ALL PROGRAMS

1a)

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int stack[100], ch, top, x,i;
void push();
void pop();
void display();
int main()
{
        top=-1;
        printf("1:push 2:pop 3:display \n");
        for(;;)
        {
                printf("enter your choice: \n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:push();
                        break;
                        case 2:pop();
                        break;
                        case 3:display();
                        break;
                        default: printf("Invalid choice \n");
                        exit(0);
                }
        }
}

void push()
{
        if(top==SIZE-1)
        {
                printf("stack overflow \n");
        }
        else
        {
                printf("enter the element to be pushed:\n");
                scanf("%d",&x);
                top=top+1;
                stack[top]=x;
        }

}

void pop()
{
        if(top==-1)
        {
                printf("stack underflow\n");
        }
        else
        {
                printf("the poped element is %d", stack[top]);
                top--;
        }
}
```

```
void display()
{
        if(top>=0)
        {
                printf("the element are:\n");
                for(i=top; i>=0; i--)
                {
                        printf("%d\n", stack[i]);
                }

        }
        else{
                printf("stack is empty");
        }
}
```

1b)

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

char stack[100];
int top = -1;

void push(char x)
{
   stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
    char exp[100];
    char *e, x;
    printf("Enter the expression : ");
    scanf("%s",exp);
    printf("\n");
    e = exp;

    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c ",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c ",pop());
            push(*e);
        }
        e++;
    }

    while(top != -1)
    {
        printf("%c ",pop());
    }return 0;
}
```

2a)

```
#include<stdio.h>
#define maxsize 3
int q[maxsize], front=0,rear=-1;
void insert()
{
int n;
if(rear==maxsize-1)
printf("\nQueue full\n");
```

```
else

{

printf("\nEnter the data to be added\n");

scanf("%d", &n);

q[++rear]=n;

}

}

void delete()

{

if(front>rear)

printf("\nQueue is empty\n");

else

{

printf("\n%d is deleted\n",q[front++]);

if(front>rear && rear==maxsize-1)

{

printf("\nReinit\n");

front=0; rear=-1;

}

}

}

void display()

{

int i;

if(front>rear)

printf("\nQueue is empty\n");

else

{

printf("\nQueue status is\n");

for(i=front;i<=rear;i++)

printf("%d\t",q[i]);
}
}
int main()
{
int ch;
while(1)
{
printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
puts("\nEnter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: insert(); break;
case 2:delete(); break;
case 3:display(); break;
case 4: return 0;
default :printf("\nInvalid choice\n");
}
}
}
```

2b)

```
#include <stdio.h>
#define sz 5
int count =0;
int f=0,r=-1;
int q[sz];

void push(){
        int item;
        if(count == sz){
                printf("Queue overflow");
                return;
        }
        printf("Enter the element ");
        scanf("%d",&item);
        r = (r+1)%sz;
        q[r]=item;
        count++;
}

void pop(){
        if(count == 0){
                printf("queue underflow");
                return;
        }
        printf("Item deleted is %d \n",q[f]);
        f = (f+1)%sz;
        count--;
}

void display(){
        if(count == 0){
                printf("Queue is empty \n");
                return;
        }
        int j = f;
        for(int i=0;i<count;i++){
                printf("%d \t",q[j]);
                j = (j+1)%sz;
        }
        printf("\n");
}

int main()
{
        int option;
        while(1){
                printf("1.Push 2.Pop 3.Display 4. Exit \n");
                printf("Enter the option ");
                scanf("%d",&option);
                switch(option){
                        case 1:push();
                                break;
                        case 2:pop();
                                break;
                        case 3:display();
                                break;
                        case 4:return 0;
                }
        }
        return 0;
}
```

3a)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *link;
```

```
};
typedef struct node *NODE;
NODE insertLoc(NODE first)
{
int loc,count;
NODE temp,cur;
printf("\nEnter the location\n");
scanf("%d",&loc);
temp = (NODE)malloc(sizeof(struct node));
printf("\nEnter the data\n");
scanf("%d",&temp->info);
temp->link=NULL;
if(first==NULL)
{
if(loc==1)
first = temp;
else
printf("Invalid location\n");
}
else if(loc==1)
{
temp->link=first;
first=temp;
}
else
{
cur=first;
count=1;
while(cur!=NULL)
{
if(count==loc-1)
{
temp->link=cur->link;
cur->link=temp;
break;
}
cur=cur->link;
count++;
}
if(cur==NULL)
printf("Invalid location\n");
}
return first;
}
NODE delete (NODE first)
{
NODE temp;
if (first == NULL)
{
printf ("List Empty\n");
return first;
}
temp = first;
first = first->link;
printf ("%d is deleted\n",temp->info);
free (temp);
return first;
}
void display (NODE first)
{
NODE temp;
if (first == NULL)
```

```
printf ("List is Empty\n");
else
{
printf ("Content of List\n");
temp = first;
while (temp != NULL)
{
printf ("%d\t",temp->info);
temp = temp->link;
}
printf ("\n");
}
}
int main ()
{
int ch;
NODE first = NULL;
for (;;)
{
printf ("1:INSERT 2:DELETE 3:DISPLAY 4.EXIT\n");
scanf ("%d",&ch);
switch (ch)
{
case 1: first = insertLoc (first);
break;
case 2: first = delete (first);
break;
case 3: display (first);
break;
default: exit(0);
}
}
}
```

3b)

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

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

NODE inst(NODE first,int ele){
        NODE newnode;
        newnode = (NODE)malloc(sizeof(struct node));
        newnode->info =ele;
        if(first == NULL){
                first = newnode;
                first->link = NULL;
                return first;
        }
        newnode->link = first;
        first = newnode;
        return first;
}

NODE add(NODE first,NODE second,NODE sum){
        NODE t1=first,t2 = second;
        int result,carry=0;
        while(t1 != NULL && t2 != NULL){
                result = t1->info + t2->info +carry;
                sum = inst(sum,result%10);
                carry = result/10;
                t1 = t1->link;
                t2 = t2->link;
        }
        while(t1 != NULL){
                result = t1->info +carry;
                sum = inst(sum,result);
                carry = result/10;
                t1 = t1->link;
        }
        while(t2 != NULL){
                result = t2->info + carry;
                sum = inst(sum,result);
                carry = result/10;
                t2 = t2->link;
        }

        return sum;
}

void display(NODE first){
        NODE temp = first;
        printf("Sum is ");
        while(temp != NULL){
                printf("%d",temp->info);
                temp = temp->link;
        }
        return;
```

```
}

int main(int argc, char **argv)
{
        NODE fir = NULL , sec = NULL , res = NULL;
        char a[100],b[100];
        int i;
        printf("Enter the first number \n");
        scanf("%s",a);
        for(i=0;i<strlen(a);i++){
                fir = inst(fir,a[i]-'0');
        }
        printf("Enter the Second number \n");
        scanf("%s",b);
        for(i=0;i<strlen(b);i++){
                sec = inst(sec,b[i]-'0');
        }
        res = add(fir,sec,res);
        display(res);

        return 0;
}
```

4a)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *llink;
struct node *rlink;
};
typedef struct node *NODE;
NODE first = NULL,last = NULL;
void insert (int data)
{
NODE newnode;
newnode = (NODE)malloc(sizeof(struct node));
newnode->info = data;
newnode->llink = NULL;
newnode->rlink = NULL;
if(first == NULL)
{
first=last=newnode;
return;
}
newnode->rlink = first;
first->llink = newnode;
first = newnode;
}
void delete (int key)
{
int flag =0;
NODE prev,cur,next;
if (first == NULL)
{
printf ("List Empty\n");
return;
}
if(first->rlink == NULL) // one node in the list
{
```

```
if (first->info == key)
{
printf ("%d is deleted\n",first->info);
free (first);
first=last=NULL;
return ;
}
}
if(key == first->info)
{
printf("\n%d is deleted\n",first->info);
cur = first;
first = first->rlink;
first->llink = NULL;
free(cur);
cur=NULL;
return;
}
if(key == last -> info)
{
printf("\n%d is deleted\n",last->info);
cur = last;
last = last->llink;
last->rlink = NULL;
free(cur);
cur=NULL;
return;
}
cur = first->rlink;
while(cur!=last)
{
```

```
if(cur->info==key)
{
prev = cur->llink;
next = cur->rlink;
printf("\n%d is deleted\n",cur->info);
prev->rlink = next;
next->llink = prev;
free(cur);
cur = NULL;
flag =1;
break;
}
cur=cur->rlink;
}
if(flag==0)
printf("\nKey not found\n");
}
void display ()
{
NODE temp;
if (first == NULL)
printf ("List is Empty\n");
else
{
printf ("Content of List\n");
temp = first;
while (temp != NULL)
{
printf ("%d\t",temp->info);
temp = temp->rlink;
}
```

```
printf ("\n");
}
}
int main ()
{
int ch,data;
for (;;)
{
printf ("1:INSERT 2:DELETE 3:DISPLAY 4:EXIT\n");
scanf ("%d",&ch);
switch (ch)
{
case 1: printf ("Enter the data\n");
scanf ("%d",&data);
insert (data);
break;
case 2: printf ("Enter the data to be deleted\n");
scanf ("%d",&data);
delete (data);
break;
case 3: display ();
break;
default: exit(0);
}
}
}
```

4b)

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

```
struct node{
	int info;
	struct node *left;
	struct node *right;
}; typedef struct node *NODE;

NODE inst(NODE first,int ele){
	NODE newnode,temp;
	newnode =(NODE)malloc(sizeof(struct node));
	newnode->info = ele;
	if(first == NULL){
		newnode->left = newnode->right = NULL;
		first = newnode;
		return first;
	}
	newnode->right=NULL;
	temp = first;
	while(temp->right != NULL){
		temp = temp->right;
	}

	temp->right = newnode;
	newnode->left = temp;
	return first;
}

NODE lremove(NODE first,NODE second){
	NODE temp = first;
	while(temp != NULL){
		if(temp->info != 0){
```

```
                        second = inst(second,temp->info);
                }
                temp = temp->right;
        }
        return second;
}
void display(NODE first){
        NODE temp = first;
        while(temp!=NULL){
                printf("%d \t",temp->info);
                temp = temp->right;
        }
        printf("\n");
}

int main()
{
        NODE first = NULL,second= NULL;
        int m,n,ele;
        printf("Enter the size of matrix \n");
        scanf("%d %d",&m,&n);
        printf("Enter the elements of matrix \n");
        for(int i=1;i<=m;i++){
                for(int j=1;j<=n;j++){
                        scanf("%d",&ele);
                        first = inst(first,ele);
                }
        }
        printf("COntents of matrix are \n");
        display(first);
        second = lremove(first,second);
```

```
	printf("The list after removing zeros \n");
	display(second);
	return 0;
}
```

5a)

```
#include <stdio.h>
#include <stdlib.h>
#define sz 100
void bt (int a[sz],int ele)
{
int c,p,i;
if (a[0] == '\0')
{
a[0] = ele;
return;
}
c = 0;
while (a[c] != '\0')
{
p = c;
if (ele < a[c])
c = 2*c+1;
else
c = 2*c+2;
}
if (ele < a[p])
c = 2*p+1;
else
c = 2*p+2;
a[c] = ele;
printf ("Constructed Binary Tree is \n");
for (i=0;i<sz;i++)
if (a[i] != '\0')
printf ("a[%d]==>>%d\n",i,a[i]);
}
int main ()
{
int n,a[sz],i,ele;
for (i=0;i<sz;i++)
a[i] = '\0';
printf ("Enter the no of Data to Binary Tree\n");
scanf ("%d",&n);
printf ("Enter the Data to Binary Tree\n");
for (i=0;i<n;i++)
{
scanf ("%d",&ele);
bt (a,ele);
}
}
```

5b)

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
// Structure definition
struct node
{
   char info;
   struct node *left;
   struct node *right;
};
typedef struct node *NODE;
// Structure definition
struct stack
{
   int top;
   NODE data[10];
};
typedef struct stack STACK;
// Function for Precedence
int preced(char item)
{
   switch(item)
   {
          case '^': return 5;
          case '*':
          case '/': return 3;
          case '+':
          case '-': return 1;
   }
}
//Function to Display Tree in Preorder
void preorder(NODE root)
{
   if(root != NULL)
   {
          printf("%c\t", root->info);
          preorder(root->left);
          preorder(root->right);
   }
```

```
}
//Function to Display Tree in Inorder
void inorder(NODE root)
{
   if(root != NULL)
   {
          inorder(root->left);
          printf("%c\t", root->info);
          inorder(root->right);
   }
}
//Function to Display Tree in Postorder
void postorder(NODE root)
{
   if(root != NULL)
   {
          postorder(root->left);
          postorder(root->right);
          printf("%c\t", root->info);
   }
}
//Function to Push Item into Stack
void push(STACK *s, NODE temp)
{
   s->data[++(s->top)] = temp;
}
//Function to Pop Item from Stack
NODE pop(STACK *s)
{
   return (s->data[(s->top)--]);
}
```

```
//Function create Node
NODE createnode(char item)
{
   NODE temp;
   temp = (NODE)malloc(sizeof(struct node));
   temp->info = item;
   temp->left = NULL;
   temp->right = NULL;
   return temp;
}
//Function to create Expression Tree
NODE createExpTree(char expr[20])
{
char symbol;
int i;
           NODE temp, t, l, r;
            STACK tree, operator;
           tree.top = -1;
           operator.top = -1;

   for (i=0; expr[i] != '\0'; i++)
   {
           symbol = expr[i];
           temp = createnode(symbol);
           if(isalnum(symbol))
              push(&tree, temp);
           else{
              if(operator.top == -1)
                     push(&operator, temp);
              else{
```

```
                        while(operator.top != -1 && preced((operator.data[operator.top])->info) >=
preced(symbol))
                        {
                            t = pop(&operator);
                            r = pop(&tree);
                            l = pop(&tree);
                            t->right = r;
                            t->left = l;
                            push(&tree, t);
                        }
                        push(&operator, temp);
                }
            }
    }
    while(operator.top != -1){
      t = pop(&operator);
      r = pop(&tree);
      l = pop(&tree);
      t->right = r;
      t->left = l;
      push(&tree, t);
    }
  return pop(&tree);
}

//Main Program
int main()
{
    NODE root = NULL;
    char expr[20];
    //clrscr();
    printf("Read expression\n");
    scanf("%s", expr);
    root = createExpTree(expr);
    printf("\nInorder::");
    inorder(root);
    printf("\nPreorder:");
    preorder(root);
    printf("\nPostorder:");
    postorder(root);
    return 0;
}
```

6a)

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
		int info;
		struct node * left;
		struct node * right;
		}; typedef struct node * NODE;
NODE create(NODE root)
{
		NODE newnode,pres,prev;
		newnode=(NODE)malloc(sizeof(struct node));
		printf("Enter the info \n");
		scanf("%d",&newnode->info);
		newnode->left=newnode->right=NULL;
```

```
        if(root==NULL)
        {
                root=newnode;
                return(root);
                }
                pres=root;
                while(pres!=NULL)
                {
                        if(newnode->info < pres->info)
                        {
                                prev=pres;
                                pres=pres->left;
                                }
                                else
                                {
                                        prev=pres;
                                        pres=pres->right;
                                        }
                        }
                        if(newnode->info < prev->info)
                                prev->left=newnode;
                                else
                                        prev->right=newnode;
                                        return (root);
        }
void preorder(NODE root)
{
        if(root!=NULL)
        {
                printf("%d \t \n",root->info);
                preorder(root->left);
                preorder(root->right);
                }
        }
void postorder(NODE root)
{
        if(root!=NULL)
        {
                postorder(root->left);
                postorder(root->right);
                printf("%d \t \n",root->info);
                }
        }
void inorder(NODE root)
{
        if(root!=NULL)
        {
                inorder(root->left);
                printf("%d \t \n",root->info);
                inorder(root->right);

                }
        }
void main()
{
        NODE root=NULL;
        int ch;
        for(;;)
        {
                printf("1.INSERT 2.PREORDER 3.POSTORDER 4.INORDER \n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: root=create(root);
                        break;
                        case 2: if(root==NULL)
                        printf("Tree is empty \n");
                        else
                        {
                                preorder(root);
                                break;
                                }
                        case 3: if(root==NULL)
                        printf("Tree is empty \n");
                        else
                        {
                                preorder(root);
                                break;
                                }
                        case 4: if(root==NULL)
                        printf("Tree is empty \n");
                        else
                        {
                                inorder(root);
                                break;
                                }
                        default: exit(0);
                        }
                }
        }
```

6b)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
int eid;
char ename [10];
float lt;
struct node *left;
struct node *right;
};
typedef struct node *NODE;
NODE insert (NODE root,int eid,char ename [10],float lt)
{
NODE newnode,prev,pres;
newnode = (NODE)malloc(sizeof(struct node));
newnode->left = newnode->right = NULL;
newnode->eid = eid;
strcpy (newnode->ename,ename);
newnode->lt = lt;
if (root == NULL)
{
root = newnode;
return root;
}
pres = root;
while (pres != NULL)
{
prev = pres;
if (eid < pres->eid)
pres = pres->left;
else if (eid > pres->eid)
pres = pres->right;
else
{
printf ("Duplicate\n");
return root;
}
}
if (eid < prev->eid)
prev->left = newnode;
else
prev->right = newnode;
return root;
}
void inorder (NODE root)
{
if (root != NULL)
{
inorder (root->left);
printf
("%d\t%s\t\t%.2f\n",root->eid,root->ename,root->lt);
inorder (root->right);
}
}
int main ()
{
NODE root = NULL;
int ch,eid;
char ename [10];
float lt;
for (;;)
```

```
{
printf ("1:INSERT 2:INORDER\n");
scanf ("%d",&ch);
switch (ch)
{
case 1: printf ("Enter Employee Details:\n");
printf ("Employee ID\n");
scanf ("%d",&eid);
printf ("Employee Name\n");
scanf ("%s",ename);
printf ("Login Time\n");
scanf ("%f",&lt);
root = insert(root, eid, ename, lt);
break;
case 2: if (root == NULL)
printf ("Employee Details Absent\n");
else
{
printf("Eid\tEname\tLT\n");
inorder(root);
}
break;
default: exit(0);
}
}
}
```