbol - '0');
else
       {
          a = pop();
          b = pop();
          res = op(b, a, symbol);
          push(res);
       }
   }
printf("The result of the expression is:  ");
printf("%d\n",pop( ));
return( );
}
```

Program 5: 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>
#define MAXSIZE 5
int f = -1, r = -1, q[MAXSIZE];

void insert( )
{
        int item;
        if(r == MAXSIZE-1)
                printf("OVERFLOW!!!");
        else
        {
                if(f == -1)
                        f=0;
                printf("Enter the items to be inserted:\n");
                scanf("%d",&item);
                q[++r]=item;
        }
}

void delete1(  )
{
        if(f == -1)
                printf("UNDERFLOW!!!");
        else if(f>r)
        {
                 f=r=-1;
                printf("UNDERFLOW!!!");
        }
        else
                printf("item deleted=%d",q[f++]);
}
```

```
void display( )
{
	int i;
	if(f==-1)
		printf("UNDERFLOW!!!");
	else
	{
		printf("The elements are: \n");
		for(i = f;i <= r; i++)
			printf("%d ",q[i]);
	}
}

void main( )
{
	int ch;
	for(;;)
	{
		printf("\n1:INSERT\n2:DELETE\n3:DISPLAY\n4:EXIT\n");
		scanf("%d", &ch);
		switch(ch)
		{
			case 1: printf("INSERTION\n");
				insert( );
				break;
			case 2:printf("DELETION\n");
				delete1( );
				break;
			case 3:printf("DISPLAY\n");
				display( );
				break;
			default  :exit(0);
		}
	}
}
```

Program 6: Design, develop and execute a program in C to implement linked list to insert and delete an element from the list.

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

struct node
{
        int info;
        struct node *link;
};
typedef struct node* NODE;

NODE getnode()
{
        NODE x;
x=(NODE)malloc(sizeof(struct node));
        if(x==NULL)
        {
                printf("out of memory\n");
                exit(0);
}
        return x;
}

void freenode(NODE x)
{
        free(x);
}

NODE insert_front(int item, NODE first)
{
        NODE temp;
        temp=getnode();
        temp->info=item;
        temp->link=first;
        return temp;
}
NODE delete_front(NODE first)
```

```
{
        NODE temp;
        if(first==NULL)
 {
                printf("Link is empty\n");
                return first;
 }
        temp=first;
        temp=temp->link;
        printf("The deleted item is %d \n",first->info);
        freenode(first);
        return temp;
}

NODE insert_rear(int item,NODE first)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
        temp->link=NULL;
        if(first==NULL)
                return temp;
        cur=first;
        while(cur->link!=0)
 {
                cur=cur->link;
 }
        cur->link=temp;
        return first;
}

NODE delete_rear(NODE first)
{
 NODE cur=first,prev=NULL;
        if(cur==NULL)
 {
                printf("Link is empty\n");
                return first;
 }
```

```
        while(cur->link!=NULL)
{
                prev=cur;
                cur=cur->link;
}
        freenode(cur);
        prev->link=NULL;
        return first;
}

void display(NODE first)
{ NODE temp=first;
        if(first==NULL)
{
                printf("NO NODES IN THE LIST!!\n");
                return first;
}
        else
{
                while(temp!=NULL)
{
                        printf("%d ",temp->info);
                        temp=temp->link;
}
}
}

void main()
{
        NODE first=NULL;
        int choice,item;

        for(;;)
{
                printf("\n1.INSERT FRONT\n 2.INSERT REAR\n 3.DELETE FRONT\n
                4.DELETE REAR\n 5.DISPLAY\n");
                scanf("%d",&choice);
                switch(choice)
                {
```

```
                    case 1: printf("Enter the item\n");
                            scanf("%d",&item);
                            first=insert_front(item,first);
                            break;
                    case 2: printf("Enter the item\n");
                            scanf("%d",&item);
                            first=insert_rear(item,first);
                            break;
                    case 3:first=delete_front(first);
                            break;
                    case 4:first=delete_rear(first);
                            break;
                    case 5:display(first);
                            break;
                    default:exit(0);
 }
 }
}
```

Program 7: Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix.

```
#include<stdio.h>
#include<conio.h>
#define SROW 50
#define MROW 20
#define MCOL 20
int main()
{
int mat[MROW][MCOL],sparse[SROW][3];
inti,j,nzero=0,mr,mc,sr,s,elem;
printf("Enter number of rows:\n");
scanf("%d",&mr);
printf("Enter number of column:\n");
scanf("%d",&mc);
printf("Enter the matrix\n");
for(i=1;i<=mr;i++)
   {
for(j=1;j<=mc;j++)
      {
scanf("%d",&mat[i][j]);
if(mat[i][j]!=0)
         {
nzero++;
         }
      }
   }
sr=nzero+1;
sparse[1][1]=mr;
sparse[1][2]=mc;
sparse[1][3]=nzero;
   s=2;
for(i=1;i<=mr;i++)
   {
for(j=1;j<=mc;j++)
      {   if(mat[i][j]!=0)
         {
sparse[s][1]=i;
sparse[s][2]=j;
sparse[s][3]=mat[i][j];
s++;
         }
      }
   }
printf("\nSparse matrix is \n");
for(i=1;i<=sr;i++)
   {
for(j=1;j<=3;j++)
      {
printf("%d ",sparse[i][j]);
```

```
        }
printf("\n");
    }
printf("Enter the element to be searched \n");
scanf("%d",&elem);
for(i=2;i<sr;i++)
    {
if(sparse[i][3]==elem)
        {printf("Element found at (row,col)=(%d,%d)",sparse[i][1],sparse[i][2]);
getch();
return 1;
        }
    }
printf("Element not found");
getch();
return 0;

}
```

Sample Output:

Enter number of rows: 3
Enter number of columns: 3
Enter the Matrix:
0 0 1
0 0 2
0 0 3
Sparse matrix is:

3	3	3
1	3	1
2	3	2
3	3	3

Enter the element to be searched: 3
Element found at (row, col) : (3, 3)

Program 8: Design, develop, and execute a program in C to create a max heap of integers by accepting one element at a time and by inserting it immediately in to the heap. Use the array representation for the heap. Display the array at the end of insertion phase.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXSIZE 10
int insertion(int item, int a[ ], int n)
{
int c,p;
if(n==MAXSIZE)
{
        printf("HEAP IS FULL!!!\n");
        return;
 }
 c=n;
 p=(c-1)/2;
while(c!=0&&item>a[p])
 {
        a[c]=a[p];
        c=p;
        p=(c-1)/2;
   }
a[c]=item;
return n+1;
}

void display(int a[],int n)
{
int i;
if(n==0)
{
        printf("HEAP IS EMPTY!!!\n");
        return;
 }
printf("The array elements are \n");
for(i=0;i<n;i++)
        printf("%d ",a[i]);
}
```

```
void main( )
{
int a[MAXSIZE],n=0,ch,item;

for(;;)
{
printf("\n1.INSERT\n2.DISPLAY\nEXIT\n");
scanf("%d",&ch);


switch(ch)
      {
           case 1:   printf("Enter the element");
                     scanf("%d",&item);
                     n=insertion(item,a,n);
                     break;
case 2: display(a,n);
          break;
default: exit(0);
      }
   }
}
```

Program 9:Design, develop and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations:

i. Create a doubly linked list by adding each node at the front.

ii. Insert a new node to the left of the node whose key value is read as an input. iii. Delete the node of a given data if it is found, otherwise display appropriate message iv. Display the contents of the list.

(Note: Only either (a, b and d) or (a, c and d) may be asked in the examination)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *llink;
struct node *rlink;
};
struct node *first, *cur, *prev, *next;

void insertfront()
{
        struct node *temp;
temp = (struct node*)malloc(sizeof(struct node));
temp->llink= temp->rlink= NULL;
printf("enter the node to be inserted \n");
scanf("%d",&temp->info);
if(first==NULL)
{
first=temp;
         return;
         }
temp->rlink=first;
        first->llink=temp;
first=temp;
}
void insertleft()
{
        int key;
```

```
struct node *temp;
     temp = (struct node*) malloc(sizeof(struct node));
               temp->llink= temp->rlink= NULL;
           if(first==NULL)
{
printf("list is empty \n");
           return;
}
printf("enter the key before which node is to be inserted \n");
          scanf("%d",&key);
printf("enter the node to be inserted \n");
scanf("%d",&temp->info);
prev=NULL;
cur=first;
while(cur!=NULL)
{
                    if(cur->info==key && cur==first)
{
temp->rlink=cur;
cur->llink=temp;
first=temp;
return;
}
if(cur->info==key)
{
temp->rlink=cur;
cur->llink=temp;
temp->llink=prev;
prev->rlink=temp;
return;
}
prev=cur;
cur=cur->rlink;
}
printf("key not found");
}
void Delete1( )
{
```

```
        int key;
printf("enter the node to be deleted \n");
scanf("%d", &key);
if(first==NULL)
{
printf("list is empty \n");
return;
}
if(first->rlink==NULL)
{
if(key==first->info)
{
                        printf("%d node is deleted", key);
free(first);
first = NULL;
return;
}
else
{
printf("key not found \n");  return;
}
}
cur=first;
while(cur!=NULL)
{
if (cur->info==key)
{
if(cur == first)
{
first=first->rlink;
                         printf("%d node is deleted", key);
free(cur);
return;
}

                        if(cur->rlink==NULL)
{
prev=cur->llink;
```

```
                                prev->rlink=NULL;
 printf("%d node is deleted \n", key);  free(cur);
 return;
 }
                                prev=cur->llink;
 next=cur->rlink;
                        prev->rlink=next;
 next->llink=prev;
 printf("%d node is deleted \n", key);  free(cur);
 return;
 }
 cur=cur->rlink;
 }
 printf("key not found \n");
 }

void display( )
 {
          cur=NULL;
          if(first==NULL)
 {
 printf("list is empty \n");
 return;
 }
 printf("list is: \n");
 cur=first;
 while(cur!=NULL)
 {
 printf("%d \n", cur->info);
 cur=cur->rlink;
 }
 }
void main( )
 {
 int ch;
 printf("enter 1.insert at front \n 2.insert before a node \n 3.delete at node\n 4.display\n");
for(;;)
 {
```

```
printf("enter the choice \n");
scanf("%d", &ch);
switch(ch)
{
case 1: insertfront();
break;
case 2: insertleft();
break;
case 3: Delete1( );
break;
case 4: display( );
break;
default : exit(0);
}
}
}
```

Design, develop and execute a program in C to create a Binary Tree and to perform inorder, preorder and postorder traversals.

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node
{
int info;
struct node *left;
struct node *right;
};
typedef struct node NODE;
NODE *root=NULL;

void create (int x)
{
NODE *temp,*prev,*cur;
temp=(NODE*)malloc(sizeof(NODE));
temp->left=NULL;
temp->right=NULL;
temp->info=x;
if(root==NULL)
{
root=temp;
return;
}
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
if(x<cur->info) cur=cur->left;
else
if(x>cur->info)cur=cur->right;
else
{
printf("\n duplicate value not allowed");
return;
}
}
if(x<prev->info)
prev->left=temp;
else
prev->right=temp;
}
void preorder(NODE *root)
{
if(root!=NULL)
{
```

```
printf("%d",root->info);
preorder(root->left);
preorder(root->right);
}
}
void inorder(NODE *root)
{
if(root!=NULL)
{
inorder(root->left);
printf("%d",root->info);
inorder(root->right);
}
}
void postorder(NODE *root)
{
if(root!=NULL)
{
postorder(root->left);
postorder(root->right);
printf("%d",root->info);
}
}

void main()
{
int item,choice,flag;
while(1)
{
clrscr();
printf("\n..... binary search tree...\n\n");
printf("1.create\n2.preorder\n3.inorder\n4.postorder\n5.exit\n");
printf("\n enter ur choice");
scanf("%d",&choice);
switch(choice)
{
case 1 :printf("\n enter element to b inserted:");
scanf("%d",&item);
create(item);
break;
case 2: preorder(root);
break;
case 3:inorder(root);
break;
case 4:postorder(root);
break;
default: return;
}
getch();
}
```

}

Program 1: Write a C program to store employee details using structures and perform the following operations

a. To display details of all employees.

a. To search for a specific employee based on the employee id and if found, display the employee details. In case the employee id does not exist, suitable message should be displayed. Both the options in this case must be demonstrated.

```
#include <stdio.h>
#include <stdlib.h>
struct employee
{
        int id;
        char name[20];
        float salary;
};

void main( )
{
int i, n, ch, searchid;
struct employee emp[5];
printf("Enter the number of employees: ");
scanf("%d", &n);
printf("Enter %d employee details: \n", n);
for (i=0; i<n; i++)
{
        printf("Enter employee id: ");
        scanf("%d", &emp[i].id);
        printf("Enter employee name: ");
        scanf("%s", emp[i].name);
        printf("Enter employee salary: ");
        scanf("%f", &emp[i].salary);
}
while(1)
{
printf("\n1. Display \n 2. Search \n3. Exit \n Enter your choice ");
scanf("%d",&ch);
switch(ch)
{
case 1 : for (i=0; i < n; i++)
        {
```

```
        printf("\nEmployee id: %d, Name: %s, Salary: Rs.%f", emp[i].id, emp[i].name,
        emp[i].salary);
        }
        break;

case 2 : printf("Enter Emp ID to be searched:");
        scanf("%d", &searchid);

        for (i=0; i < n; i++)
        {
                if(emp[i].id== searchid)
                  {
                        printf("\nEmployee id: %d, Name: %s, Salary: Rs.%f", emp[i].id,
                        emp[i].name, emp[i].salary);
                        break;
                  }
        }
        if(i==n)
                printf("Empolyee ID not found");
        break;

case 3: exit(0);
}
}
}
```

Program 2: Design, develop and execute a program in C to simulate the working of a stack of integers using an array. Provide the following operations: (a) Push (b) Pop (c) Display.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 3

int s[MAX_SIZE], top = -1;
void push( )
{ int x;
  if(top == MAX_SZE – 1)
  {
        printf("Stack  overflow\n");
        return;
}
printf("Enter element: ");
scanf("%d", &x)
top =top+1;
s[top] = x;
}

int pop( )
{       if(top == -1)
        {       printf("Stack   underflow\n");
                return;
        }
        printf("Popped element is %d", s[top]);
        top=top-1;
}

void display( )
{       int i;
        if (top == -1)
        {
                printf("Stack is empty");
                return;
        }
        else
        {
        printf("Contents of the stack: \n");
        for(i=0;  i<=top; i++)
        {
                printf("%d   ", s[i]);
```

```
        }
}

void main( )
{
int ch, x;
for(;;)
{
printf(1.Push     2.Pop     3.Display)
printf("Enter the choice:   ");
scanf("%d",&ch);
switch (ch)
{
case 1: push( );
        break;
case 2: pop( );
        break;
case 3: display( );
        break;
default: exit(0);
}
}
}
```