NAME – Arka Sinha

USN – 1BM19CS024

SUBJECT – Data Structures Lab Report

ACADEMIC YEAR – 2020-2021

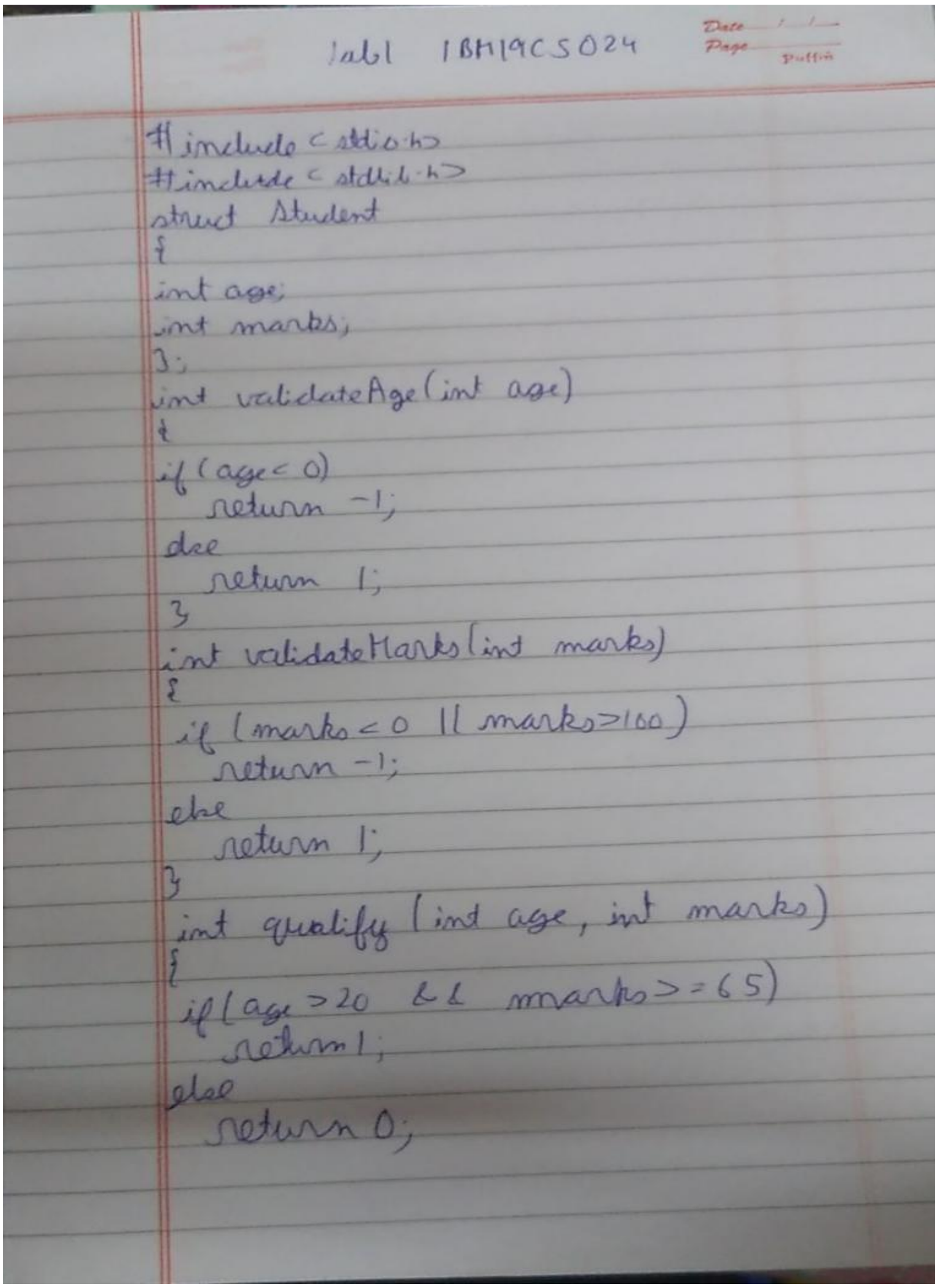
LAB PROGRAM 1

Source code-

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

struct Student
{
    int age;
    int marks;
};
int validateAge(int age)
{
    if(age<0)
        return -1;
    else
        return 1;
}
int validateMarks(int marks)
{
    if(marks<0 || marks>100)
        return -1;
    else
        return 1;
}
int qualify(int age,int marks)
{
    if(age>20 && marks>=65)
        return 1;
    else
        return 0;
}
int main()
{
    struct Student s;
    int checkAge,checkMarks,qualified;
    printf("Enter age and marks:\n");
    scanf("%d%d",&s.age,&s.marks);
    checkAge=validateMarks(s.age);
    checkMarks=validateMarks(s.marks);
    if(checkAge==1 && checkMarks==1){
        qualified=qualify(s.age,s.marks);
        if (qualified==1)
            printf("Qualified!!");
        else
```

```
                printf("Not Qualified");
        }
        else
            printf("Invalid age or marks");
        return 0;
}
```

Writeup-

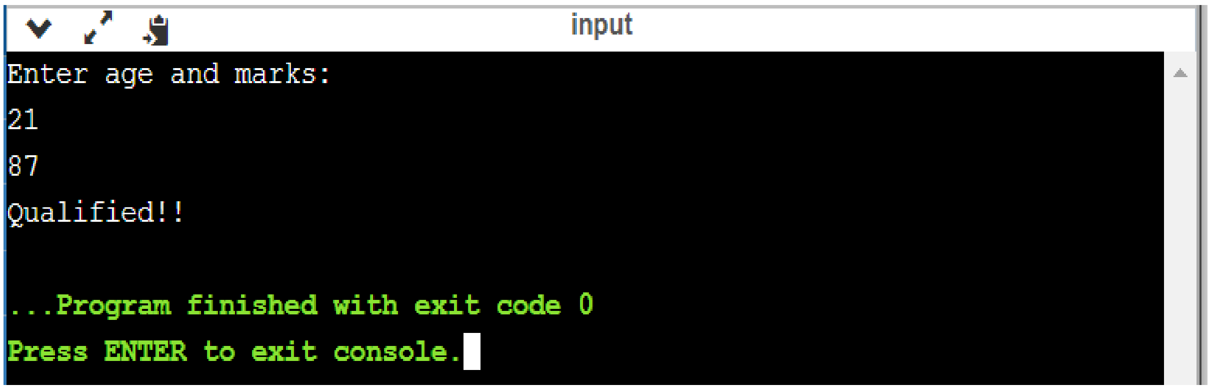


lab1 1BM19CS024

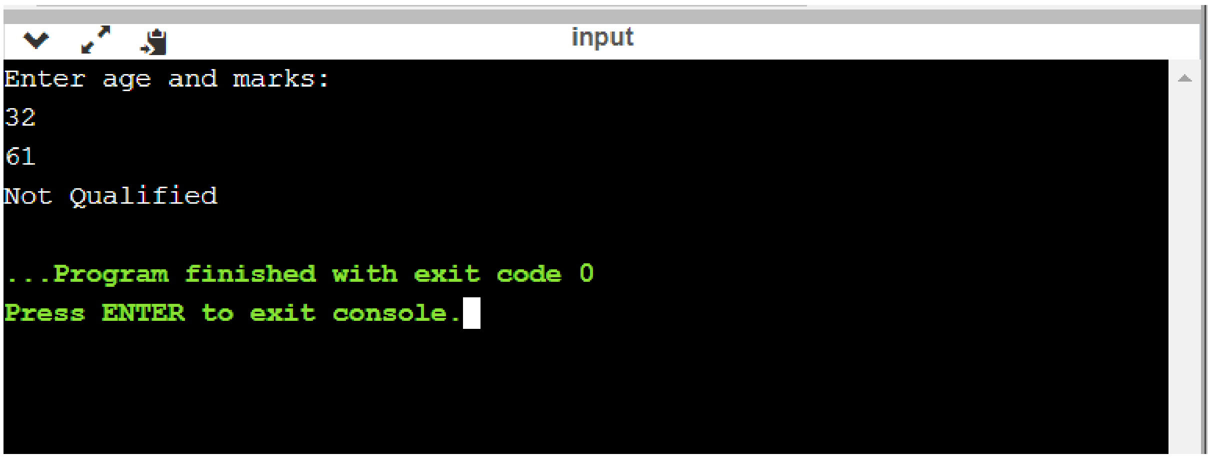
Date / /
Page

```
#include <stdio.h>
#include <stdlib.h>
struct Student
{
int age;
int marks;
};
int validateAge(int age)
{
if(age< 0)
  return -1;
else
  return 1;
}
int validateMarks(int marks)
{
 if (marks< 0 || marks>100)
  return -1;
else
  return 1;
}
int qualify (int age, int marks)
{
 if(age >20  && marks>=65)
   return 1;
else
 return 0;
```

ScreenShot-



```
Enter age and marks:
21
87
Qualified!!

...Program finished with exit code 0
Press ENTER to exit console.
```



```
Enter age and marks:
32
61
Not Qualified

...Program finished with exit code 0
Press ENTER to exit console.
```

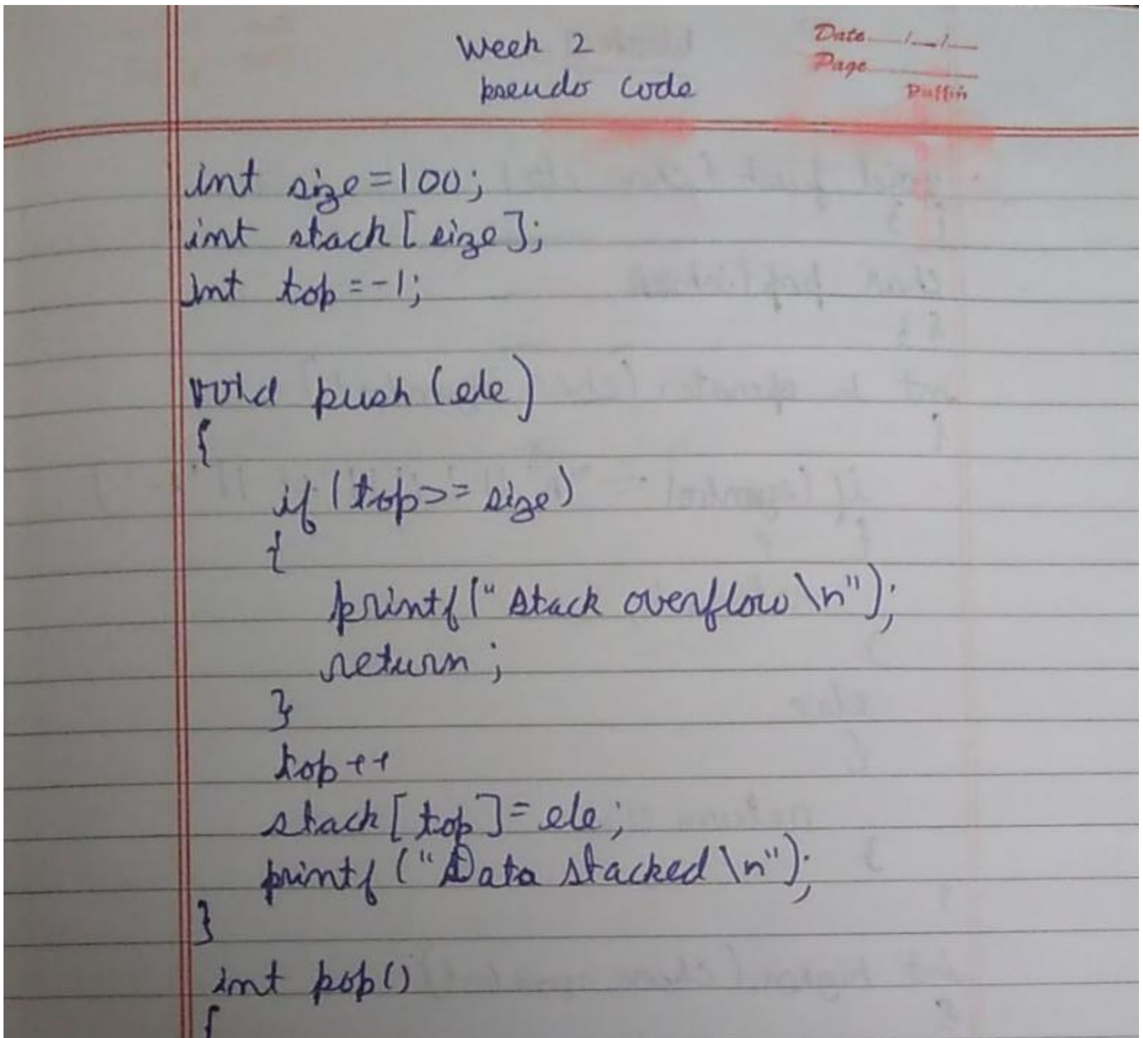
LAB PROGRAM 2

Source code-

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

                    #define size 3
```

```
int stack[size];
int top=-1;


int main()
{
    int c,n,a=1;
    while(a==1)
    {
        printf("\n1-Push\n2-Pull\n3-Display\n");
        printf("Enter your choice: ");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                printf("Enter number to push into stack: ");
                scanf("%d",&n);
                push(n);
                break;
            case 2:
                n=pop();
                if(n!=INT_MIN)
                    printf("Number popped is %d\n",n);
                break;
            case 3:
                display();
                break;
            default:
                printf("Enter valid choice\n");
        }
        printf("\nPress 1 to continue: ");
        scanf("%d",&a);
    }
    return 0;
}

void push(int ele)
{
    if(top>=size)
    {
        printf("Stack overflow\n");
        return;
    }
    top++;
    stack[top]=ele;
    printf("Number stacked\n");
}

int pop()
{
    if(top<0)
    {
        printf("Stack empty\n");
        return INT_MIN;
    }
    return stack[top--];
}

void display()
{
    if(top==-1)
    {
        printf("Stack is empty\n");
    }
    int i;
    for(i=top;i>=0;i--)
        printf("%d\n",stack[i]);
}
```

Writeup-

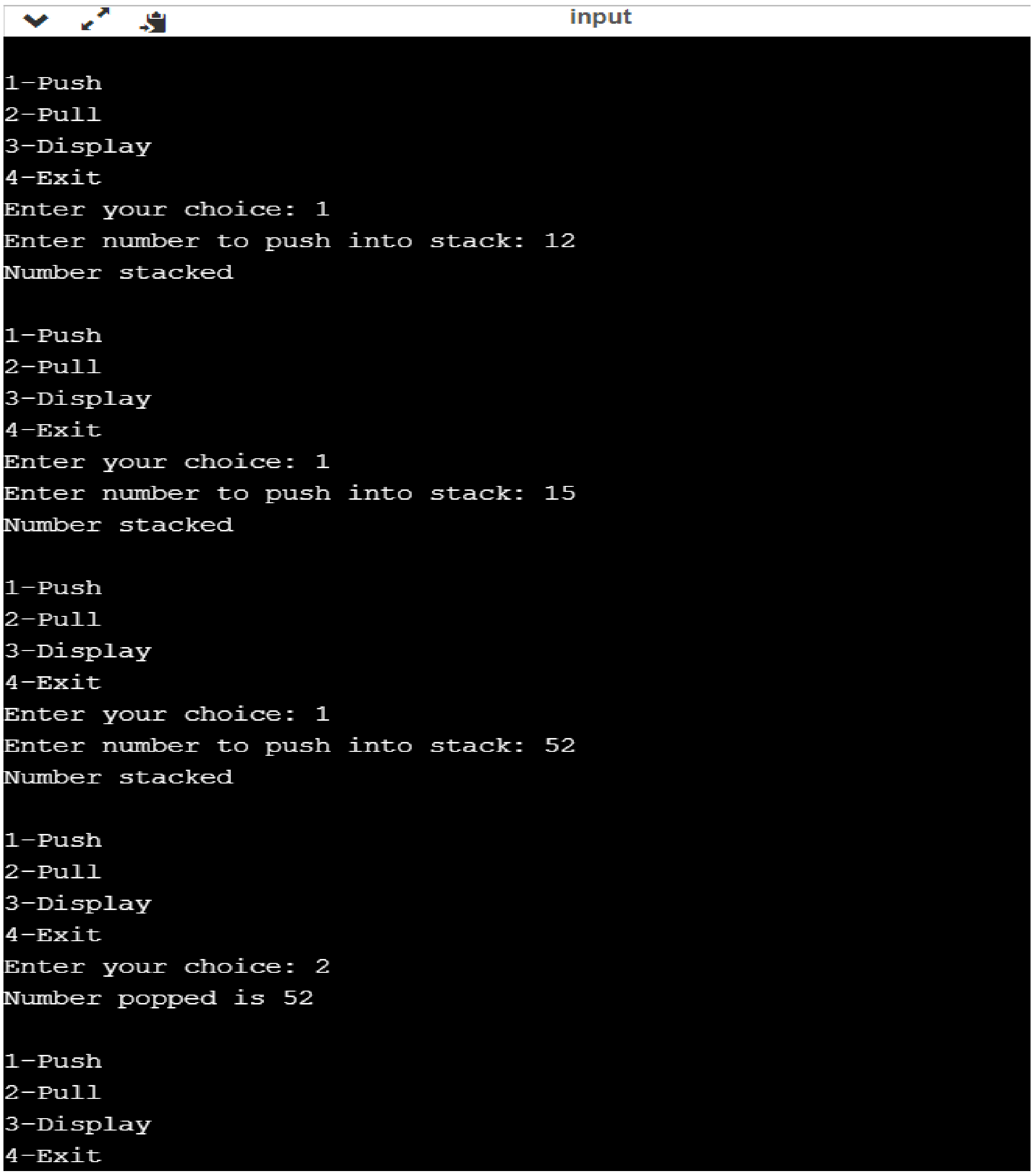


Week 2
pseudo code

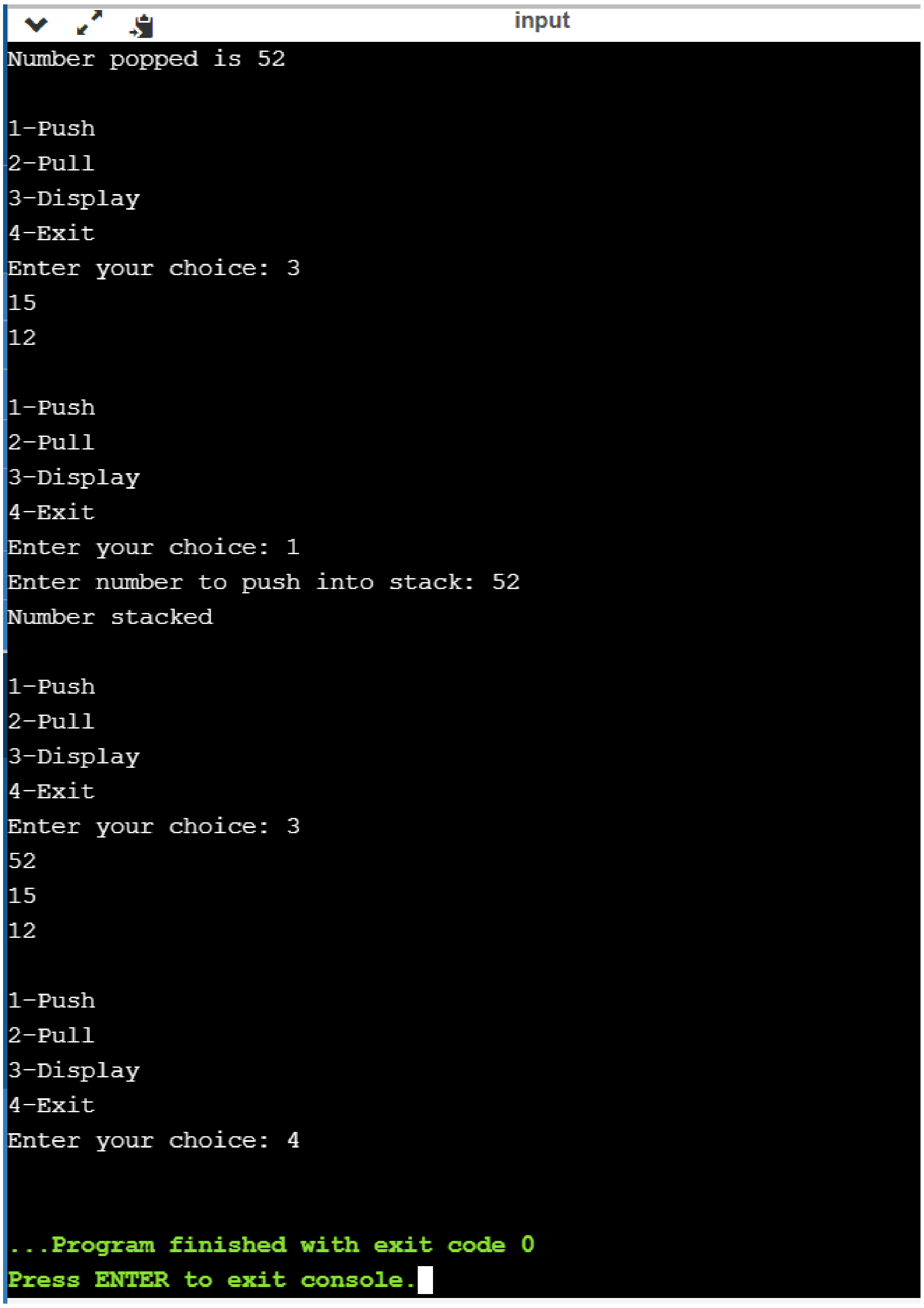
Date __/__/__
Page ______
Puffin

```
int size=100;
int stack[size];
int top=-1;

void push(ele)
{
    if(top>=size)
    {
        printf("Stack overflow\n");
        return;
    }
    top++
    stack[top]=ele;
    printf("Data stacked\n");
}
int pop()
{
    if(top<0)
    {
        printf("Stack empty\n");
        return INT_MIN;
    }
    return stack[top--]
}
void display(int i)
{
    for(i=0; i<top+1; i++)
        printf("%d",stack[i]);
}
```

Screen Shot-



```
1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 1
Enter number to push into stack: 12
Number stacked

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 1
Enter number to push into stack: 15
Number stacked

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 1
Enter number to push into stack: 52
Number stacked

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 2
Number popped is 52

1-Push
2-Pull
3-Display
4-Exit
```



```
Number popped is 52

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 3
15
12

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 1
Enter number to push into stack: 52
Number stacked

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 3
52
15
12

1-Push
2-Pull
3-Display
4-Exit
Enter your choice: 4


...Program finished with exit code 0
Press ENTER to exit console.
```

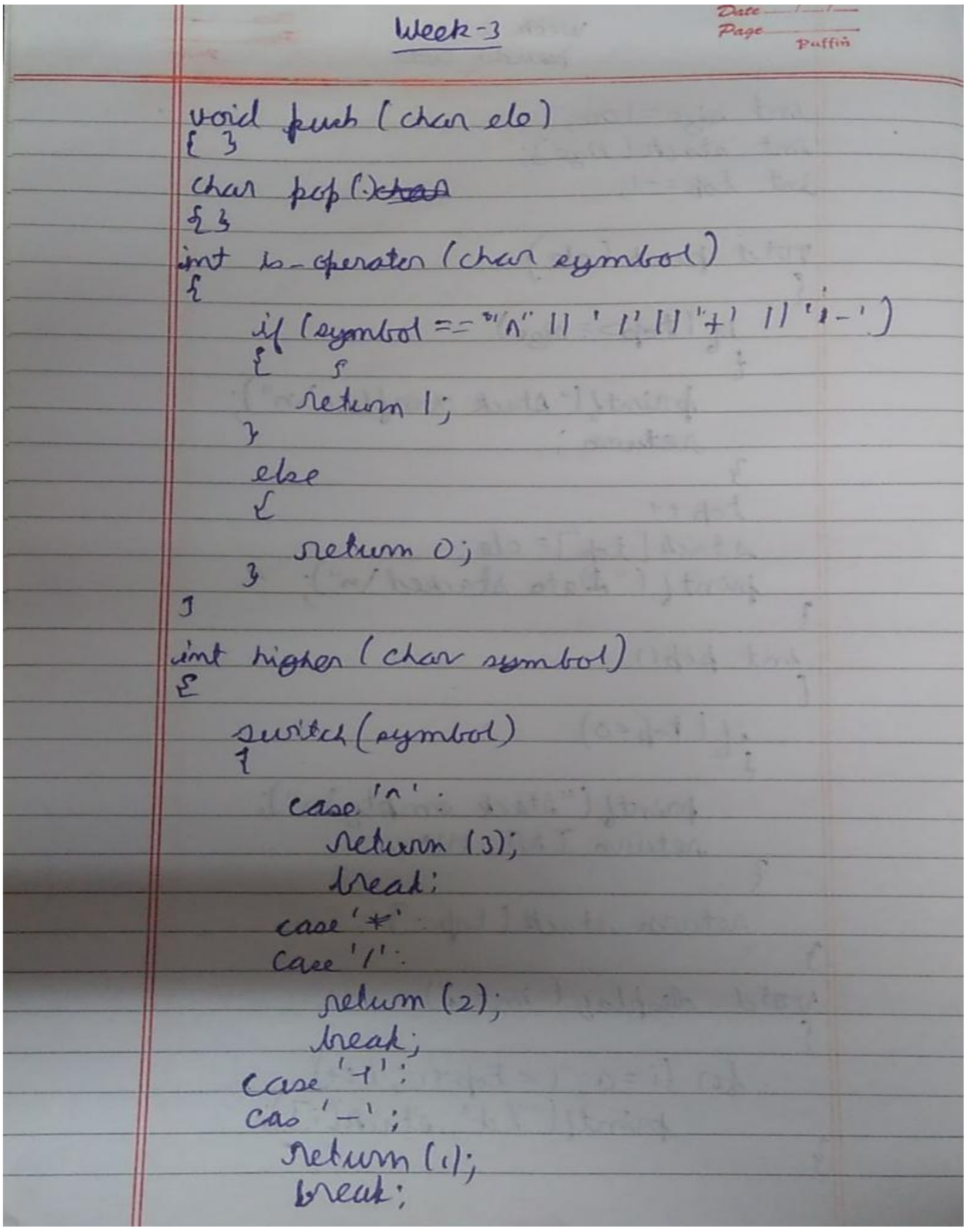
Lab Program 3

Source Code-

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

                    #define SIZE 20

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

                    void push(char ele)
                    {
                            if(top >= SIZE)
                            {
                                    printf("\nStack Overflow.");
                            }
                            else
                            {
                                    top = top+1;
                                    stack[top] = ele;
                            }
                    }
                    char pop()
                    {
                            char ele ;
                            if(top==-1)
                            {
                                    printf("stack under flow: invalid infix expression");
                                    getchar();
                                    exit(1);
                            }
                            else
                            {
                                    ele = stack[top];
                                    top = top-1;
                                    return(ele);
                            }
                    }
                    int is_operator(char symbol)
                    {
                            if(symbol == '^' || symbol == '*' || symbol == '/' || symbol
                    == '+' || symbol =='-')
                            {
```

```
               return 1;
        }
        else
        {
         return 0;
        }
}
int higher(char symbol)
{
    switch(symbol)
    {
        case '^':
            return(3);
            break;
        case '*':
        case '/':
            return(2);
            break;
        case '+':
        case '-':
            return(1);
            break;
        default:
            return(0);
            break;
    }
}
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
        int i=0, j=0;
        char ele;
        char x;
        push('(');
        strcat(infix_exp,")");
        ele=infix_exp[i];
        while(ele != '\0')
        {
               if(ele == '(')
               {
                      push(ele);
               }
               else if(ele=='A' || ele=='B' || ele=='C' || ele=='D'
|| ele=='E' || ele=='F' || ele=='G' || ele=='H' || ele=='I' ||
ele=='J' || ele=='K' || ele=='L' || ele=='M' || ele=='N' || ele=='O'
|| ele=='P' || ele=='Q' || ele=='R' || ele=='S' || ele=='T' ||
ele=='U' || ele=='V' || ele=='W' || ele=='X' || ele=='Y' || ele=='Z'
|| ele=='0' || ele=='1' || ele=='2' || ele=='3' || ele=='4' ||
ele=='5' || ele=='6' || ele=='7' || ele=='8' || ele=='9')
               {
                       postfix_exp[j] = ele;
                       j++;
               }
               else if(is_operator(ele) == 1)
               {
                       x=pop();
                       while(is_operator(x) == 1 && higher(x)>=
higher(ele))
                       {
                               postfix_exp[j] = x;
                               j++;
                               x = pop();
                       }
                       push(x);
                       push(ele);
               }
               else if(ele == ')')
               {
                       x = pop();
                       while(x != '(')
                       {
                               postfix_exp[j] = x;
                               j++;
                               x = pop();
                       }
               }
               else
               {
                       printf("\nInvalid infix Expression.\n");
                       getchar();
                       exit(1);
               }
               i++;
        ele = infix_exp[i];
        }
        postfix_exp[j] = '\0';
}
int main()
{
        char infix[SIZE], postfix[SIZE];
        printf("\nEnter Infix expression : ");
        gets(infix);
        InfixToPostfix(infix,postfix);
        printf("Postfix Expression is: ");
        puts(postfix);
        return 0;
}
```

Writeup-



Week-3

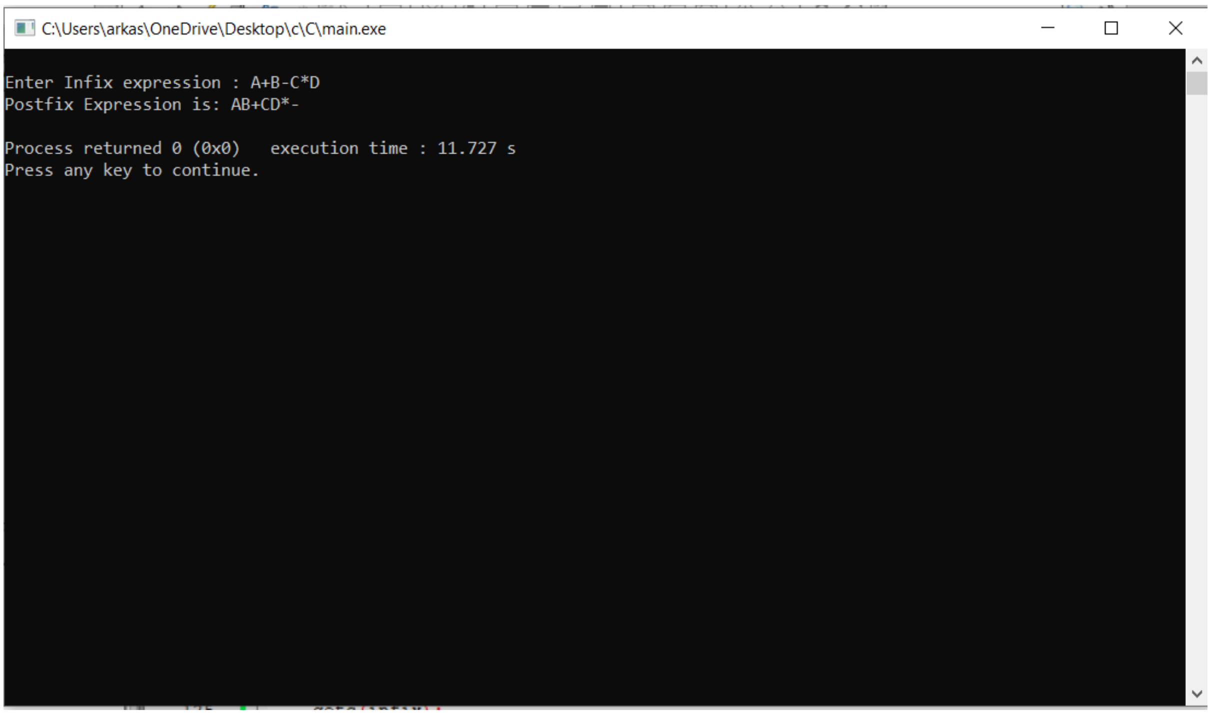
Date / /
Page
Puffin

```
void push (char ele)
{ }
char pop () ~~char~~
{}
int is_operator (char symbol)
{
    if (symbol == '^' || '/' || '+' || '-')
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int higher (char symbol)
{
    switch(symbol)
    {
        case '^':
            return (3);
            break;
        case '*'
        case '/':
            return (2);
            break;
      case '+';
      cas '-';
        return (1);
        break;
```

Date __/__/__
Page ______
Puffin

```
    default:
        return 0;
        break;
}  }

}
void  infixtopostfix (char
```

ScreenShot-



```
Enter Infix expression : A+B-C*D
Postfix Expression is: AB+CD*-

Process returned 0 (0x0)   execution time : 11.727 s
Press any key to continue.
```

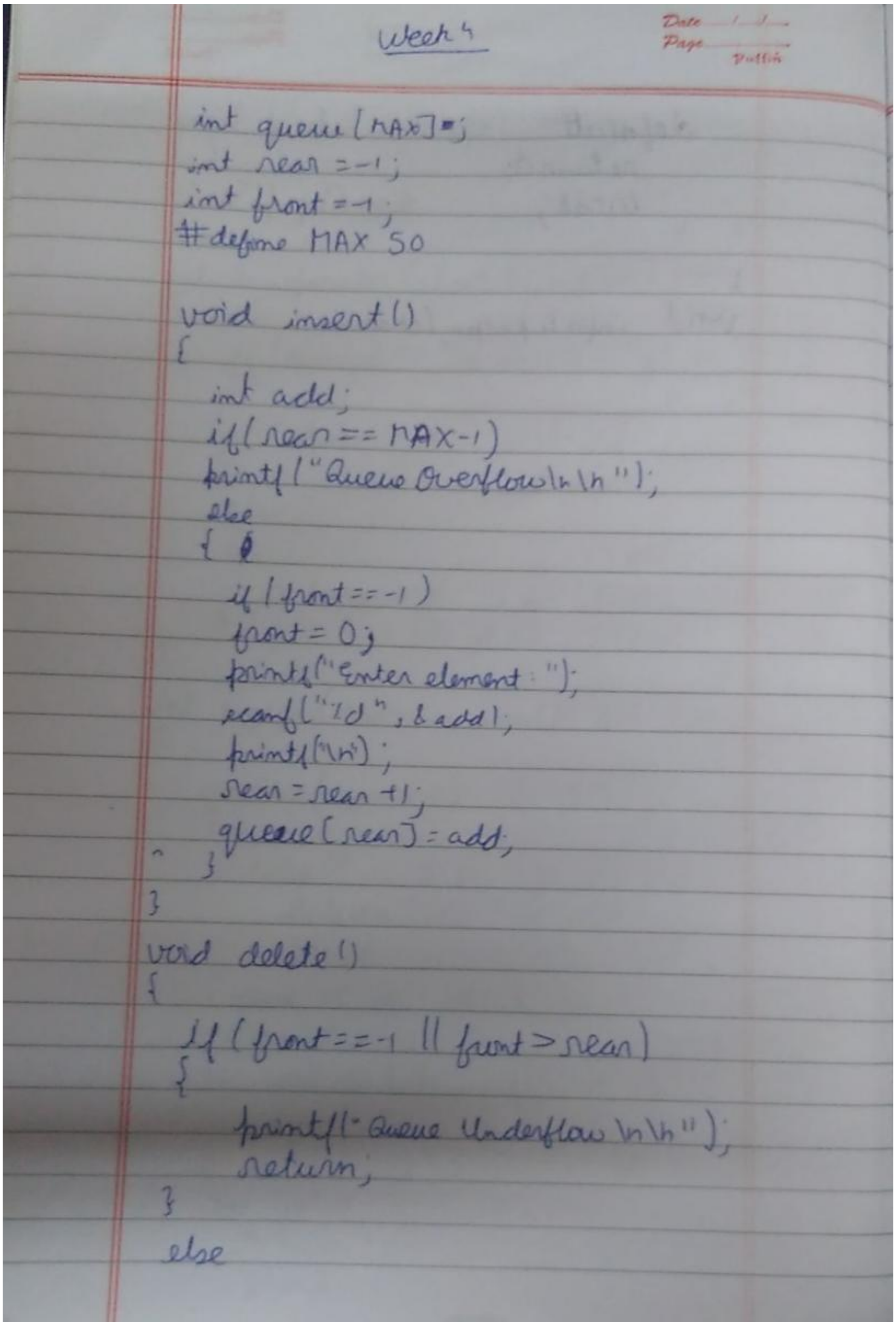
Lab Program 4

Source Code-

```
#include
<stdio.h>
            #define MAX 3

            void insert();
            void delete();
            void display();
            int queue[MAX];
            int rear = - 1;
            int front = - 1;
            void main()
            {
                int choice;
                while (1)
                {
                    printf("1.Insert element to queue \n");
                    printf("2.Delete element from queue \n");
                    printf("3.Display all elements of queue \n");
                    printf("4.Quit \n");
                    printf("Enter your choice : ");
                    scanf("%d", &choice);
                    switch(choice)
                    {
                        case 1:
                            insert();
                            break;
                        case 2:
                            delete();
                            break;
                        case 3:
                            display();
                            break;
                        case 4:
                            exit(0);
                        default:
                            printf("Enter a valid choice\n\n");
                    }
                }
            }

            void insert()
            {
```

```
    int add;
    if (rear == MAX - 1)
    printf("Queue Overflow\n\n");
    else
    {
        if (front == - 1)
            front = 0;
        printf("Enter element: ");
        scanf("%d", &add);
        printf("\n");
        rear = rear + 1;
        queue[rear] = add;
    }
}

void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow\n\n");
        return ;
    }
    else
    {
        printf("Deleted element: %d\n\n", queue[front]);
        front = front + 1;
    }
}

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty\n\n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d\n", queue[i]);
        printf("\n");
    }
}
```

Writeup-



Week 4

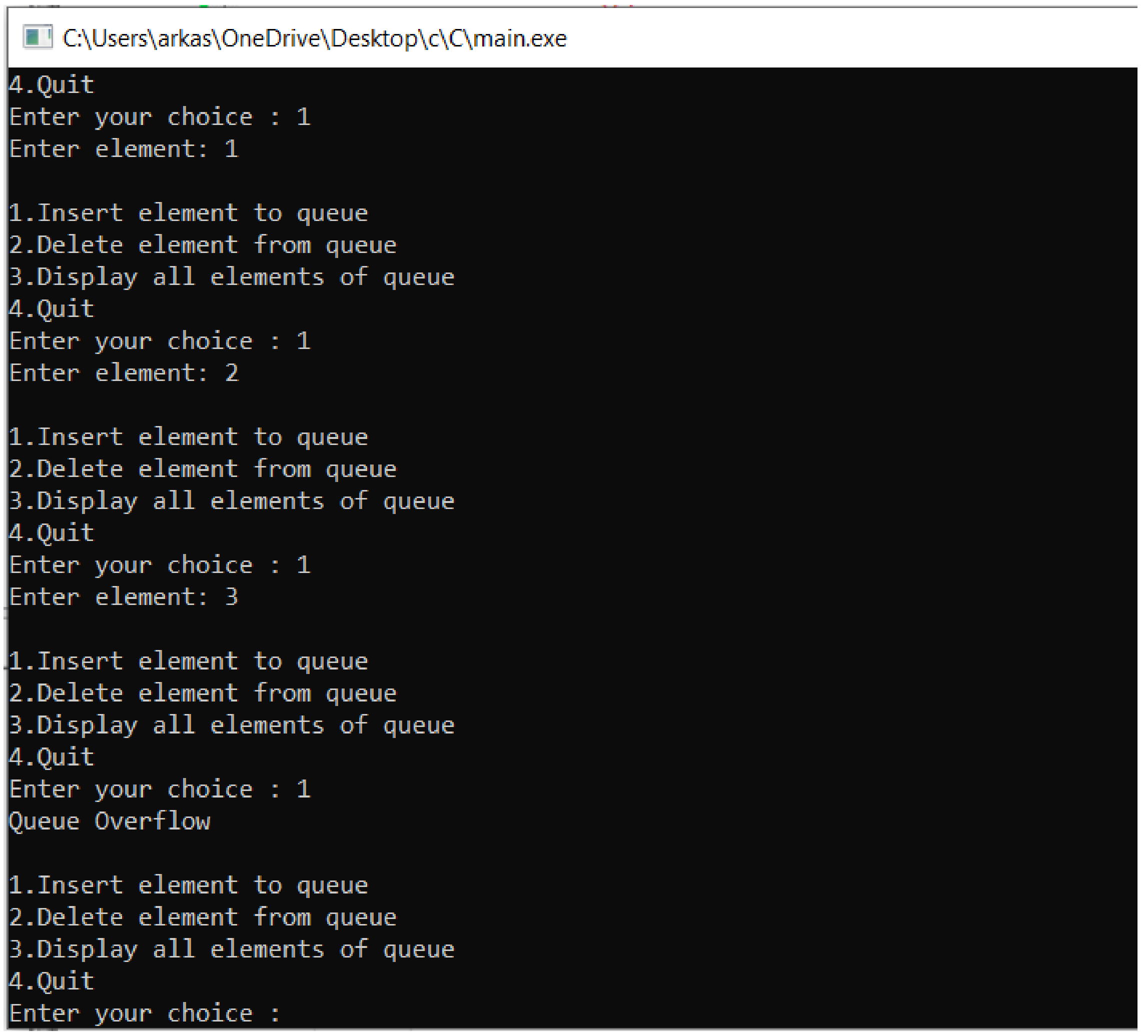
Date / /
Page

```
int queue[MAX];
int rear=-1;
int front=-1;
#define MAX 50

void insert()
{
  int add;
  if(rear==MAX-1)
  printf("Queue Overflow\n\n");
  else
  {
    if(front==-1)
    front=0;
    printf("Enter element: ");
    scanf("%d", &add);
    printf("\n");
    rear=rear+1;
    queue[rear]=add;
  }
}
void delete()
{
  if(front==-1 || front>rear)
  {
      printf("Queue Underflow\n\n");
      return;
  }
  else
```

Date__/__/__
Page____
Puffin

```
    {
     printf("Deleted element:%d\n\n", queue[front]),
     front = front +1;
    }
}
void display ()
{
   int i;
   if (front = =-1)
      printf("Queue is empty \n\n");
   else
   {
     printf(Queue is: \n");
     for (i= front; i<=rear; i++)
         printf("%d \n", queue[i]);
     printf(" \n"),
   }
}
```

ScreenShot-



C:\Users\arkas\OneDrive\Desktop\c\C\main.exe

```
4.Quit
Enter your choice : 1
Enter element: 1

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Enter element: 2

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Enter element: 3

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Queue Overflow

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice :
```

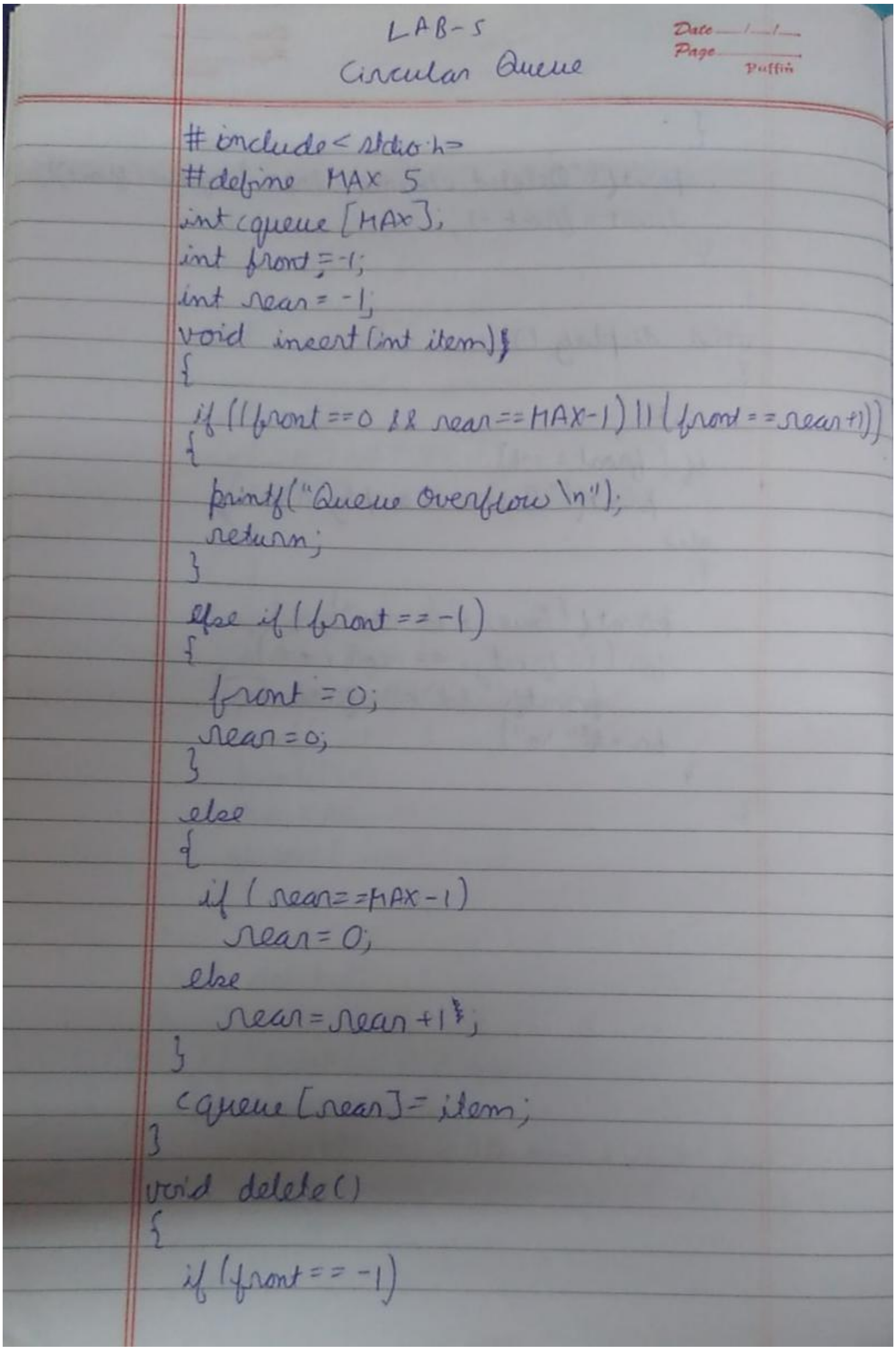
Lab Program 5

Source code-

```
#include<stdio.h>
                    # define MAX 5
                    int cqueue[MAX];
                    int front = -1;
                    int rear = -1;
                    void insert(int item)
                    {
                        if((front == 0 && rear == MAX-1) || (front == rear+1))
                        {
                            printf("Queue Overflow\n");
                            return;
                        }
                        else if(front == -1)
                        {
                            front = 0;
                            rear = 0;
                        }
                        else
                        {
                            if(rear == MAX-1)
                                rear = 0;
                            else
                                rear = rear+1;
                        }
                        cqueue[rear] = item ;
                    }
                    void delete()
                    {
                        if(front == -1)
                        {
                            printf("Queue Underflow\n");
                            return ;
                        }
                        printf("Element deleted from queue is : %d\n",cqueue[front]);
                        if(front == rear)
                        {
                            front = -1;
                            rear=-1;
                        }
                        else
                        {
                            if(front == MAX-1)
```

```
            front = 0;
        else
            front = front+1;
    }
}
void display()
{
    int front_pos = front,rear_pos = rear;
    if(front == -1)
    {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements :\n");
    if( front_pos <= rear_pos )
        while(front_pos <= rear_pos)
        {
            printf("%d ",cqueue[front_pos]);
            front_pos++;
        }
    else
    {
        while(front_pos <= MAX-1)
        {
            printf("%d ",cqueue[front_pos]);
            front_pos++;
        }
        front_pos = 0;
        while(front_pos <= rear_pos)
        {
            printf("%d ",cqueue[front_pos]);
            front_pos++;
        }
    }
    printf("\n");
}
int main()
{
    int choice,item;
    do
    {
        printf("1.Insert\n2.Delete\n3.Display\n4.Quit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1 :
                printf("Input the element for insertion in queue : 
");
                scanf("%d", &item);
                printf("\n");
                insert(item);
                break;
            case 2 :
                delete();
                break;
            case 3:
                display();
                break;
            case 4:
                break;
            default:
                printf("Enter a valid choice!!\n");
        }
    }while(choice!=4);
    return 0;
}
```

Writeup-



LAB-5
Circular Queue

Date __/__/__
Page ______
Puffin

```
#include<stdio.h>
#define MAX 5
int cqueue[MAX];
int front=-1;
int rear= -1;
void insert(int item)
{
 if((front==0 && rear==MAX-1)||(front==rear+1))
 {
  printf("Queue Overflow\n");
  return;
 }
 else if(front==-1)
 {
  front =0;
  rear=0;
 }
 else
 {
  if(rear==MAX-1)
   rear=0;
 else
   rear=rear+1;
 }
 cqueue[rear]= item;
}
void delete()
{
 if(front==-1)
```

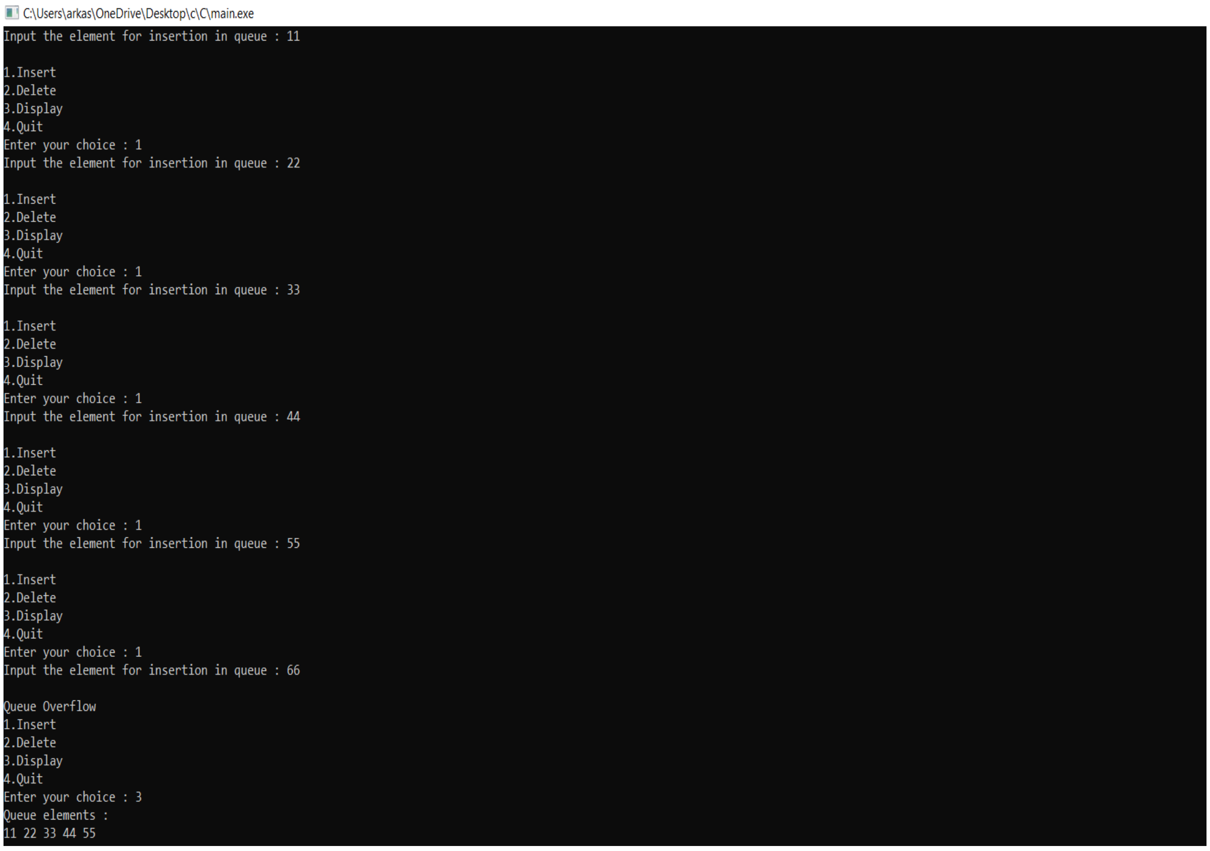
Date / /
Page
Puffin

```
{
 printf("Queue Underflow \n");
 return;
}
printf("Elemend deleted from queue is %d \n",(queue[front]);
if (front==rear)
{
 front=-1;
 rear=-1;
}
else
{
 if (front==MAX-1)
  front=0;
 else
  front=front+1;
 }
}
void display()
{
 int front_pos=front, rear_pos=rear;
 if (front==-1)
 {
  printf("~~Que~~ Queue empty \n");
  return;
 }
 printf("Queue elements: \n");
 if (front_pos<=rear_pos)
    while (front_pos<=rear_pos)
  {
   printf ("%d ", queue [front_pos]);
   front_pos ++;
  }
 else
 {
  while (front_pos <= MAX -1)
  {
   printf (%d ", queue [front_pos]);
   front_pos ++;
  }
  front_pos = 0;
  while (front_pos <= rear_pos)
  {
   printf ("%d", queue [front_pos]);
   front_pos ++;
  }
 }
 printf ("\n");
}
int main()
{
int choice, item;
do
{
 printf ("1.Insert\n2. Delete \n3. Display\n4. Quit\n");
printf ("Enter your choice: ");
scanf ("%d", &choice);
switch (choice)
```

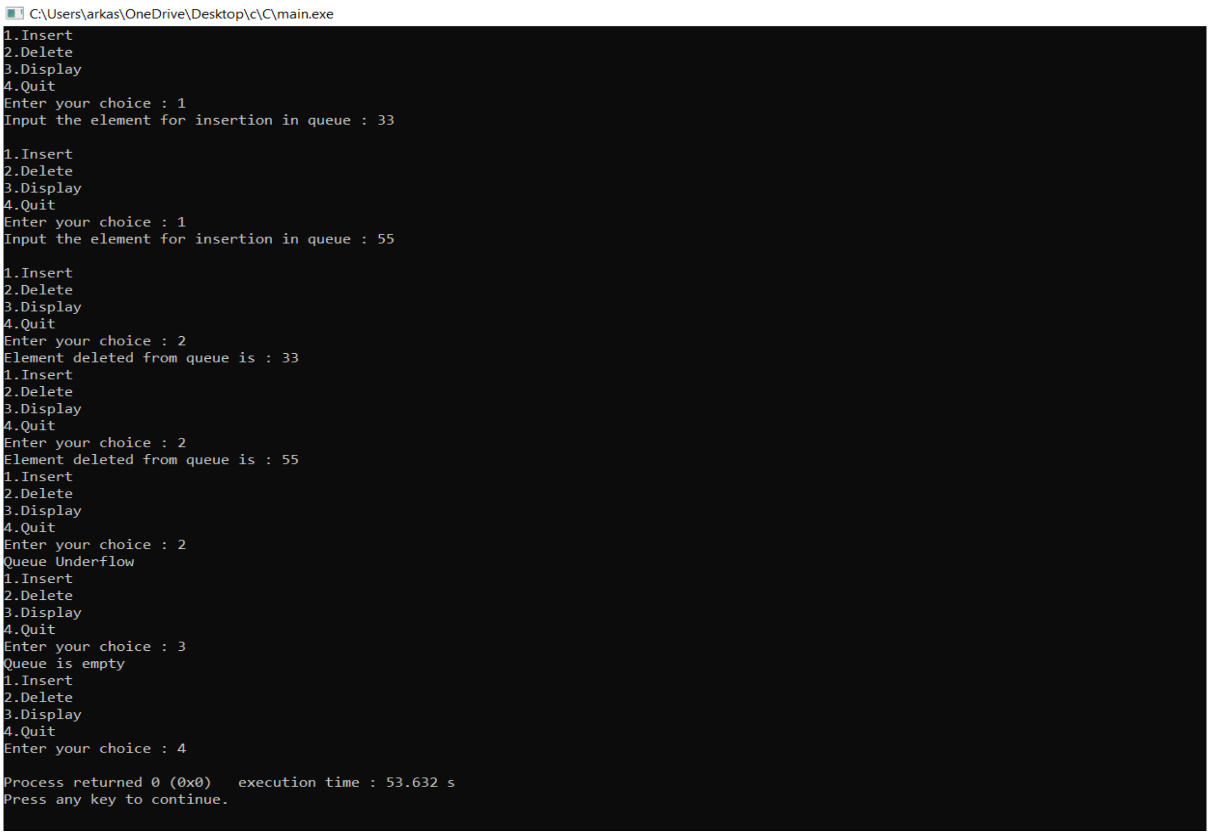
Date__/__/__
Page____
Puffin

```
{
 case 1 {:
  printf ("Element for insertion: ");
  scanf ("%d", &item);
  printf ("\n");
  insert (item);
  break;
 case 2:
  delete ();
  break;
 case 3 {:
  display ();
  break;
 case 4:
  break;
 default:
  printf ("Enter valid choice!! \n");
 }
} while (choice != 4);
return 0;
}.
```

ScreenShot-



```
C:\Users\arkas\OneDrive\Desktop\c\C\main.exe
Input the element for insertion in queue : 11

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 22

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 33

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 44

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 55

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 66

Queue Overflow
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
11 22 33 44 55
```



```
C:\Users\arkas\OneDrive\Desktop\c\C\main.exe
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 33

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 55

1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 33
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 55
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Queue Underflow
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue is empty
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 4

Process returned 0 (0x0)   execution time : 53.632 s
Press any key to continue.
```

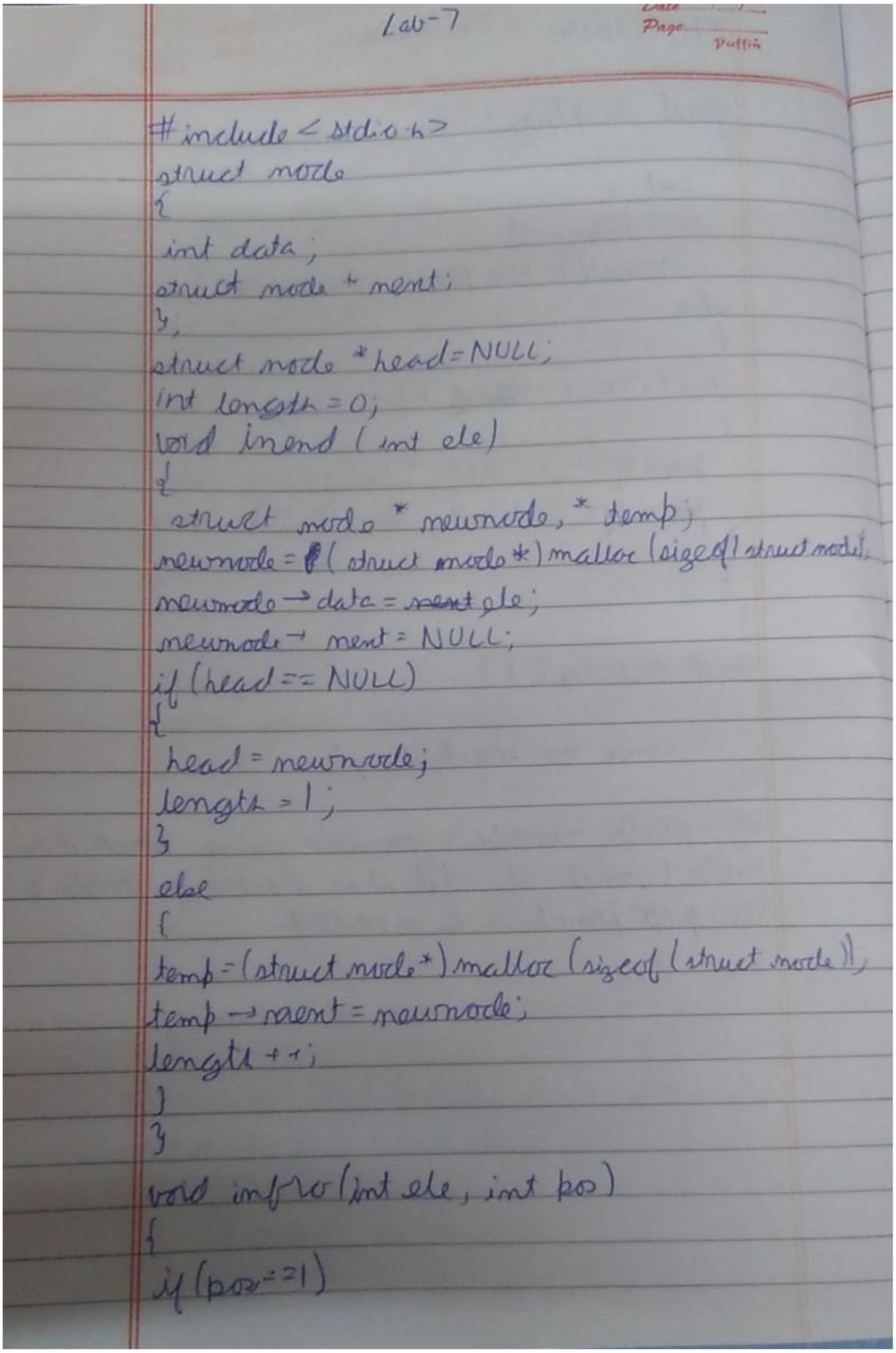
Lab Program 7

Source Code-

```
#include<stdio.h>

struct node
{
    int data;
    struct node *next;
};

struct node *head=NULL;

int length=0;

void inend(int ele)
{
    struct node *newnode,*temp;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(head==NULL)
    {
        head=newnode;
        length=1;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        length++;
    }

}

void infro(int ele)
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=ele;
    temp->next=head;
    head=temp;
    length++;
}

void inran(int ele,int pos)
{
    if(pos==1)
        infro(ele);
    else if(pos>=length)
        inend(ele);
    else
    {
        struct node *inst;
        inst=(struct node*)malloc(sizeof(struct node));
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        for(int i=1;i<pos-1;i++)
        {
               temp=temp->next;
        }
        inst->data=ele;
        inst->next=temp->next;
        temp->next=inst;
        length++;

    }

}

void del(int ele)
{
     struct node *temp,*del;
     temp=(struct node*)malloc(sizeof(struct node));
     del=(struct node*)malloc(sizeof(struct node));
     del=NULL;
     if(head->data==ele)
     {
         del=head;
         head=head->next;
         del->next=NULL;
     }
     else
     {
         temp=head;
         while(temp->next!=NULL)
         {
            if(temp->next->data==ele)
            {

                del=temp->next;
                temp->next=del->next;
                del->next=NULL;
                length--;
                break;
            }
            else
            {
                temp=temp->next;
            }

         }
     }
     if(del==NULL)
     {
         printf("\nElement not found.\n");
     }
}

void display()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head;
    if(temp==NULL)
    {
        printf("\n List is empty \n");
    }
    else
    {
        printf("\nThe contents of the list are :\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }

}

int main()
```

```
{
    int choice,ele,pos;
    char ch;
    do
    {
    printf("\n1. Inset at end \n2.Insert at front \n3.Insert in
between \n4. Display  \n5. Delete  \n6.Exit");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                inend(ele);
                break;
        case 2: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                infro(ele);
                break;
        case 3: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                printf("Enter the position \n");
                scanf("%d",&pos);
                inran(ele,pos);
                break;
        case 4: display();
                break;
        case 5: printf("Enter the element to be deleted\n");
                scanf("%d",&ele);
                del(ele);
                break;
    }
    }while(choice!=6);
    return 0;
}
```

Writeup-



Lab-7

Date / /
Page
Puffin

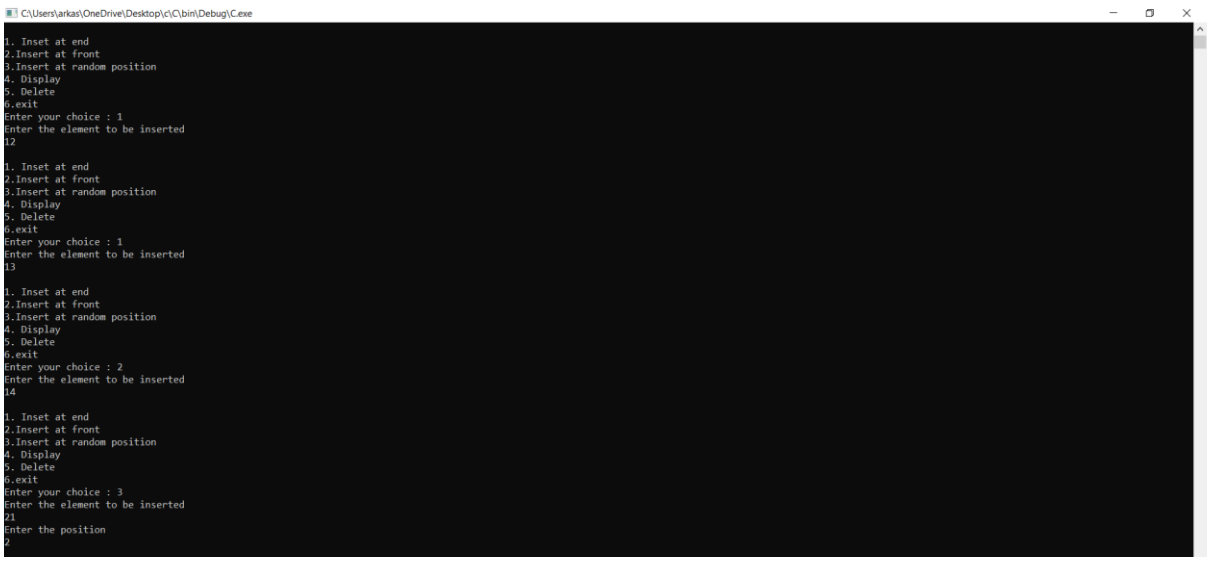
```
#include <stdio.h>
struct node
{
 int data;
struct node * next;
};
struct node *head=NULL;
int length=0;
void inend (int ele)
{
 struct node * newnode, * temp;
newnode = ~~f~~ (struct node *) malloc (sizeof(struct node));
newnode → data = ~~next~~ ele;
newnode → next = NULL;
if (head == NULL)
{
 head = newnode;
length = 1;
}
else
{
temp = (struct node*) malloc (sizeof (struct node));
temp → next = newnode;
length ++;
}
}
void infro(int ele, int pos)
{
if (pos==1)
```

Date __/__/__
Page ______

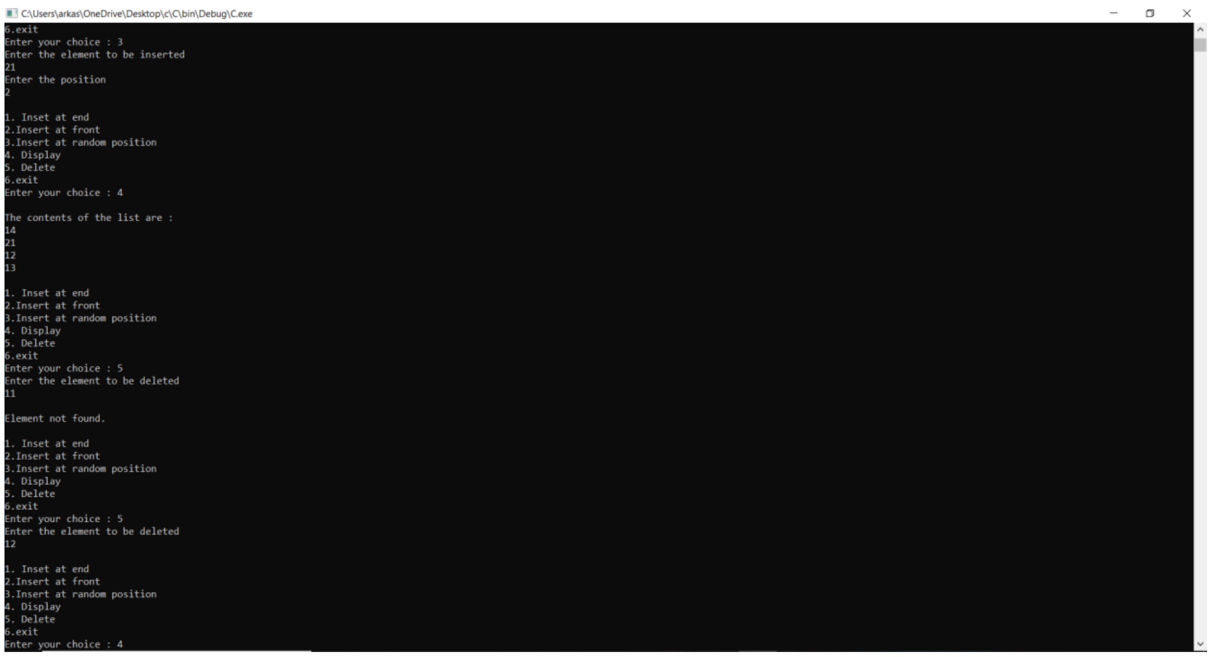
```
infro (ele);
else if (pos>=length)
 inend (ele);
else
 {
struct node * inst;
inst = (struct node*) malloc (sizeof (struct node));
struct node *temp;
temp = (struct node *) malloc (sizeof ( struct node));
temp = head;
for (int i = 1; i< pos-1; i++)
{
 temp = temp -> next;
 }
Inst -> data = ele;
int -> next = temp -> next;
temp -> next = int;
length ++;
 }
}
void del (int ele)
{
struct node * temp, *del;
temp = (struct node*) malloc (sizeof (struct node));
del = (struct node*) malloc (sizeof (struct node));
del = NULL;
if (head -> data == ele)
{
 del = head;
head=head→next;
del→next=NULL;
}
else
{
temp=head;
while(temp→next!=NULL)
{
 if(temp→next→data==ele)
{
del=temp→next;
temp→next=del→next;
del→next=NULL;
length--;
break;
}
else
{
  temp=temp→next;
 }
 }
}
if(del==NULL)
{
printf("\n Element not found.\n");
}
}
void display()
{
```

```
struct node* temp;
temp=(struct node*)malloc(sizeof(struct node));
temp=head;
if(temp==NULL)
{
printf("\n List is empty \n");
}
else
{
printf("\nThe contents of the list are: \n");
while(temp!=NULL)
{
 ~~while~~(printf("%d\n", temp→data);
 temp=temp→next;
 }
}
}
```

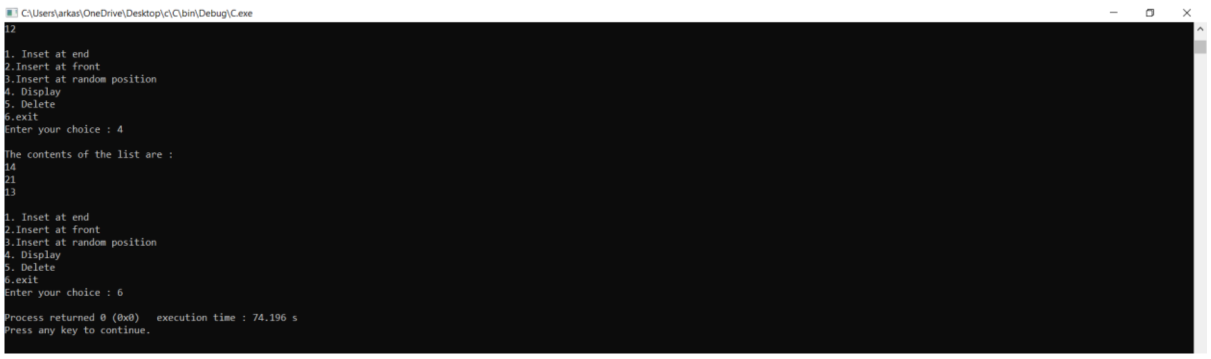
ScreenShot-



```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 2
Enter the element to be inserted
14

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 3
Enter the element to be inserted
21
Enter the position
2
```



```
6.exit
Enter your choice : 3
Enter the element to be inserted
21
Enter the position
2

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 4

The contents of the list are :
14
21
12
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
11

Element not found.

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 4
```



```
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 4

The contents of the list are :
14
21
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 6

Process returned 0 (0x0)   execution time : 74.196 s
Press any key to continue.
```

Lab Program 8

Source Code-

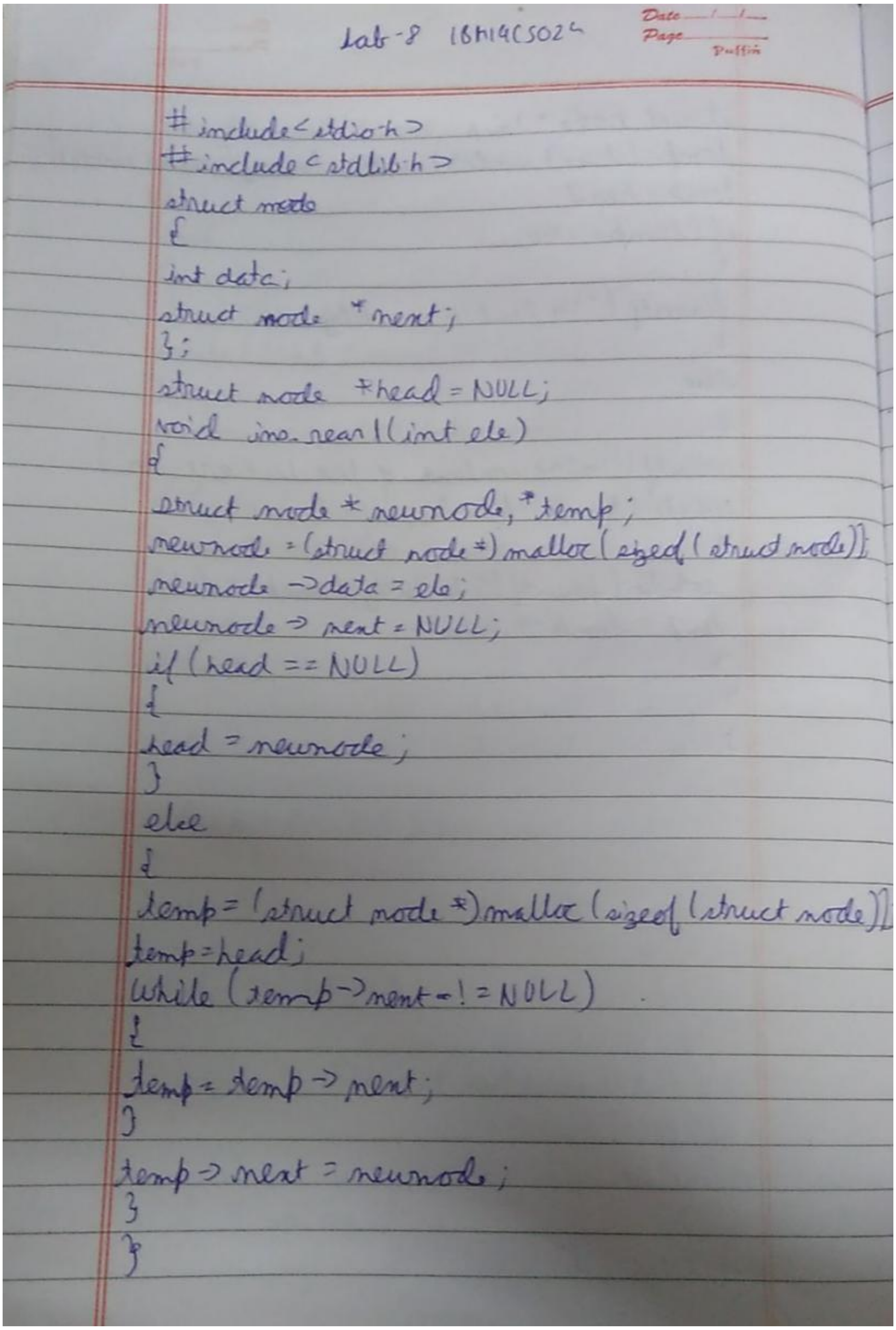
```
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
};
struct node *head1=NULL;
struct node *head2=NULL;
int length1=0;
int length2=0;
void ins_rear1(int ele)
{
    struct node *newnode, *temp;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(head1==NULL)
    {
        head1=newnode;
        length1=1;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head1;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        length1++;
    }
}
void ins_rear2(int ele)
{
    struct node *newnode, *temp;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(head2==NULL)
    {
        head2=newnode;
        length2=1;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head2;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        length2++;
    }
}
void del_front1()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    if(head1==NULL)
    {
        printf("List is empty!!\n");
    }
    else
    {
        temp=head1;
        head1=temp->next;
        printf("%d is removed front front!!\n",temp->data);
        free(temp);
    }
}
void del_front2()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    if(head2==NULL)
    {
        printf("List is empty!!\n");
    }
    else
    {
        temp=head2;
        head2=temp->next;
        printf("%d is removed front front!!\n",temp->data);
        free(temp);
    }
}
void del_rear1()
{
    struct node *temp, *run;
    temp=(struct node *)malloc(sizeof(struct node));
    run=(struct node*)malloc(sizeof(struct node));
    temp=head1->next;
    if(run==NULL)
    {
        printf("List is empty!!\n");
    }
    else
    {
        while(temp->next!=NULL)
        {
            run=temp;
            temp=temp->next;
        }
        printf("%d is deleted from rear\n",temp->data);
        run->next=NULL;
        free(temp);
    }
}
void del_rear2()
{
    struct node *temp, *run;
    temp=(struct node *)malloc(sizeof(struct node));
    run=(struct node*)malloc(sizeof(struct node));
    temp=head2->next;
    run=head2;
    if(run==NULL)
    {
        printf("List is empty!!\n");
    }
    else
    {
        while(temp->next!=NULL)
        {
            run=temp;
            temp=temp->next;
        }
        printf("%d is deleted from rear\n",temp->data);
        run->next=NULL;
        free(temp);
    }
```

```
}
void del1(int ele)
{
    struct node *temp, *del;
    temp=(struct node*)malloc(sizeof(struct node));
    del=(struct node*)malloc(sizeof(struct node));
    del=NULL;
    if(head1->data==ele)
    {
        del=head1;
        head1=head1->next;
        del->next=NULL;
    }
    else
    {
        temp=head1;
        while(temp!=NULL)
        {
            if(temp->next->data==ele)
            {
                del=temp->next;
                temp->next=del->next;
                del->next=NULL;
                length1--;
                break;
            }
            else
            {
                temp=temp->next;
            }
        }
    }
    if(del==NULL)
    {
        printf("\nElement not found!!\n");
    }
}
void del2(int ele)
{
    struct node *temp, *del;
    temp=(struct node*)malloc(sizeof(struct node));
    del=(struct node*)malloc(sizeof(struct node));
    del=NULL;
    if(head2->data==ele)
    {
        del=head2;
        head1=head2->next;
        del->next=NULL;
    }
    else
    {
        temp=head2;
        while(temp->next!=NULL)
        {
            if(temp->next->data==ele)
            {
                del=temp->next;
                temp->next=del->next;
                del->next=NULL;
                length2--;
                break;
            }
            else
            {
                temp=temp->next;
            }
        }
    }
    if(del==NULL)
    {
        printf("\nElement not found!!\n");
    }
}
void sort1(struct node *h)
{
    int i,j,a;

    struct node *temp1;
    struct node *temp2;

    for(temp1=h;temp1!=NULL;temp1=temp1->next)
      {
        for(temp2=temp1->next;temp2!=NULL;temp2=temp2->next)
          {
            if(temp2->data < temp1->data)
              {
                a = temp1->data;
                temp1->data = temp2->data;
                temp2->data = a;
              }
           }
       }
```

```
}
void sort2(struct node *h)
{
    int i,j,a;

    struct node *temp1;
    struct node *temp2;

    for(temp1=h;temp1!=NULL;temp1=temp1->next)
      {
        for(temp2=temp1->next;temp2!=NULL;temp2=temp2->next)
          {
            if(temp2->data < temp1->data)
              {
                a = temp1->data;
                temp1->data = temp2->data;
                temp2->data = a;
              }
           }
       }
}
void rev1(struct node *h)
{
    int i,j,a;

    struct node *temp1;
    struct node *temp2;

    for(temp1=h;temp1!=NULL;temp1=temp1->next)
    {
        for(temp2=temp1->next;temp2!=NULL;temp2=temp2->next)
        {
            a = temp1->data;
            temp1->data = temp2->data;
            temp2->data = a;
        }
    }
}
void rev2(struct node *h)
{
    int i,j,a;

    struct node *temp1;
    struct node *temp2;

    for(temp1=h;temp1!=NULL;temp1=temp1->next)
    {
        for(temp2=temp1->next;temp2!=NULL;temp2=temp2->next)
        {
            a = temp1->data;
            temp1->data = temp2->data;
            temp2->data = a;
        }
    }
}
void conc()
{
    struct node *temp, *run;
    temp=(struct node*)malloc(sizeof(struct node));
    run=(struct node*)malloc(sizeof(struct node));
    run=head1;
    if(length1==0 && length2==0)
    {
        printf("\nBoth lists empty\n");
    }
    else
    {
        temp=head1;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=head2;
    }
    printf("\nThe elements in the concatenated list is-\n");
    while(run!=NULL)
    {
        printf("%d\n",run->data);
        run=run->next;
    }
}
void display1()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head1;
    if(temp==NULL)
    {
        printf("\nList is empty!!\n");
    }
    else
    {
        printf("\nThe contents in list 1 are-\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }
}
void display2()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head2;
    if(temp==NULL)
    {
        printf("\nList is empty!!\n");
    }
    else
    {
        printf("\nThe contents in list 2 are-\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }
}
int main()
{
    int ch, item;
    printf("Choose any option-");
    do{
        printf("\n1.Insert at rear in list 1\n2.Insert at rear in
list 2\n3.Delete random from list 1\n4.Delete random from list
2\n5.Remove from front in list 1\n6.Remove from front in list
2\n7.Remove from rear in list 1\n8.Remove from rear in list
2\n9.Display list 1\n10.Display list 2\n11.Sort list 1\n12.Sort list
2\n13.Reverse list 1.\n14.Reverse list 2\n15.Concatenate list 1 and
list 2\n16.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:
            printf("\nEnter the number to be inserted: ");
            scanf("%d",&item);
            printf("\n");
            ins_rear1(item);
            break;
        case 2:
            printf("\nEnter the number to be inserted: ");
            scanf("%d",&item);
            printf("\n");
            ins_rear2(item);
            break;
        case 3:
            printf("\nEnter the number to be deleted: ");
            scanf("%d",&item);
            printf("\n");
            del1(item);
            break;
        case 4:
            printf("\nEnter the number to be deleted: ");
            scanf("%d",&item);
            printf("\n");
            del2(item);
            break;
        case 5:
            del_front1();
            break;
        case 6:
            del_front2();
            break;
        case 7:
            del_rear1();
            break;
        case 8:
            del_rear2();
            break;
        case 9:
            display1();
            break;
        case 10:
            display2();
            break;
        case 11:
            sort1(head1);
            break;
        case 12:
            sort2(head2);
            break;
        case 13:
            rev1(head1);
```

```
                break;
            case 14:
                rev2(head2);
                break;
            case 15:
                conc();
                break;
            case 16:
                break;
            default:
                printf("\nEnter a valid choice!!\n");
            }
        }while(ch!=16);
        return 0;
    }
```

Writeup-



Lab-8 1BM19CS024

Date / /
Page
Puffin

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head=NULL;
void ins_rear(int ele)
{
struct node *newnode,*temp;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=ele;
newnode->next=NULL;
if(head==NULL)
{
head=newnode;
}
else
{
temp=(struct node*)malloc(sizeof(struct node));
temp=head;
While(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
}
}
```

Date __/__/__
Page ____
Puffin

```
void del-front()
{
struct node *temp;
temp=(struct node *) malloc (sizeof (struct node));
if(head==NULL)
{
printf("List is Empty \n");
}
else
{
temp=head;
head=temp->~~hea~~ next;
printf("%d is removed \n", temp->data);
free(temp);
}
}
void del-rear()
{
struct node *temp, *run;
temp= (struct node *)malloc(sizeof(struct node));
run = (struct node *)malloc(sizeof(struct node));
temp =head -> next
if(run==NULL)
{
printf("List is Empty \n");
}
else
{
while (temp->next != NULL)
```

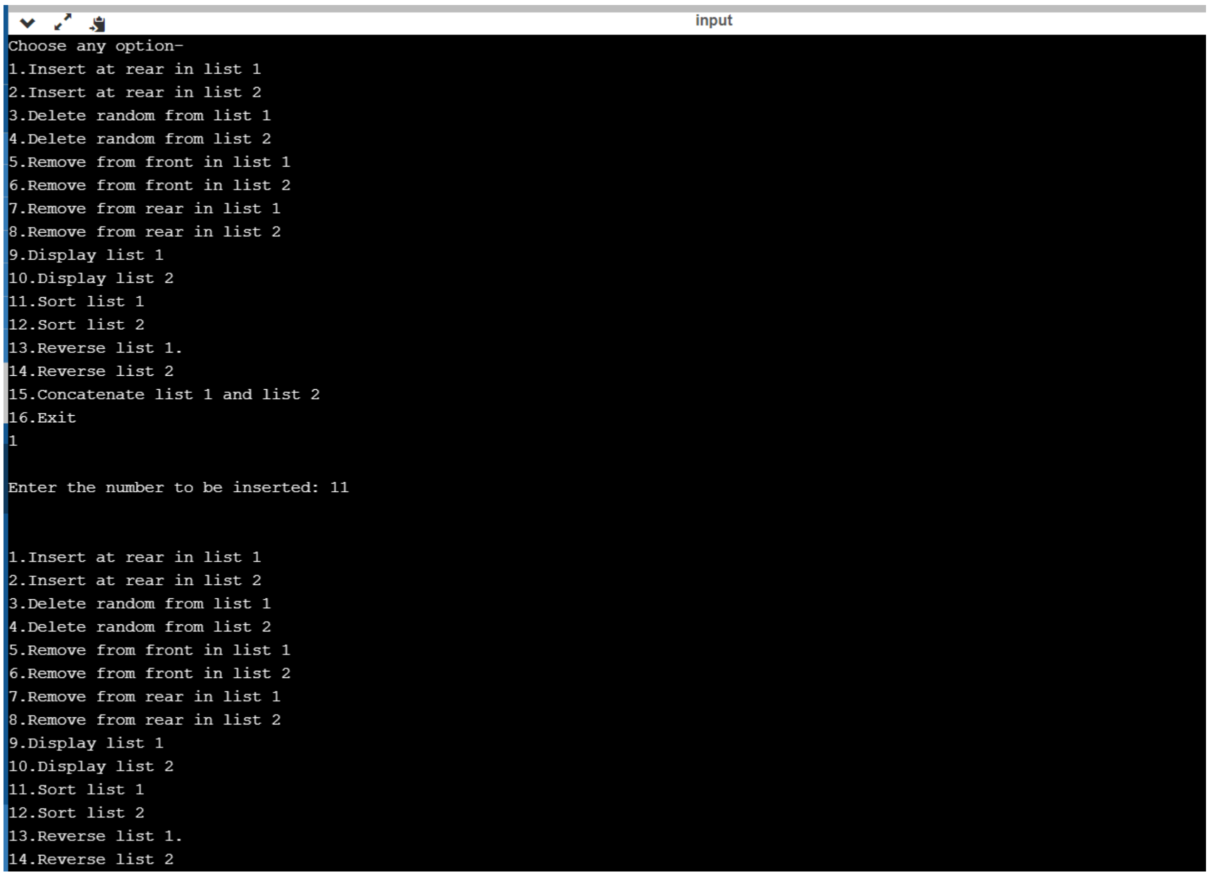
```
{
rem = temp;
temp = temp -> next;
}
printf ("%d is deleted \n", temp -> data);
rem -> next = NULL;
free(temp);
}
}
void del(int ele)
{
struct node *temp, *del;
temp=(struct node*)malloc(sizeof(struct node));
del = (struct node *)malloc (sizeof (struct node));
del = NULL;
if( head -> data = = ele)
{
del = head;
head = head -> next;
del -> next = NULL;
}
else
{
temp = head;
while (temp != NULL)
{
if (temp -> next -> data = = ele)
{
del = temp -> next;
```

Date __/__/__
Page ____
Puffin

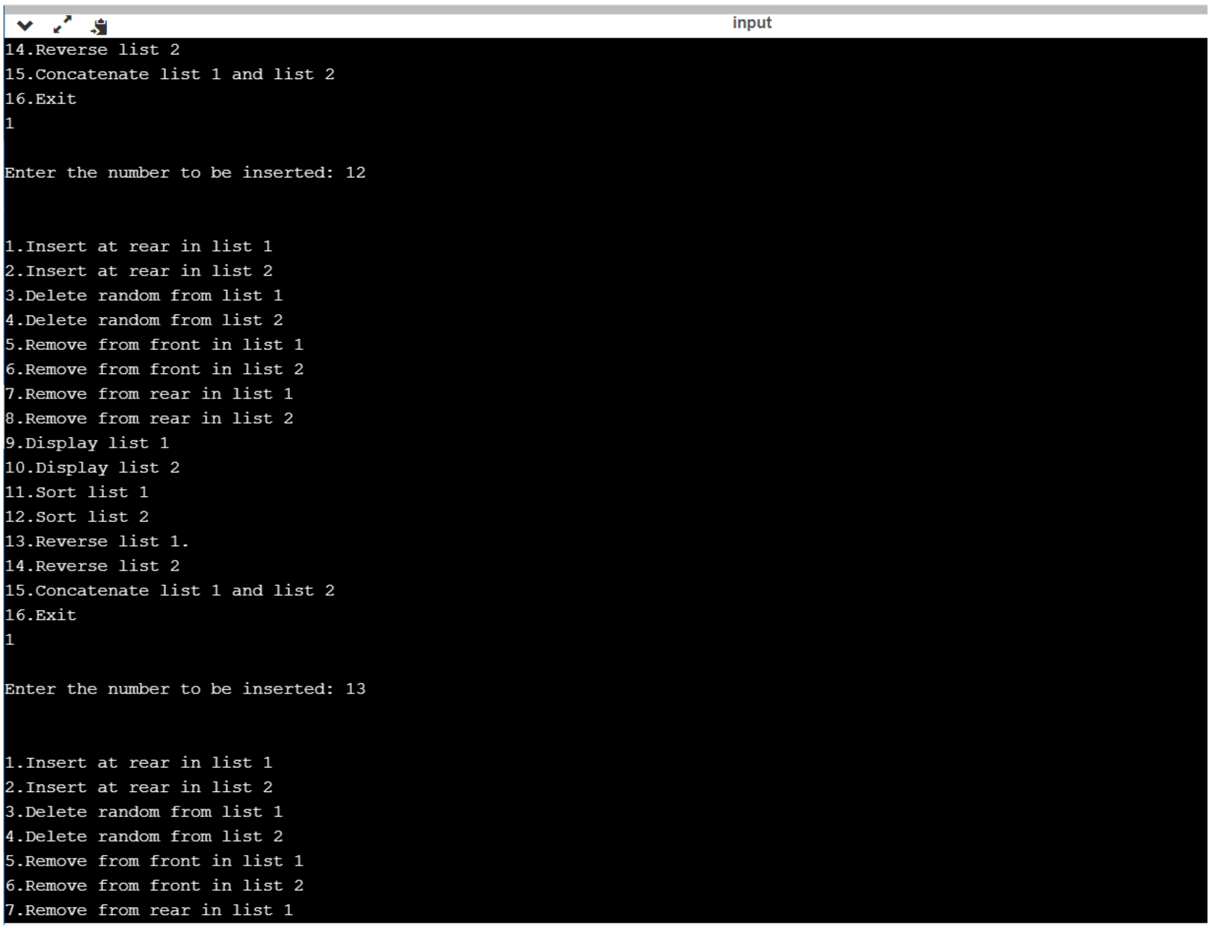
```
temp -> next = del -> next;
del -> next = NULL;
break;
}
else
{
temp = temp -> next;
}
}
}
if(del == NULL)
{
printf("\nElement not found\n");
}
}
void sort(struct node *h);
{
int j, i, a;
struct node * temp1;
struct node *temp2;
for(temp1=h; temp1!=NULL; temp1=temp1->next)
{
for(temp2=temp1; temp2!=NULL; temp2=temp2->next)
{
if(temp2->data < temp1->data)
{
a = temp1 -> data;
temp1 -> data = temp2 -> data;
temp2 -> data = a;
}}}}
```

```
void rev(struct node *h)
{
int i, j, a;
struct node *temp1, *temp2;
for(temp1=h; temp1!=NULL; temp1=temp1->next)
{
for(tem2=temp1; temp2!=NULL; temp2=temp2->next)
{
a=temp1->data;
temp1->data=temp2->data;
temp2->data=a;
}}}
void conc
{
struct node *temp, *run;
temp=(struct node *)malloc(sizeof(struct node));
run=(struct node *)malloc(sizeof(struct node));
run=head;
if(head1==NULL && head2==NULL)
{
printf("Both lists empty\n");
}
else
{
temp=head1;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=head2;
}
printf("\n The concatenated list - \n");
while (run !=NULL)
{
printf("%d \n", run->data);
run = run->next;
}}
```

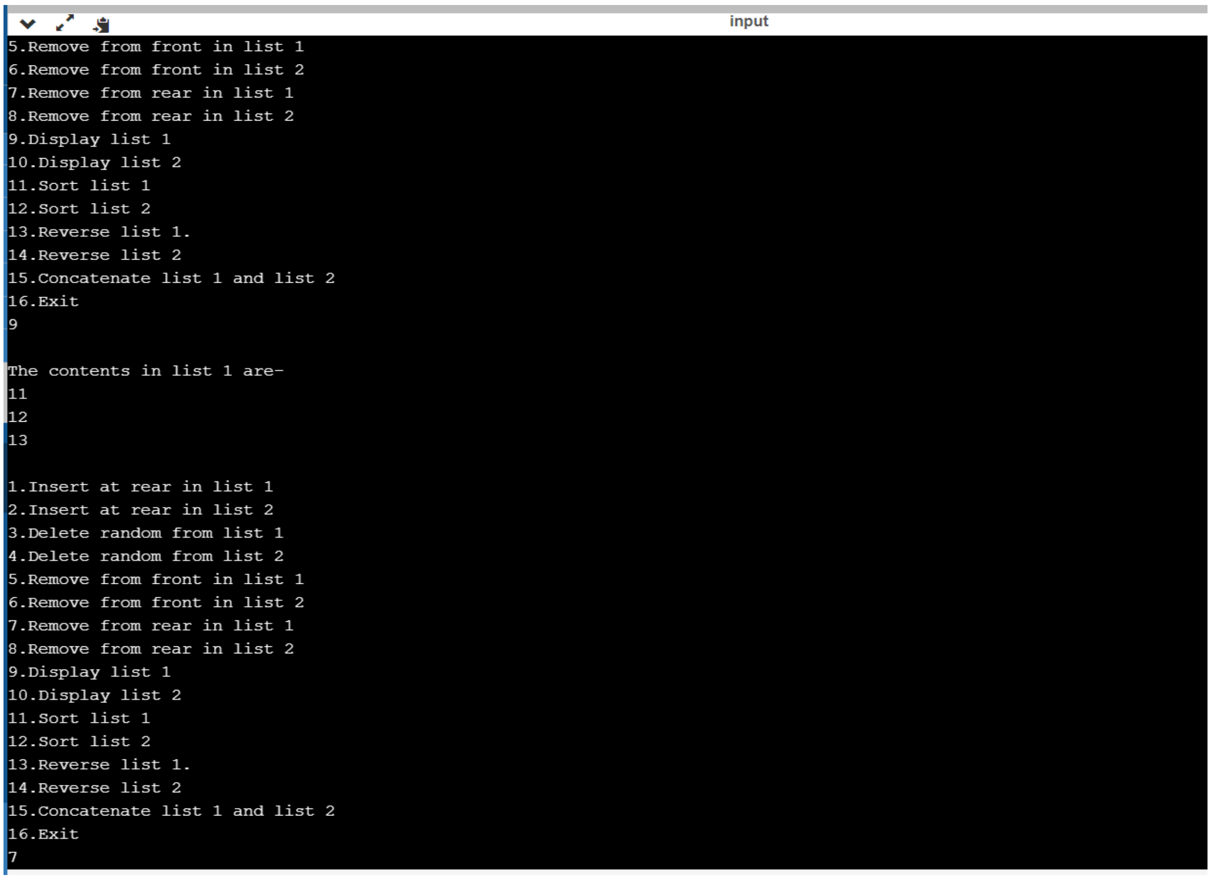
Screen Shots-


input

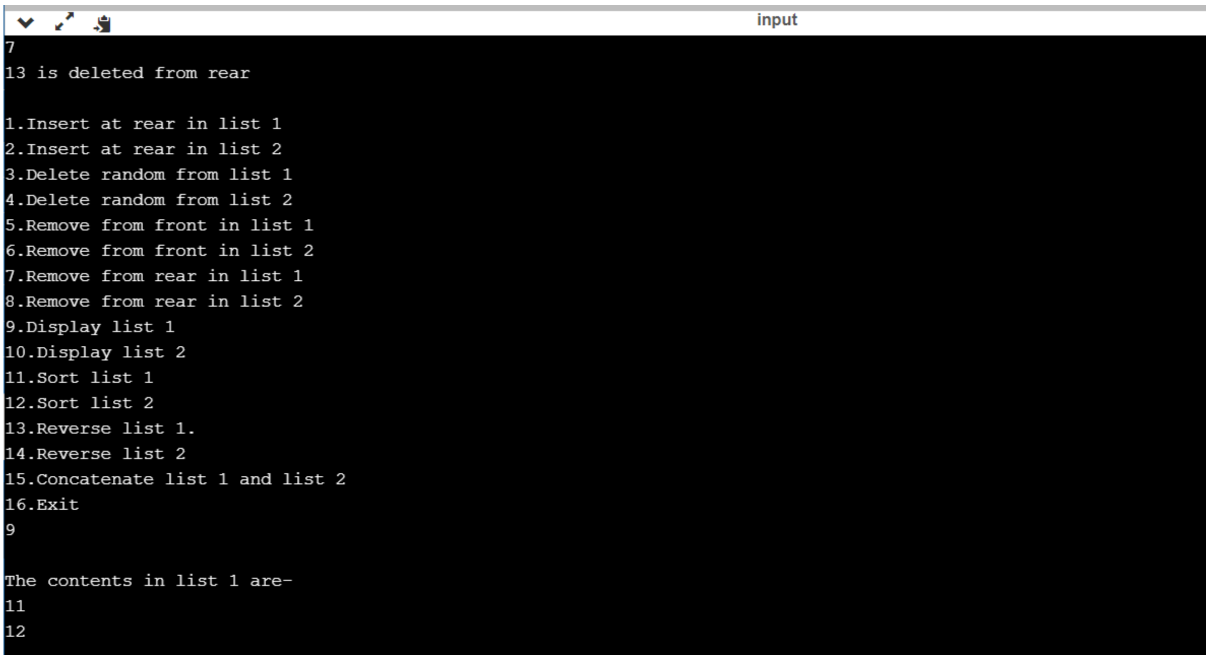
```
Choose any option-
1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
1

Enter the number to be inserted: 11


1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
```



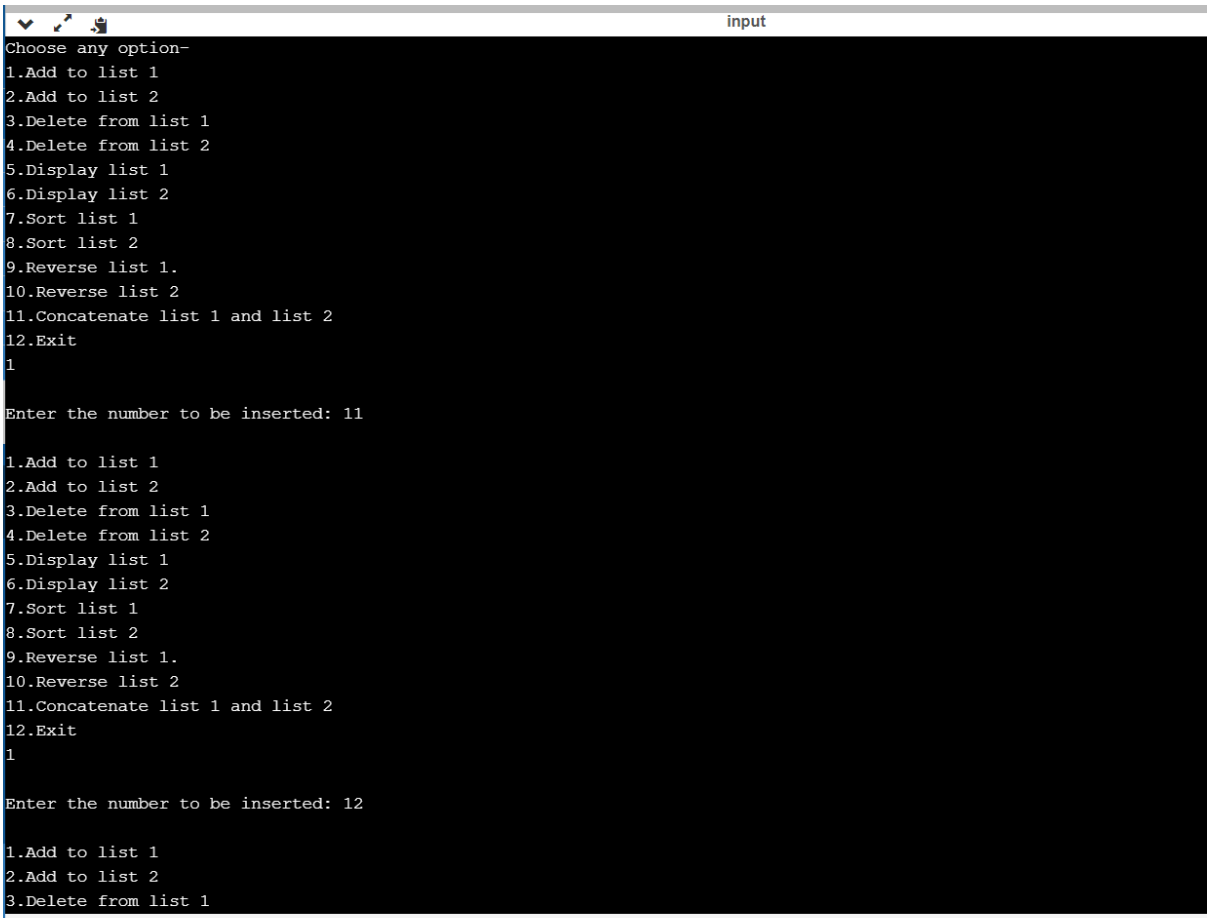
```
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
1

Enter the number to be inserted: 12


1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
1

Enter the number to be inserted: 13


1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
```



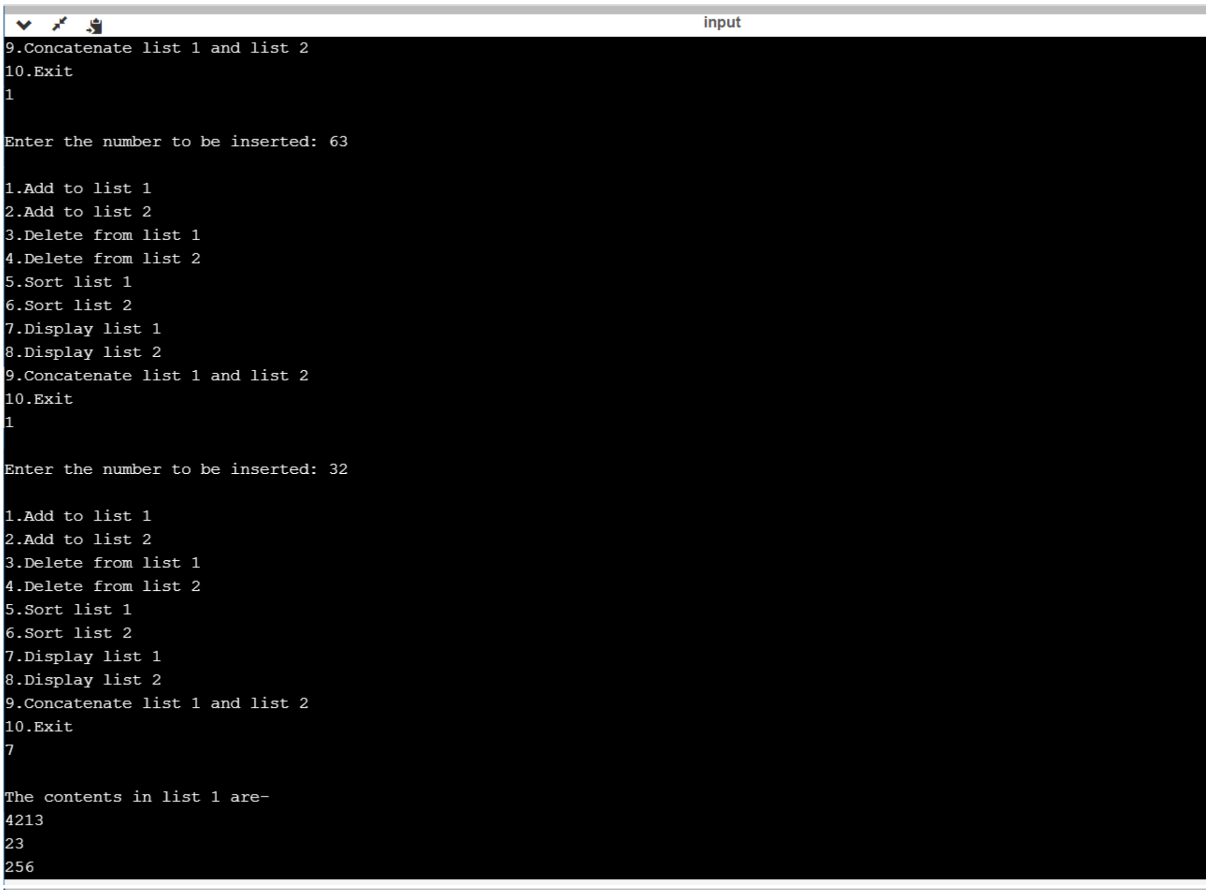
```
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
9

The contents in list 1 are-
11
12
13

1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
7
```



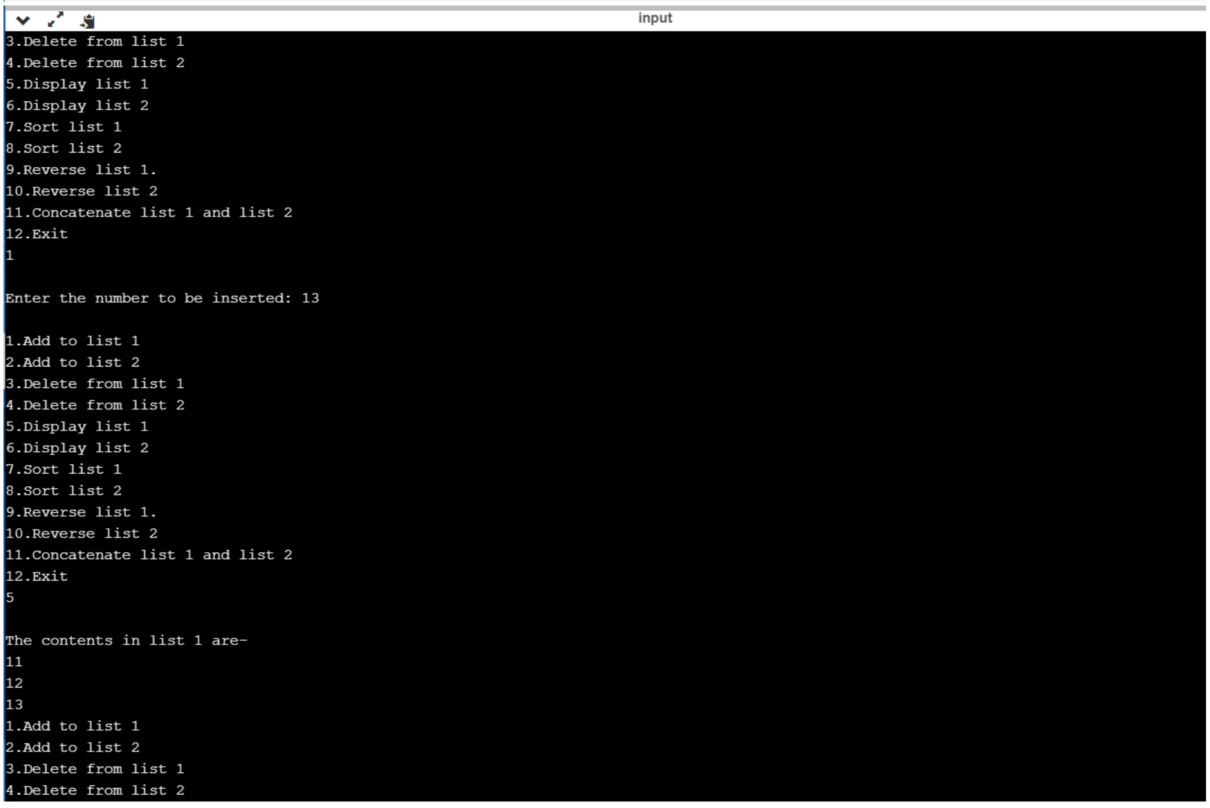
```
7
13 is deleted from rear

1.Insert at rear in list 1
2.Insert at rear in list 2
3.Delete random from list 1
4.Delete random from list 2
5.Remove from front in list 1
6.Remove from front in list 2
7.Remove from rear in list 1
8.Remove from rear in list 2
9.Display list 1
10.Display list 2
11.Sort list 1
12.Sort list 2
13.Reverse list 1.
14.Reverse list 2
15.Concatenate list 1 and list 2
16.Exit
9

The contents in list 1 are-
11
12
```



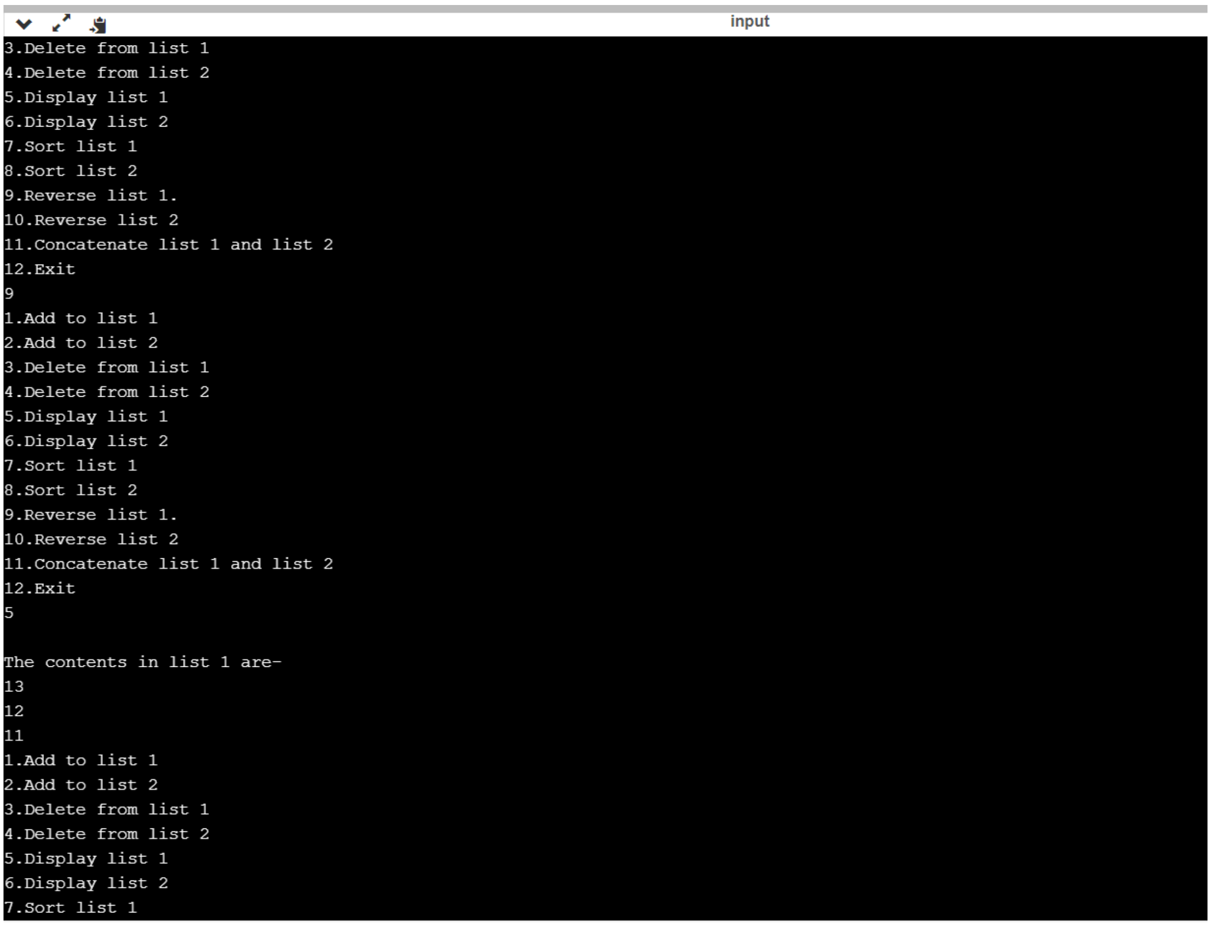
```
Choose any option-
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
1

Enter the number to be inserted: 11

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
1

Enter the number to be inserted: 12

1.Add to list 1
2.Add to list 2
3.Delete from list 1
```



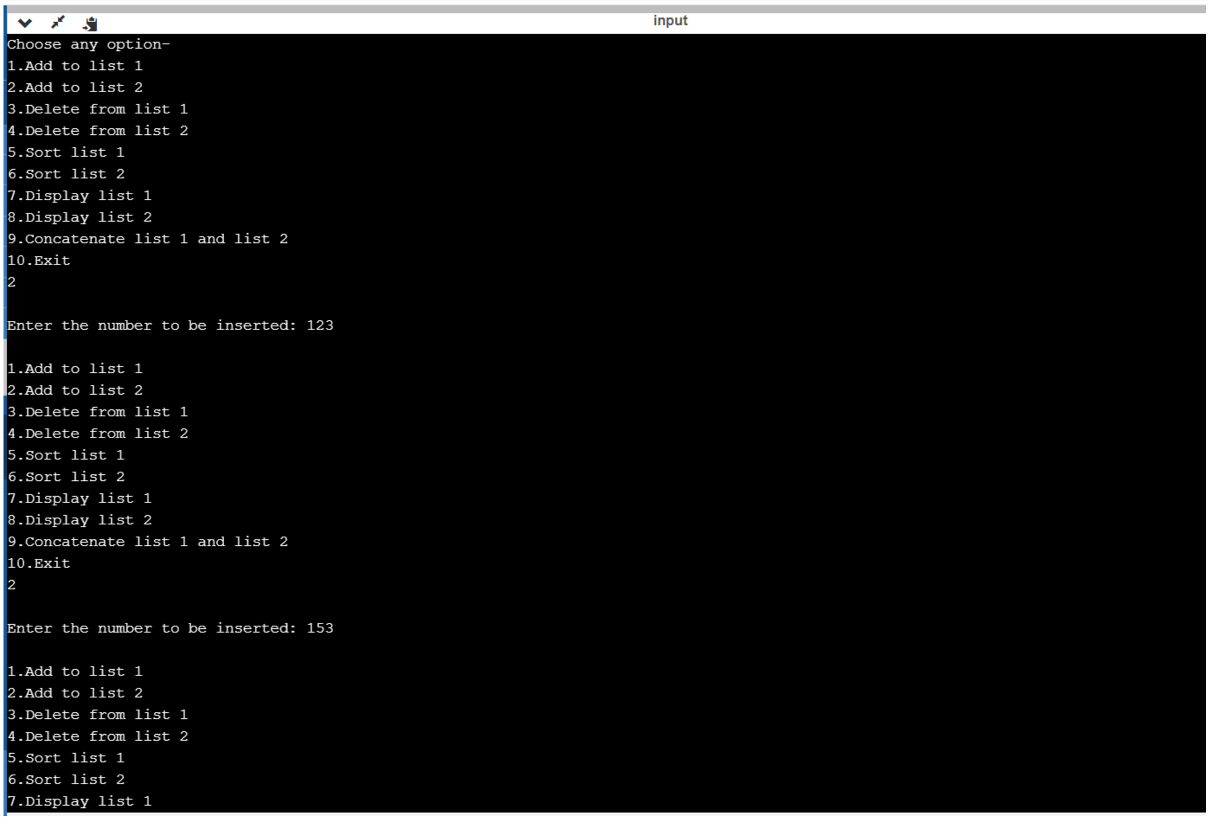
```
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 63

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 32

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
7

The contents in list 1 are-
4213
23
256
```



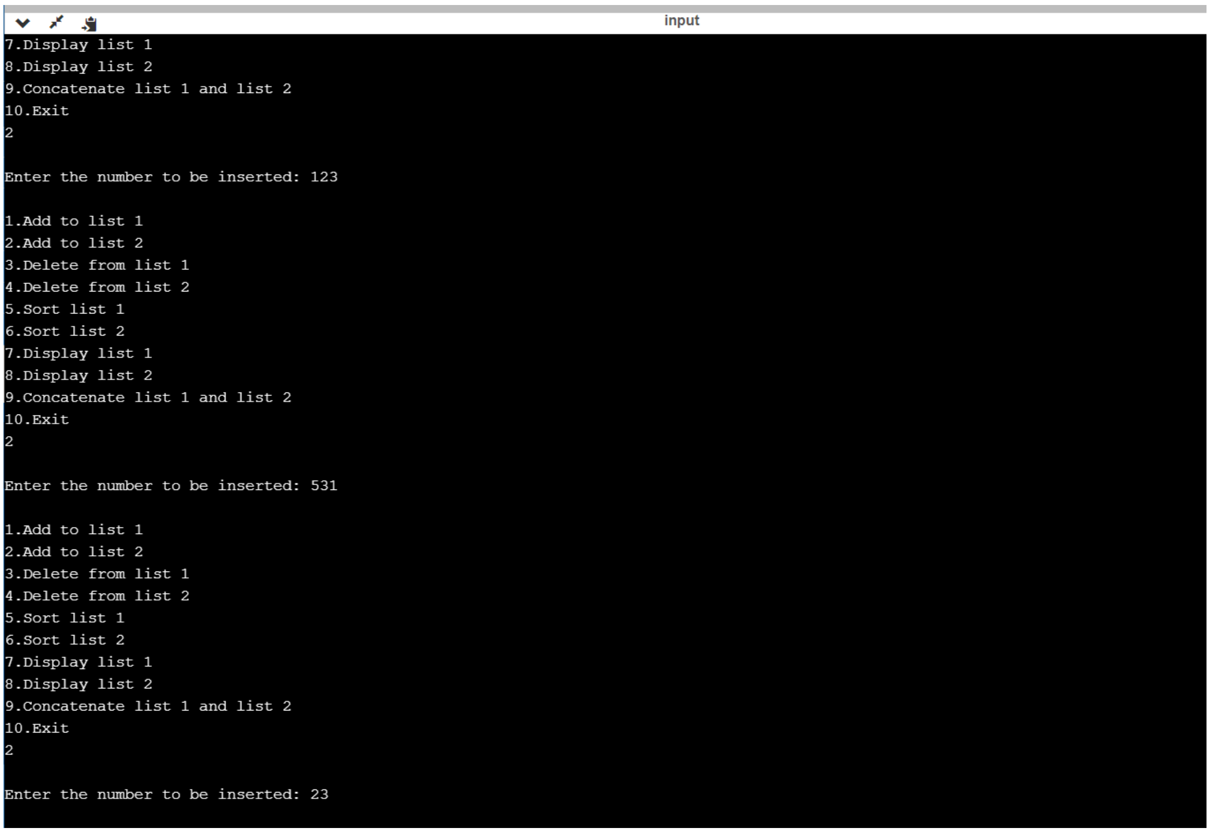
```
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
1

Enter the number to be inserted: 13

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
5

The contents in list 1 are-
11
12
13
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
```



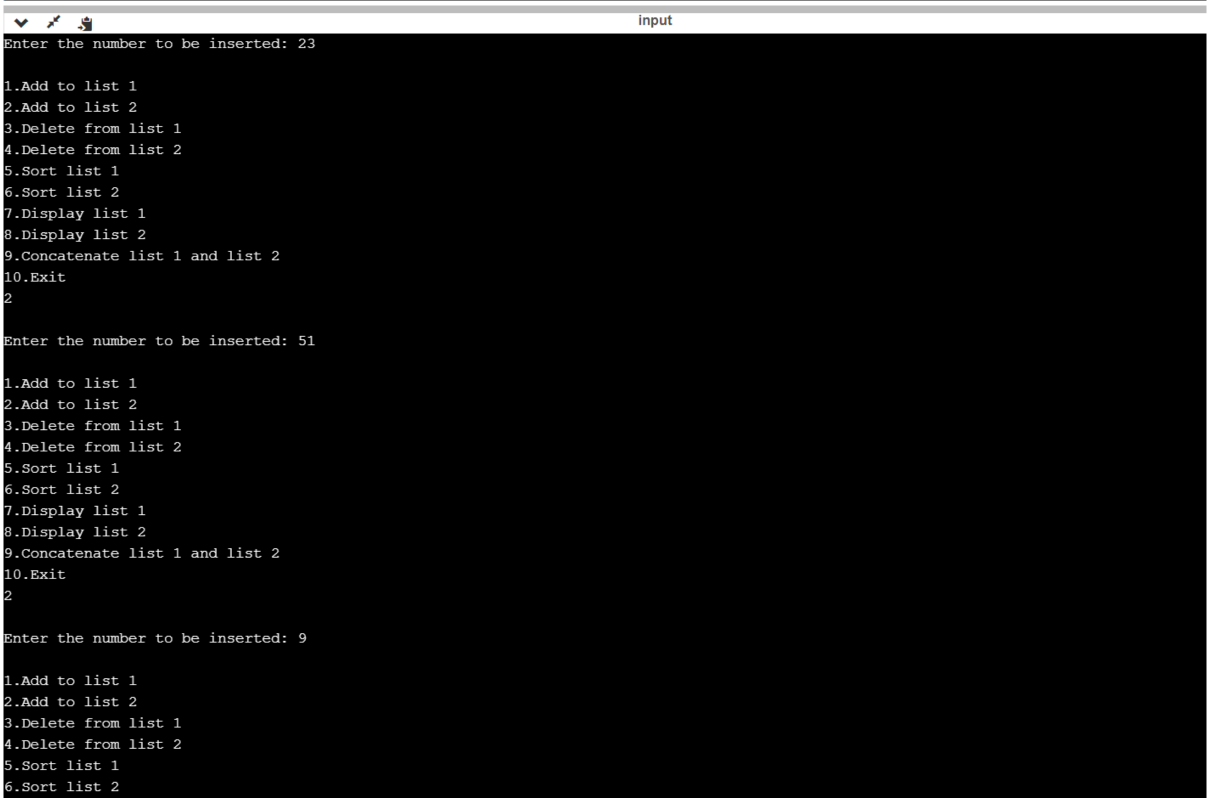
```
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
9
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
8.Sort list 2
9.Reverse list 1.
10.Reverse list 2
11.Concatenate list 1 and list 2
12.Exit
5

The contents in list 1 are-
13
12
11
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Display list 1
6.Display list 2
7.Sort list 1
```



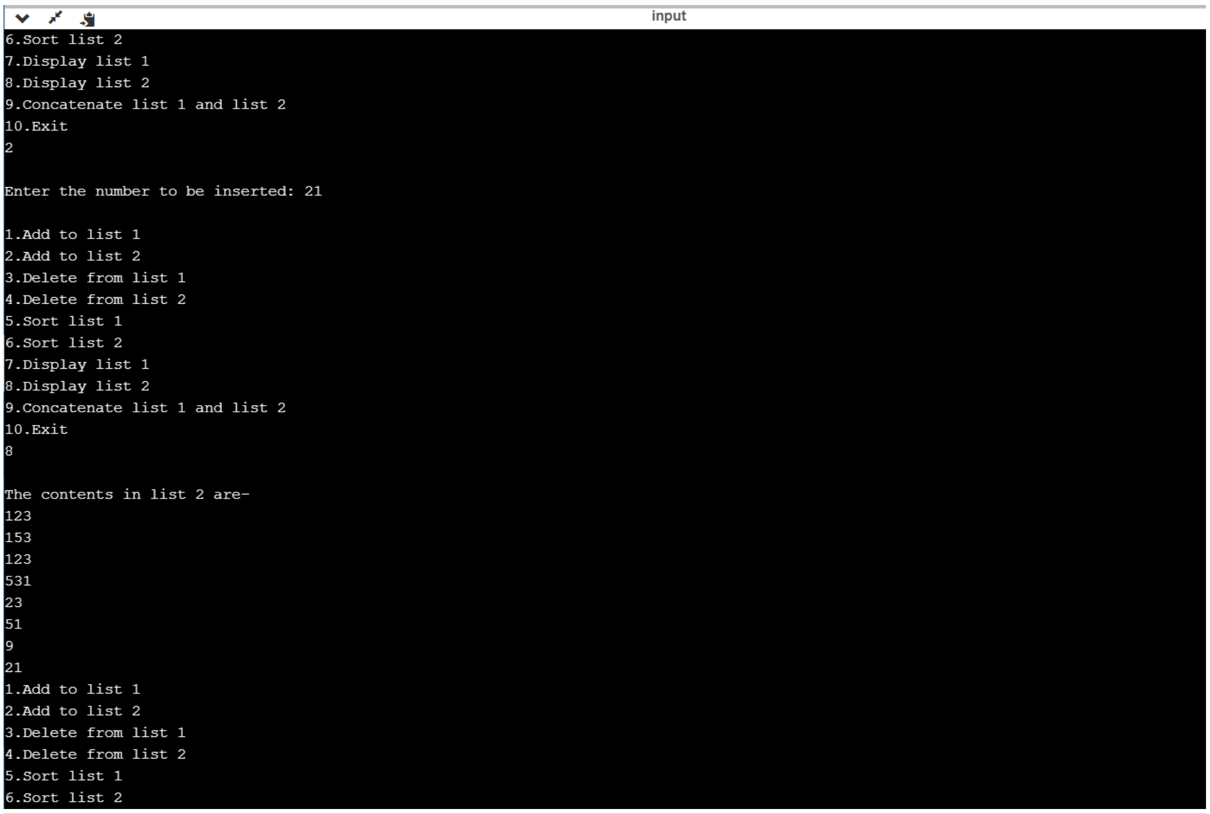
```
Choose any option-
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 123

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 153

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
```



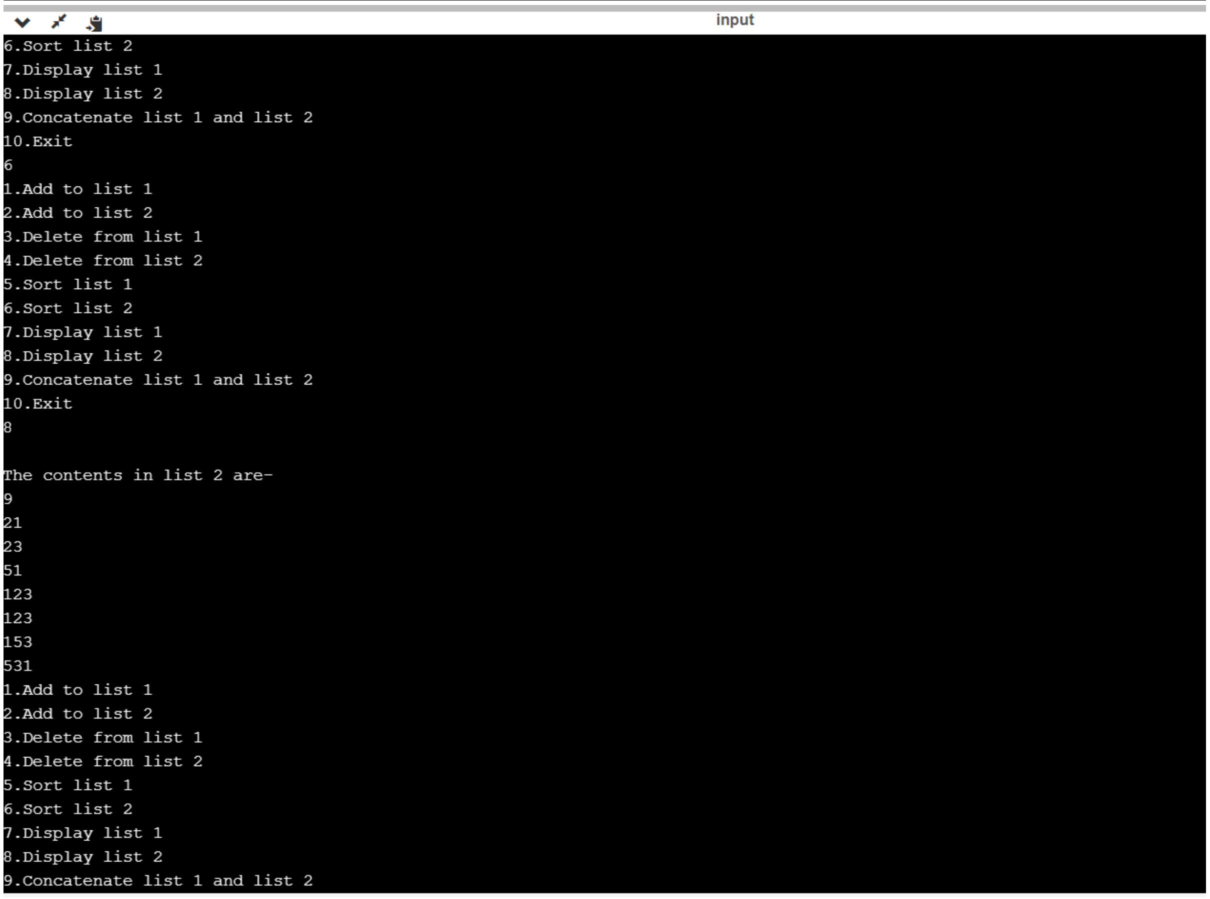
```
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 123

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 531

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 23
```



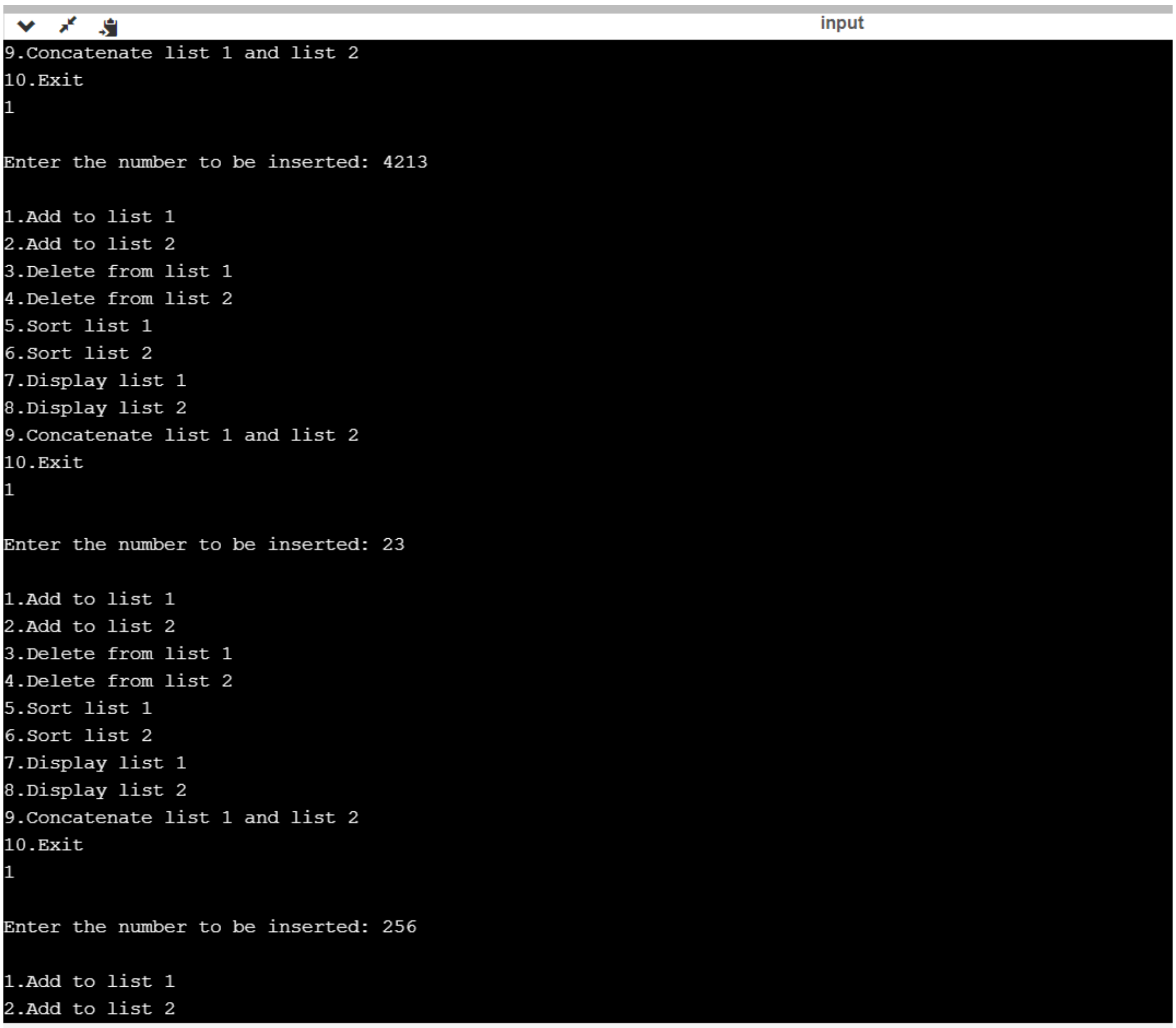
```
Enter the number to be inserted: 23

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 51

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 9

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
```



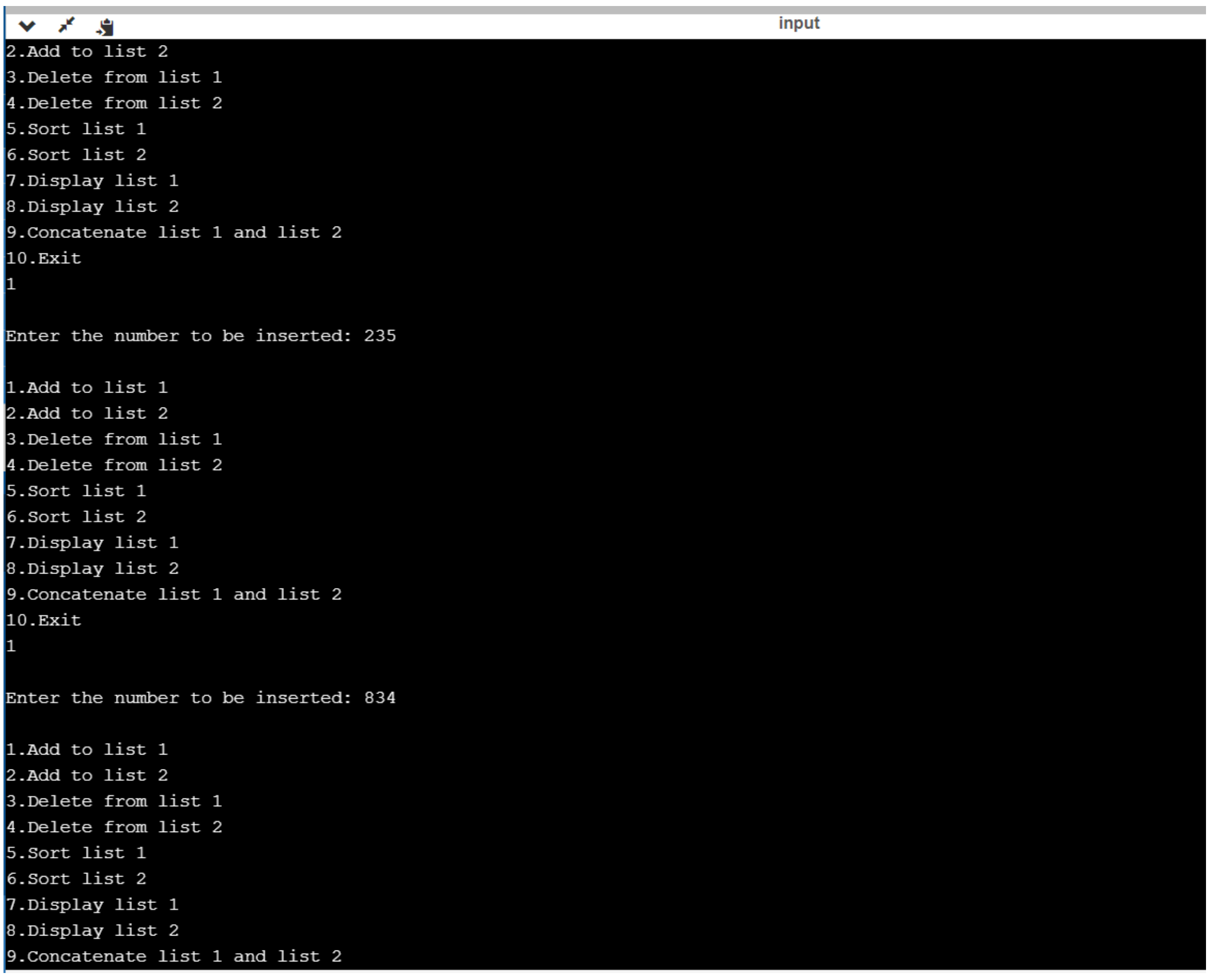
```
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
2

Enter the number to be inserted: 21

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
8

The contents in list 2 are-
123
153
123
531
23
51
9
21
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
```



```
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
6
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
8

The contents in list 2 are-
9
21
23
51
123
123
153
531
1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
```



```
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 4213

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 23

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 256

1.Add to list 1
2.Add to list 2
```



```
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 235

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
10.Exit
1

Enter the number to be inserted: 834

1.Add to list 1
2.Add to list 2
3.Delete from list 1
4.Delete from list 2
5.Sort list 1
6.Sort list 2
7.Display list 1
8.Display list 2
9.Concatenate list 1 and list 2
```

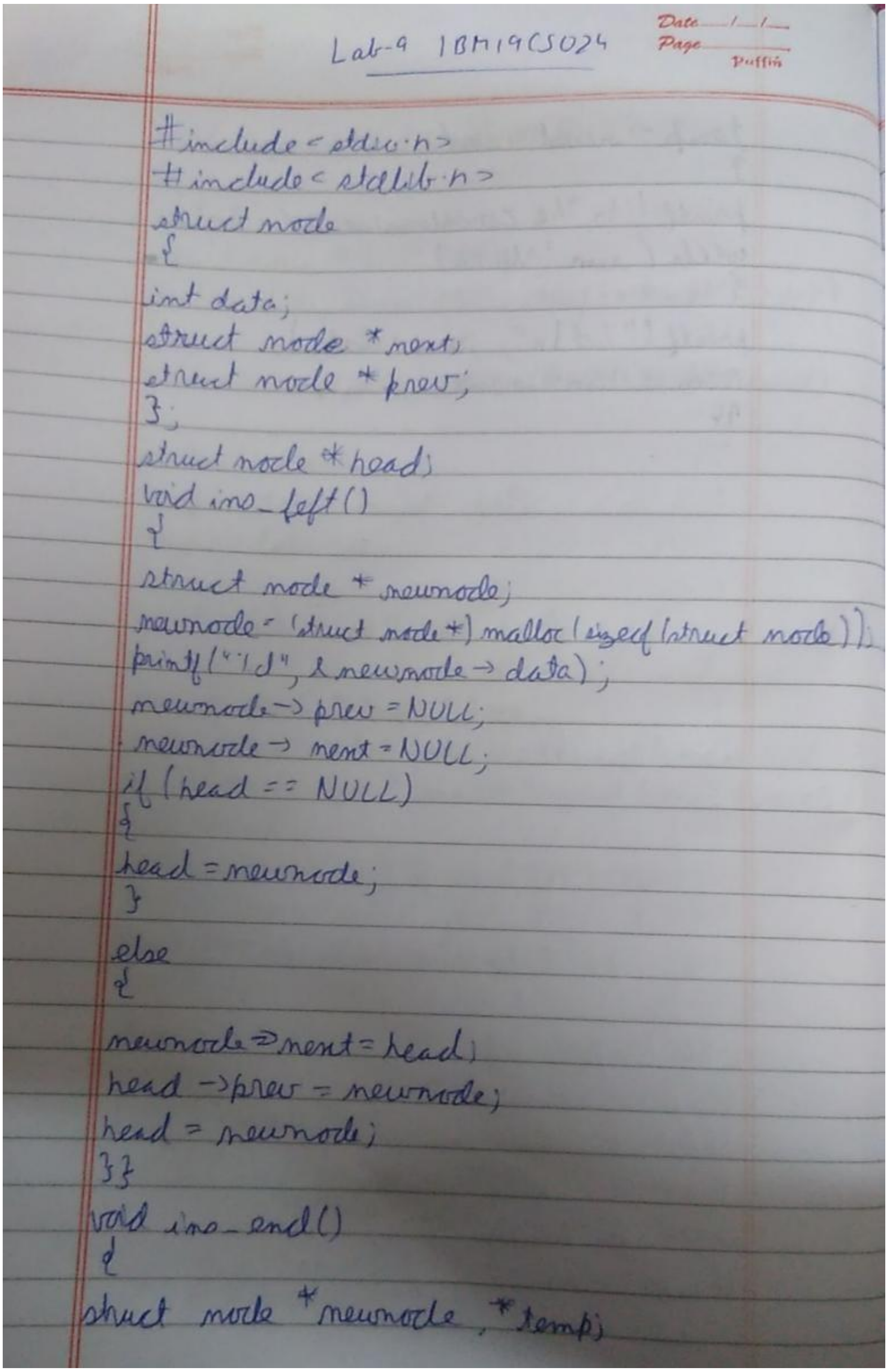
Lab Program 9

Source Code-

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

                    struct node
                    {
                        int data;
                        struct node *next;
                        struct node *prev;
                    };

                    struct node *head;
                    void ins_left()
                    {
                        struct node *newnode;
                        newnode=(struct node*)malloc(sizeof(struct node));
                        printf("Enter the value: ");
                        scanf("%d",&newnode->data);
                        newnode->prev=NULL;
                        newnode->next=NULL;
                        if(head==NULL)
                        {
                            head=newnode;
                        }
                        else
                        {
                            newnode->next=head;
                            head->prev=newnode;
                            head=newnode;
                        }
                    }
                    void ins_end()
                    {
                            struct node *newnode,*temp;
                            newnode=(struct node*)malloc(sizeof(struct node));
                            printf("Enter the number: ");
                            scanf("%d",&newnode->data);
                            newnode->next=NULL;
                            newnode->prev=NULL;
                            if(head==NULL)
                            {
                                    head=newnode;
                            }
```

```
        else
        {
                temp=head;
                while(temp->next!=NULL)
                temp=temp->next;
                temp->next=newnode;
                newnode->prev=temp;

        }

}
void ins_ran()
{
        int ele;
        struct node *newnode,*temp;
        printf("Enter the element in the list: ");
        scanf("%d",&ele);
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the new node data: ");
        scanf("%d",&newnode->data);
        newnode->next=NULL;
        newnode->prev=NULL;
    if(head==NULL)
    {
        printf("List is Empty/n");
        return;
    }
        temp=head;
        while(temp->data!=ele)
        {
               temp=temp->next;
               if(temp==NULL)
               {
                      printf("Element is  not in the list\n");
                      return;
               }
        }
        newnode->next=temp->next;
        temp->next=newnode;
        newnode->prev=temp;
        newnode->next->prev=newnode;
}
void del()
{
        struct node *temp;
        int ele;
    if(head==NULL)
    {
        printf("List is empty!!\n");
        return;
    }
        printf("Enter the element to be deleted: ");
        scanf("%d",&ele);
        temp=head;
        while(temp->data!=ele)
        {
               temp=temp->next;
               if(temp==NULL)
               {
                printf("Element not in list\n");
                break;
               }
        }
        if(temp==head)
        {
                head=head->next;
        }
        else if(temp->next==NULL)
        {
                      temp=temp->prev;
                      temp->next=NULL;
        }

        else
        {
                temp->prev->next=temp->next;
                temp->next->prev=temp->prev;
        }
}
void display()
{
        struct node *temp;
        temp=head;
        while(temp!=NULL)
        {
                printf("%d\t",temp->data);
                temp=temp->next;
        }
        printf("\n");
}
int main()
{
    int ch;
    printf("Choose from the following: \n");
    do{
        printf("1.Insert at the beginning\n2.Insert at the
end\n3.Insert in between\n4.Delete a number\n5.Display
list\n6.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:
            ins_left();
            break;
        case 2:
            ins_end();
            break;
        case 3:
            ins_ran();
            break;
        case 4:
            del();
            break;
        case 5:
            display();
            break;
        case 6:
            break;
        default:
            printf("Invalid choice!!/n");
        }
    }while(ch!=6);
    return 0;
}
```

Writeup-



Date __/__/__
Page ____
Puffin

Lab-9 1BM19CS024

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head;
void ins_left()
{
struct node *newnode;
newnode=(struct node*)malloc(sizeof(struct node));
printf("%d", &newnode->data);
newnode->prev=NULL;
newnode->next=NULL;
if(head==NULL)
{
head=newnode;
}
else
{
newnode=>next=head;
head->prev=newnode;
head=newnode;
}}
void ins_end()
{
struct node *newnode, *temp;
```

Date / /
Page
Puffin

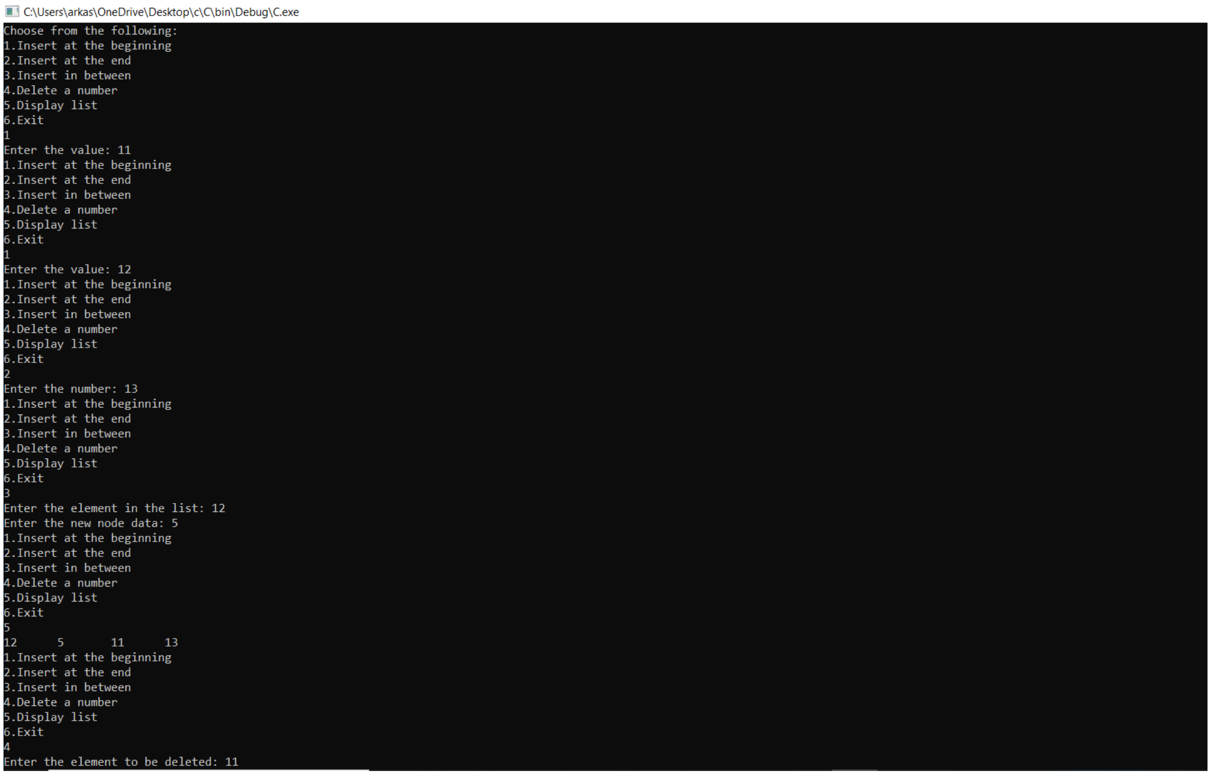
```
newnode = (struct node *) malloc (sizeof (struct node));
printf("Enter the number: ");
scanf("%d", & newnode -> data);
newnode -> next = NULL;
newnode -> prev = NULL;
if (head = = NULL)
{
head = newnode
}
else
{
temp = head;
while (temp -> next != NULL)
 temp = temp -> next;
 temp => next = newnode;
 newnode => prev = temp;
 }}
void ins_ran()
 {
int ele;
struct node *newnode, *temp;
printf("Enter element of list: ");
scanf("%d", & ele);
newnode = (struct node *) malloc (sizeof (struct node
printf ("Enter the new data: ");
scanf ("%d", & newnode -> data);
newnode -> next = NULL;
newnode -> prev = NULL;
if (head = = NULL)
 {
```

```
printf("List is empty \n");
return;
}
temp=head;
while(temp->data!=ele)
{
 temp=temp->next;
if(temp==NULL)
{
 printf("Element not in list \n");
return;
}}
newnode->next = temp->next
temp->next= newnode;
newnode->prev=temp;
~~me~~ newnode->next->prev = newnode;
}

void del()
{
struct node *temp;
int ele;
if(head==NULL)
{
printf("List is Empty \n");
return;
}
printf("Enter element to be deleted: ");
scanf("%d", &ele);
temp = head;
```

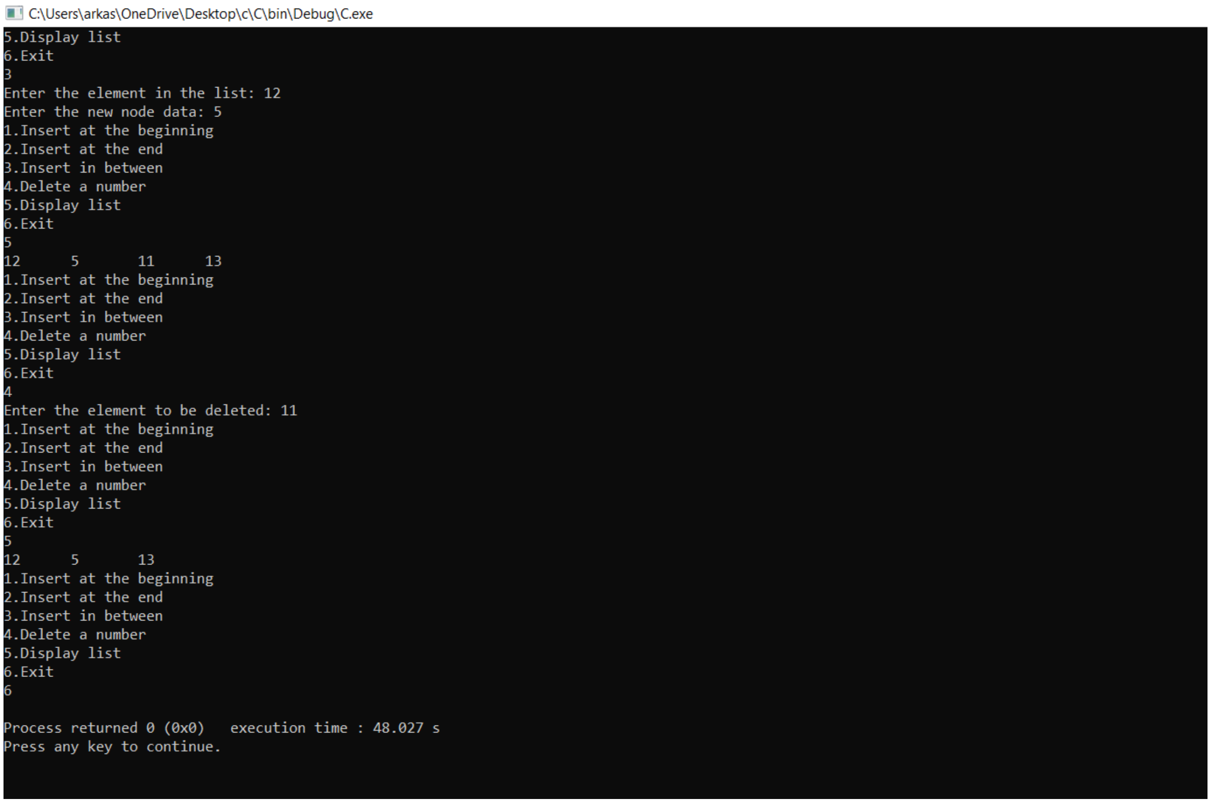
Date __/__/__
Page ____
Puffin

```
while (temp->data != ele)
{
 temp = temp->next;
if (temp == NULL)
{
printf("Element not in list\n");
break;
}}
if (temp == head)
 {
  head = head -> next;
}
else if (temp->next == NULL)
 {
temp = temp->prev;
temp->next = NULL;
}
else
{
temp -> prev -> next = temp -> next;
temp -> next -> prev = temp -> prev;
 }}
void display ()
{
struct node *temp;
temp = head;
while (temp != NULL)
{
printf("%d\t", temp->data); }
temp = temp -> next; } printf("\n"); }
```

ScreenShot-



```
C:\Users\arkas\OneDrive\Desktop\c\C\bin\Debug\C.exe
Choose from the following:
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
1
Enter the value: 11
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
1
Enter the value: 12
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
2
Enter the number: 13
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
3
Enter the element in the list: 12
Enter the new node data: 5
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
5
12      5       11      13
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
4
Enter the element to be deleted: 11
```



```
C:\Users\arkas\OneDrive\Desktop\c\C\bin\Debug\C.exe
5.Display list
6.Exit
3
Enter the element in the list: 12
Enter the new node data: 5
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
5
12      5       11      13
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
4
Enter the element to be deleted: 11
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
5
12      5       13
1.Insert at the beginning
2.Insert at the end
3.Insert in between
4.Delete a number
5.Display list
6.Exit
6

Process returned 0 (0x0)   execution time : 48.027 s
Press any key to continue.
```

Lab Program 10

Source Code-

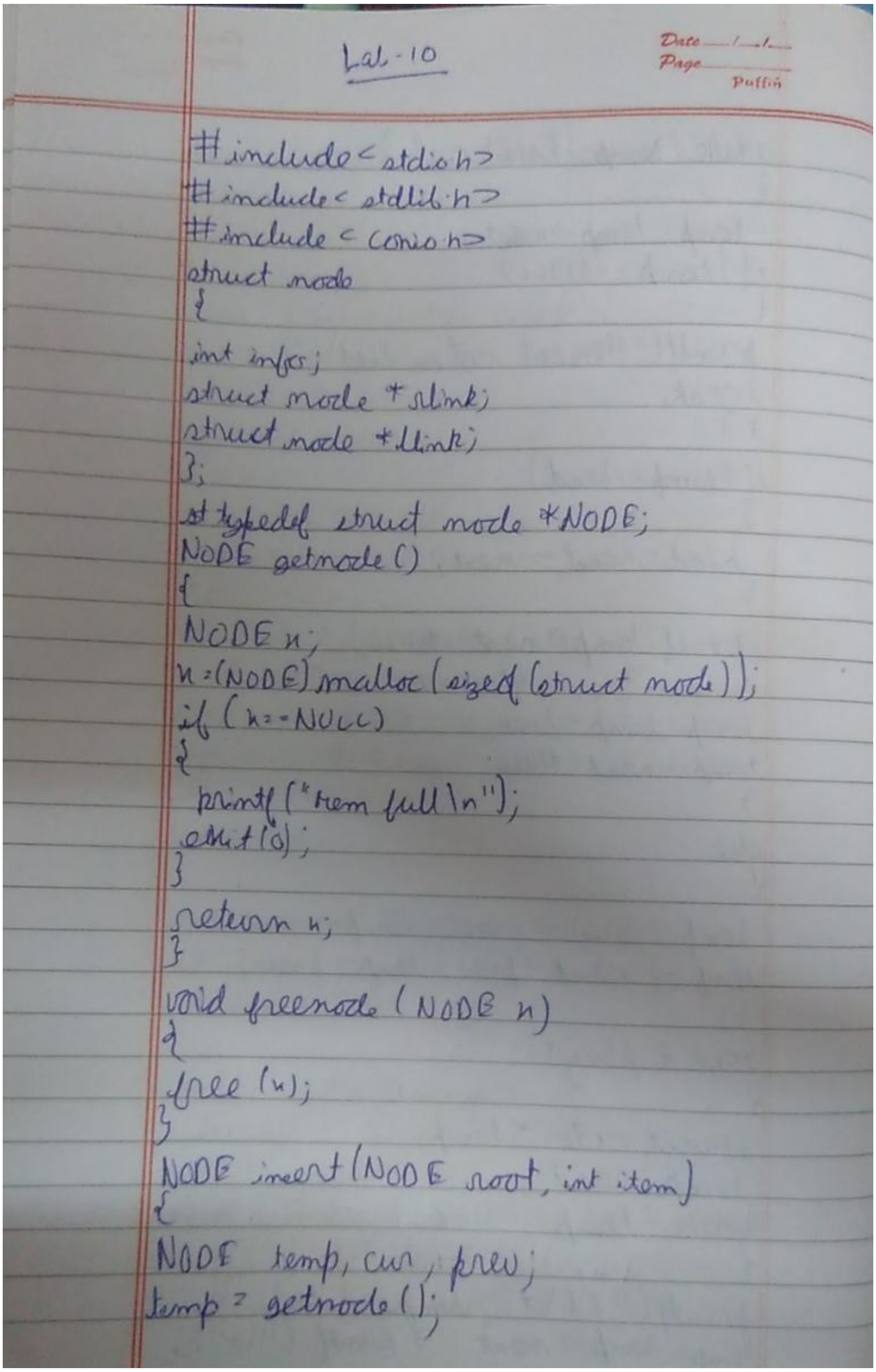
```
#include<stdio.h
>
#include<stdlib.h>
#include<conio.h>
struct node
 {
  int info;
  struct node *rlink;
  struct node *llink;
 };
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
```

```
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
         printf("  ");
   printf("%d\n",root->info);
        display(root->llink,i+1);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
```

```
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

void preorder(NODE root)
{
if(root!=NULL)
 {
  printf("%d\n",root->info);
  preorder(root->llink);
  preorder(root->rlink);
  }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {

  postorder(root->llink);
  postorder(root->rlink);
  printf("%d\n",root->info);
  }
 }
void inorder(NODE root)
{
if(root!=NULL)
 {

  inorder(root->llink);
  printf("%d\n",root->info);
  inorder(root->rlink);
  }
 }
void main()
```

```
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\
n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item\n");
              scanf("%d",&item);
              root=insert(root,item);
              break;
  case 2:display(root,0);
              break;
  case 3:preorder(root);
              break;
  case 4:postorder(root);
              break;
  case 5:inorder(root);
              break;
  case 6:printf("enter the item\n");
              scanf("%d",&item);
              root=delete(root,item);
              break;
  default:exit(0);
               break;
         }
       }
 }
```

Writeup-



Lab-10

Date __/__/__
Page ____
Puffin

```
#include<stdio.h>
#include< stdlib.h>
#include < conio.h>
struct node
 {
int info;
struct node *rlink;
struct node *llink;
};
~~st~~ typedef struct node *NODE;
NODE getnode ()
{
NODE n;
n =(NODE) malloc (sizeof (struct node));
if (n==NULL)
{
 printf ("mem full\n");
exit(0);
}
return n;
}
void freenode (NODE n)
{
free (n);
}
NODE insert(NODE root, int item)
{
NODE temp, cur, prev;
temp = getnode();
```

Date / /
Page
Puffin

```
temp->rlink=NULL;
temp->llink=NULL;
temp->infor=item;
if(root==NULL)
  return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->infor)?cur->llink:cur->rlink;
}
if(item<prev->infor)
    prev->llink=temp;
else
   prev->rlink=temp;
return root;
}
void display(NODE root, int i)
{
int j;
if(root!=NULL)
{
display(root->rlink, i+1);
for(j=0; j<i; j++).
      printf(" ");
      printf("%d\n", root->infor);
      display(root->llink, i+1);
}}
```

```
NODE delete (NODE root, int item)
{
NODE * cur, parent, q, suc;
if (root == NULL)
{
printf("Empty \n");
return root;
}
parent = NULL;
cur = root;
while (cur != NULL && item != cur->info)
{
parent = cur;
cur = (item < cur->info)? cur->llink: cur->rlink;
}
if (cur == NULL)
{
printf("not found \n");
return root;
}
if (cur->llink == NULL)
    q = cur->rlink;
else if (cur -> rlink == NULL)
    q = cur->llink;
else
{
suc = cur->rlink;
while (suc->llink != NULL)
    suc = suc->llink;
```

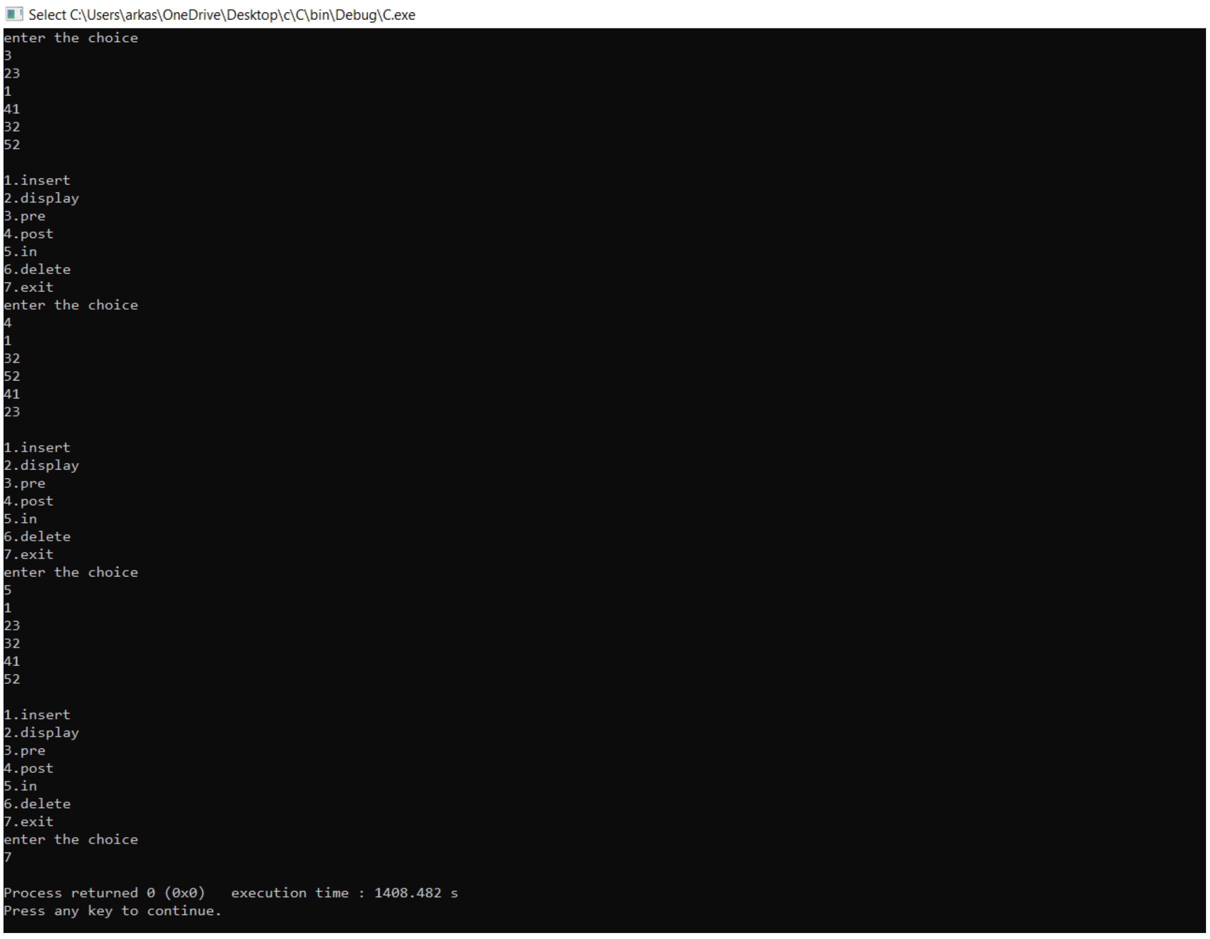
Date __/__/__
Page ____

```
    suc->llink = cur->llink;
    q = cur->rlink;
}
if(parent==NULL)
    return q;
if(cur== parent->link;
q = parent->llink = q;
else
    parent->rlink=q;
freenode(cur);
return root;
}
void preorder(NODE root)
{
 if(root!=NULL)
 {
   printf("%d\n", root->info);
   preorder(root->llink);
   preorder(root->rlink);
 }}
void postorder(NODE root)
{
 if(root!=NULL)
 {
   ~~printf~~( postorder(root->llink);
   postorder(root->rlink);
  printf("%d\n", root->info);
 }
}
```

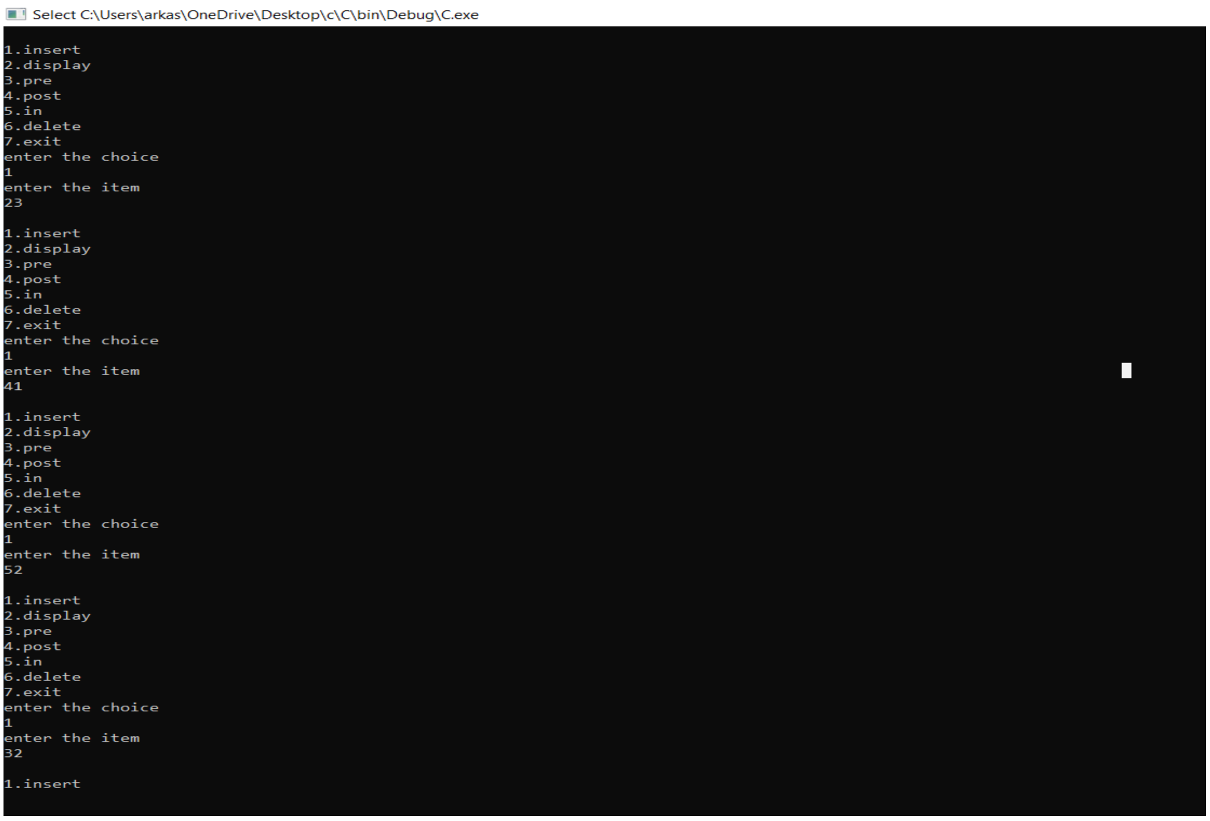
Date __/__/__
Page ______
Puffin

```
a. void inorder (NODE root)
{
if (root != NULL)
{
 inorder ( root -> llink);
 printf ("%d\n", root -> info);
inorder ( root -> rlink);
}
}
```

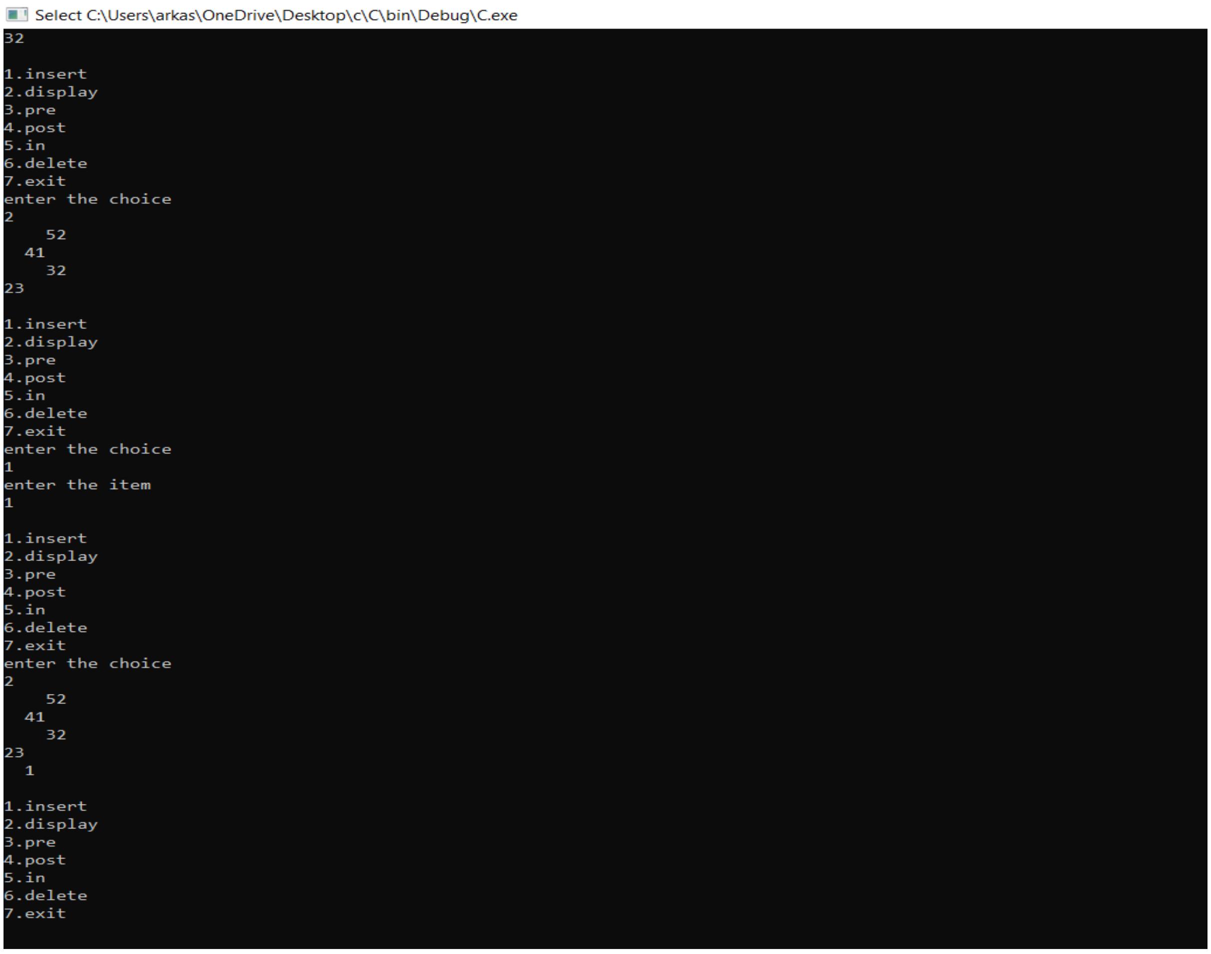
ScreenShot-



```
enter the choice
3
23
1
41
32
52

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
1
32
52
41
23

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
1
23
32
41
52

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
7

Process returned 0 (0x0)   execution time : 1408.482 s
Press any key to continue.
```


Select C:\Users\arkas\OneDrive\Desktop\c\C\bin\Debug\C.exe

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
23

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
41

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
52

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
32

1.insert
```


Select C:\Users\arkas\OneDrive\Desktop\c\C\bin\Debug\C.exe

```
32

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    52
  41
    32
23

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
1

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    52
  41
    32
23
  1

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
```