

1 a)

```
#include<stdio.h>

#include<stdlib.h>

#define sz 5

int a[sz];

int top=-1;

void push()
{
    int ele;
    if(top>=sz-1)
    {
        printf("Stack overflow \n");
    }
    else
    {
        printf("Enter the value to be pushed - \n");
        scanf("%d", &ele);
        top=top+1;
        a[top]=ele;
    }
}

void pop()
{
    if(top<=-1)
        printf("Stack underflow \n");
    else
    {
        printf("The item deleted is %d \n",a[top]);
        top=top-1;
    }
}
```

```

    }
void display()
{
    int i;
    if(top<=-1)
        printf("Stack is empty \n");
    else
    {
        printf("Elements in the stack are - \n");
        for(i=top;i>=0;i--)
        {
            printf("%d \n",a[i]);
        }
    }
}

int main()
{
    int ch;
    for(;;)
    {
        printf("1.PUSH 2.POP 3.DISPLAY \n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: push();
                    break;
            case 2: pop();
                    break;
            case 3: display();
                    break;
            default: exit(0);
        }
    }
}

```

```

        }
    }
}

```

1 b)

```

#include<stdio.h>

#include<string.h>

char pop(char s[10],int *top)
{
    char item;

    item=s[*top];

    *top=*top-1;

    return item;
}

void push(char s[10],int *top,char item)
{
    *top=*top+1;

    s[*top]=item;
}

int IPV(char ch)
{
    if(ch=='(') return 9;

    if(ch==')') return 0;

    if(ch=='+' | ch=='-') return 1;

    if(ch=='*' | ch=='/') return 3;

    return 7;
}

int SPV(char ch)
{
    if(ch=='(') return 0;

```

```
if(ch=='#') return -1;
if(ch=='+' | ch=='-') return 2;
if(ch=='*' | ch=='/') return 4;
return 8;
}
```

```
void main()
```

```
{
    char in[20],post[20],s[20];
    int top=-1;
    int i,j=0;
    printf("Enter infix expression - \n");
    scanf("%s",in);
    push(s,&top,'#');
    for(i=0;i<strlen(in);i++)
    {
        while(IPV(in[i])<SPV(s[top]))
            post[j++]=pop(s,&top);
        if(IPV(in[i])>SPV(s[top]))
            push(s,&top,in[i]);
        else
            pop(s,&top);
    }
    while(s[top]!='#')
        post[j++]=pop(s,&top);
    post[j]='\0';
    printf("Postfix form %s \n",post);
}
```

2 a)

```
#include<stdio.h>

#include<stdlib.h>

#define sz 5

int f=0;

int r=-1;

int a[sz];

void insert()
{
    int ele;

    if(r==sz-1)
    {
        printf("Q overflow \n");
    }

    else
    {
        printf("Enter the number to be inserted - \n");

        scanf("%d", &ele);

        r=r+1;

        a[r]=ele;
    }
}

void delete()
{
    if(f>r)
    {
        printf("Q underflow");
    }
}
```

```

        else
        {
            printf("The item deleted is %d \n",a[f]);
            f=f+1;
        }
    }
}

void display()
{
    int i;
    if(f>r)
    {
        printf("Q is emepty \n");
    }
    else
    {
        printf("Content of Q are \n");
        for(i=f;i<=r;i++)
        {
            printf("%d \n", a[i]);
        }
    }
}

int main()
{
    int choice;
    for(;;)
    {
        printf("1.INSERT 2.DELETE 3.DISPLAY \n");
        scanf("%d",&choice);
        switch(choice)
        {

```

```

        case 1: insert();

        break;

        case 2: delete();

        break;

        case 3: display();

        break;

        default: exit(0);

    }

}

}

```

2 b)

```

#include<stdio.h>

#include<stdlib.h>

#define sz 5

int f=0;

int r=-1;

int a[sz];

int count=0;

void insert()

{

    int ele;

    if(count==sz)

    {

        printf("CQ overflow \n");

    }

    else

    {

```

```

        printf("Enter the number to be inserted - \n");

        scanf("%d", &ele);

        r=(r+1)%sz;

        a[r]=ele;

        count=count+1;

    }

}

void delete()
{
    if(count==0)
    {
        printf("CQ underflow");
    }

    else
    {
        printf("The item deleted is %d \n",a[f]);

        f=(f+1)%sz;

        count=count-1;
    }
}

void display()
{
    int i,j;

    if(count==0)
    {
        printf("CQ is emepty \n");
    }

    else
    {
        printf("Content of CQ are \n");

        j=f;

```

```

        for(i=1;i<=count;i++)
        {
            printf("%d \n", a[j]);
            j=(j+1)%sz;
        }
    }

int main()
{
    int choice;
    for(;;)
    {
        printf("1.INSERT 2.DELETE 3.DISPLAY \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: insert();
                    break;
            case 2: delete();
                    break;
            case 3: display();
                    break;
            default: exit(0);
        }
    }
}

```

3 a)

```
#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node
{
    int info;

    struct node *link;

}; typedef struct node * NODE;

NODE inspos(NODE first)
{
    NODE newnode,prev,pres;

    int pos,count;

    printf("Pos of node to be inserted \n");

    scanf("%d",&pos);

    if(pos==1)
    {
        newnode=(NODE)malloc(sizeof(struct node));

        printf("Enter the element to be inserted - \n");

        scanf("%d",&newnode->info);

        newnode->link=first;

        first=newnode;

        return first;

    }

    else{

        count=1;

        pres=first;

        while(count<=(pos-1)&&pres!=NULL)
```

```

    {
        prev=pres;
        pres=pres->link;
        count=count+1;
    }
}
if(pres==NULL)
{
    printf("Invalid pos \n");
    return first;
}
else
{
    newnode=(NODE)malloc(sizeof(struct node));
    printf("Enter the item \n");
    scanf("%d",&newnode->info);
    prev->link=newnode;
    newnode->link=pres;
    return first;
}
}
NODE delinfo(NODE first)
{
    NODE temp;
    temp=first;
    printf("%d is deleted \n",temp->info);
    first=first->link;
    free(temp);
    return first;
}
void display(NODE first)

```

```

{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty \n");
    }
    else
    {
        temp=first;
        printf("Contents of the list are - \n");
        while(temp!=NULL)
        {
            printf("%d \n",temp->info);
            temp=temp->link;
        }
    }
}

void main()
{
    NODE first=NULL;
    int ch;
    for(;;)
    {
        printf("1.INSERT 2.DELETE 3.DISPLAY \n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:first=inspos(first);
            break;
            case 2: first=delinfo(first);
            break;

```

```

        case 3: display(first);

        break;

        default:exit(0);

    }

}

}

```

3 b)

```

#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<string.h>
struct node
{
    int info;
    struct node * link;
}; typedef struct node * NODE;
NODE insfront(NODE first, int item)
{
    NODE newnode;
    newnode=(NODE)malloc(sizeof(struct node));
    newnode->info=item;
    newnode->link=first;
    first=newnode;
    return first;
}
NODE addnum(NODE first, NODE second, NODE result)

```

```

{
    NODE temp1,temp2;
    int sum;
    int carry=0;
    temp1=first;
    temp2=second;
    while(temp1!=NULL && temp2!=NULL)
    {
        sum=temp1->info+temp2->info+carry;
        result=insfront(result,sum%10);
        carry=sum/10;
        temp1=temp1->link;
        temp2=temp2->link;
    }
    while(temp1!=NULL)
    {
        sum=temp1->info+carry;
        result=insfront(result,sum%10);
        carry=sum/10;
        temp1=temp1->link;
    }
    while(temp2!=NULL)
    {
        sum=temp2->info+carry;
        result=insfront(result,sum%10);
        carry=sum/10;
        temp2=temp2->link;
    }
    if(carry==1)
        result=insfront(result,sum%10);
    return result;
}

```

```

    }

void display(NODE result)
{
    NODE temp=result;
    while(temp!=NULL)
    {
        printf("%d",temp->info);
        temp=temp->link;
    }
}

void main()
{
    NODE first=NULL, second=NULL, result=NULL;
    char a[100],b[100];
    int i;
    printf("Enter the first number - \n");
    scanf("%s",a);
    for(i=0;i<strlen(a);i++)
    {
        first=insfront(first,a[i]-48);
    }

    printf("Enter the second number - \n");
    scanf("%s",b);
    for(i=0;i<strlen(b);i++)
        second=insfront(second,b[i]-'0');
    result=addnum(first,second,result);
    printf("The sum is \n");
    display(result);
}

```

4 a)

```
#include<stdio.h>

#include<stdlib.h>

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

NODE insfront(NODE first)
{
    NODE newnode;
    int ele;
    newnode=(NODE)malloc(sizeof(struct node));
    printf("Enter the element to be inserted - \n");
    scanf("%d",&ele);
    newnode->info=ele;
    newnode->left=NULL;
    newnode->right=first;
    first->left=newnode;
    first=newnode;
    return first;
}

void display(NODE first)
{
    NODE temp;
    if(first==NULL)
        printf("List is empty \n");
    else
```

```

    {
        temp=first;
        while(temp!=NULL)
        {
            printf("%d \t", temp->info);
            temp=temp->right;
        }
    }
}

NODE delinfo(NODE first)
{
    int key;
    NODE pres,prev;
    if(first==NULL)
    {
        printf("List is empty \n");
        return first;
    }

    else
    {
        printf("Enter the info to be deleted - \n");
        scanf("%d \n",&key);
        if(key==first->info)
        {
            pres=first;
            first=first->right;
            free(pres);
            return first;
        }

        pres=first;
        while(pres!=NULL && key!=pres->info)

```

```

        {
            prev=pres;
            pres=pres->right;
        }
        if(pres==NULL)
        {
            printf("Key not found \n");
            return first;
        }
        prev->right=pres->right;
        pres->right->left=prev;
        free(pres);
        return first;
    }
}

void main()
{
    NODE first=NULL;
    int ch;
    for(;;)
    {
        printf("1.INSERT 2.DELETE 3.DISPLAY \n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: first=insfront(first);
                    break;
            case 2: first=delinfo(first);
                    break;
            case 3: display(first);
                    break;
        }
    }
}

```

```

        default:exit(0);
    }
}
}

```

4 b)

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

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

```
struct node
```

```

{
    int row, col, val;
    struct node * left;
    struct node * right;
};

```

```
typedef struct node * node;
```

```
node root = NULL;
```

```
int m,n;
```

```
void insrear(int row, int col, int data)
```

```

{
    node temp, pres;
    temp = (node)malloc(sizeof(node));
    temp->val = data;
    temp->row = row;
    temp->col = col;
    temp->left = NULL;
    temp->right = NULL;

    if(root == NULL)
        root= temp;
}

```

```

        else
        {
            pres = root;
            while(pres->right != NULL)
                pres = pres->right;

            pres->right = temp;
            temp-> left =pres;
        }
    }
}

```

```

void display_list()

```

```

{
    node temp;
    if(root==NULL)
        printf("List is empty\n");
    else
    {
        temp= root;
        printf("ROW\tCOL\tValue\n");
        while(temp!= NULL)
        {
            printf("%d\t%d\t%d\n",temp->row,temp->col,temp->val);
            temp=temp->right;
        }
    }
}

```

```

void display_matrix()

```

```

{
    int i,j;

```

```

        node temp;

        if(root ==NULL)
printf("List is empty\n");
else
{
    temp= root;
    for(i=1;i<=m;i++)
    {
        for(j=1;j<=n;j++)
        {
            if(temp != NULL && temp-> row == i && temp->col==j)
            {
                printf("%d\t",temp->val);
                temp= temp->right;
            }
            else
                printf("0\t");
        }
        printf("\n");
    }
}
}

```

```

int main()
{
    int i,j,a;
    printf("Enter rows and columns\n");
    scanf("%d %d",&m,&n);
    printf("Enter the elements\n");
    for(i=1;i<=m;i++)
    {

```

```

        for(j=1;j<=n;j++)
        {
                scanf("%d",&a);
                if(a!=0)
                insrear(i,j,a);
        }
}

printf("List content display\n");
display_list();
printf("Matrix display from list\n");
display_matrix();
}

```

6 a)

```

#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;
        NODE 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",root->info);
            preorder(root->left);
            preorder(root->right);
        }
    }

void inorder(NODE root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%d \t",root->info);
        inorder(root->right);
    }
}

void postorder(NODE root)
{
    if(root!=NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d \t",root->info);
    }
}

void main()
{
    NODE root=NULL;
    int ch;
    for(;;)
    {

```

```

printf("1.INSERT 2.PREORDER 3.INORDER 4.POSTORDER \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
        inorder(root);
        break;
    case 4: if(root==NULL)
        printf("Tree is empty \n");
        else
        postorder(root);
        break;
}
}
}

```

6 b)

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node

```

```

{
    int eid, elogin;
    char ename[20];
    struct node * right;
    struct node * left;
    };typedef struct node * NODE;
NODE create(NODE root)
{
    NODE newnode;
    NODE pres, prev;
    newnode=(NODE)malloc(sizeof(struct node));
    printf("Enter the employee ID - \n");
    scanf("%d",&newnode->eid);
    printf("Enter the employee name - \n");
    scanf("%s",newnode->ename);
    printf("Enter the employee login time - \n");
    scanf("%d",&newnode->elogin);
    newnode->left=newnode->right=NULL;
    if(root==NULL)
    {
        root=newnode;
        return root;
    }
    pres=root;
    while(pres!=NULL)
    {
        if(newnode->eid==pres->eid)
        {
            printf("Duplicate identifiers not allowed \n");
            return root;
        }
    }
}

```

```

prev=pres;
if(newnode->eid < pres->eid)
{
    prev=pres;
    pres=pres->left;
}
else
{
    prev=pres;
    pres=pres->right;
}
}
if(newnode->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 %d\n",root->eid,root->ename,root->elgin);
        inorder(root->right);
    }
}

```

```

void main()

```

```

{
    NODE root=NULL;
    int ch,n,i;

```

```

printf("1.CREATE RECORD 2.DISPLAY \n");
for(;;)
{
    printf("Enter your choice - \n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: printf("Enter the no of employees - \n");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                root=create(root);
                break;
        case 2: if(root==NULL)
                printf("Tree is empty \n");
                else
                {
                    printf("Final report - \n");
                    printf("EmpID \t EmpName \t EmpLogin \n");
                    inorder(root);
                    break;
                }
        default: exit(0);
    }
}
}

```