VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

SHREEHARI KULKARNI (1BM19CS153)

Under the Guidance of

Prof. POOJA S
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Sep-2020 to Jan-2021

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the LAB RECORD carried out by **SHREEHARI KULKARNI(1BM19CS153)** who is the bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide
Prof. POOJA S
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner — Signature with date

1._____ _____

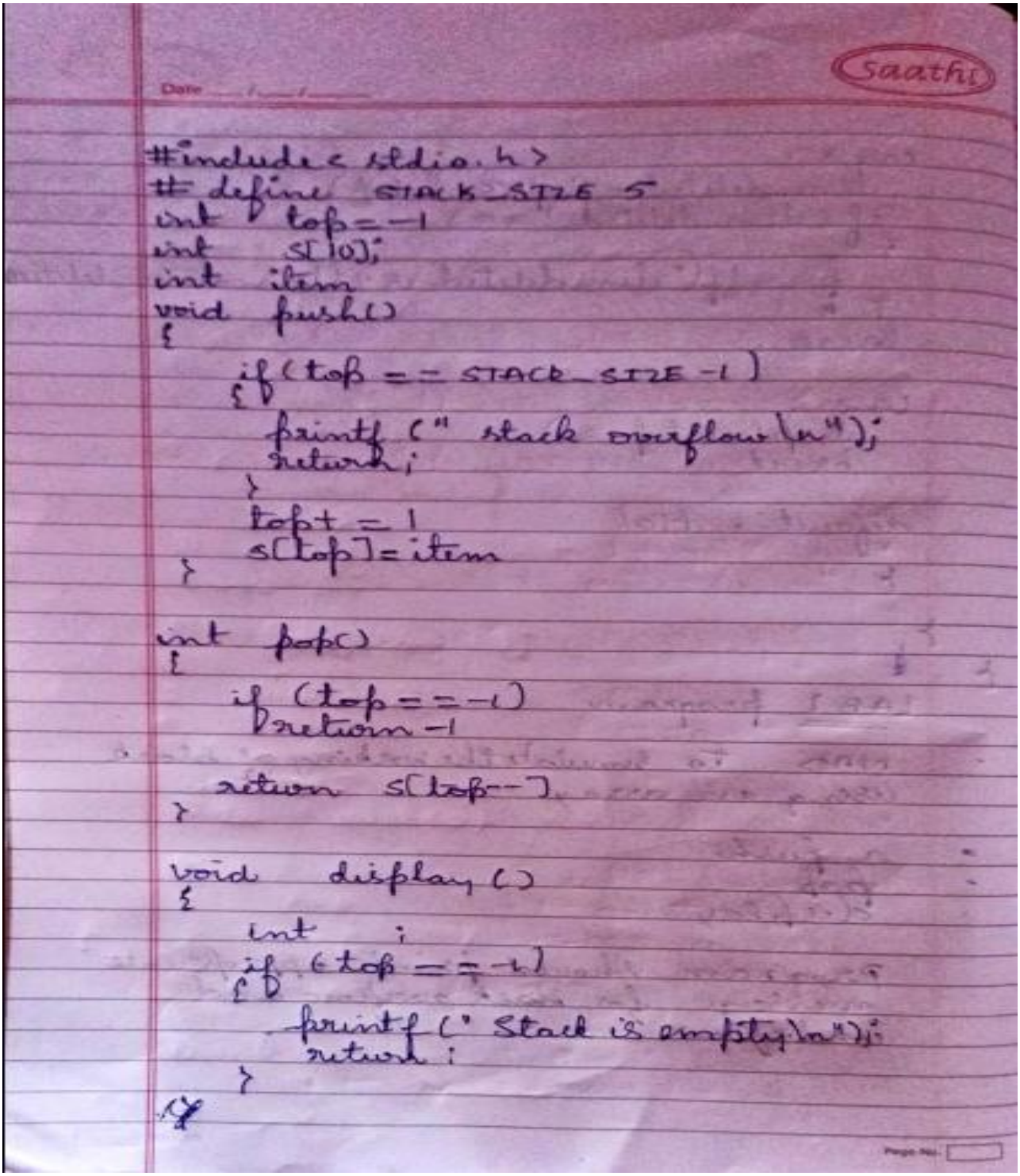
2. _____ _____

INDEX

1: Write a program to simulate the working of stack using an array with the following:

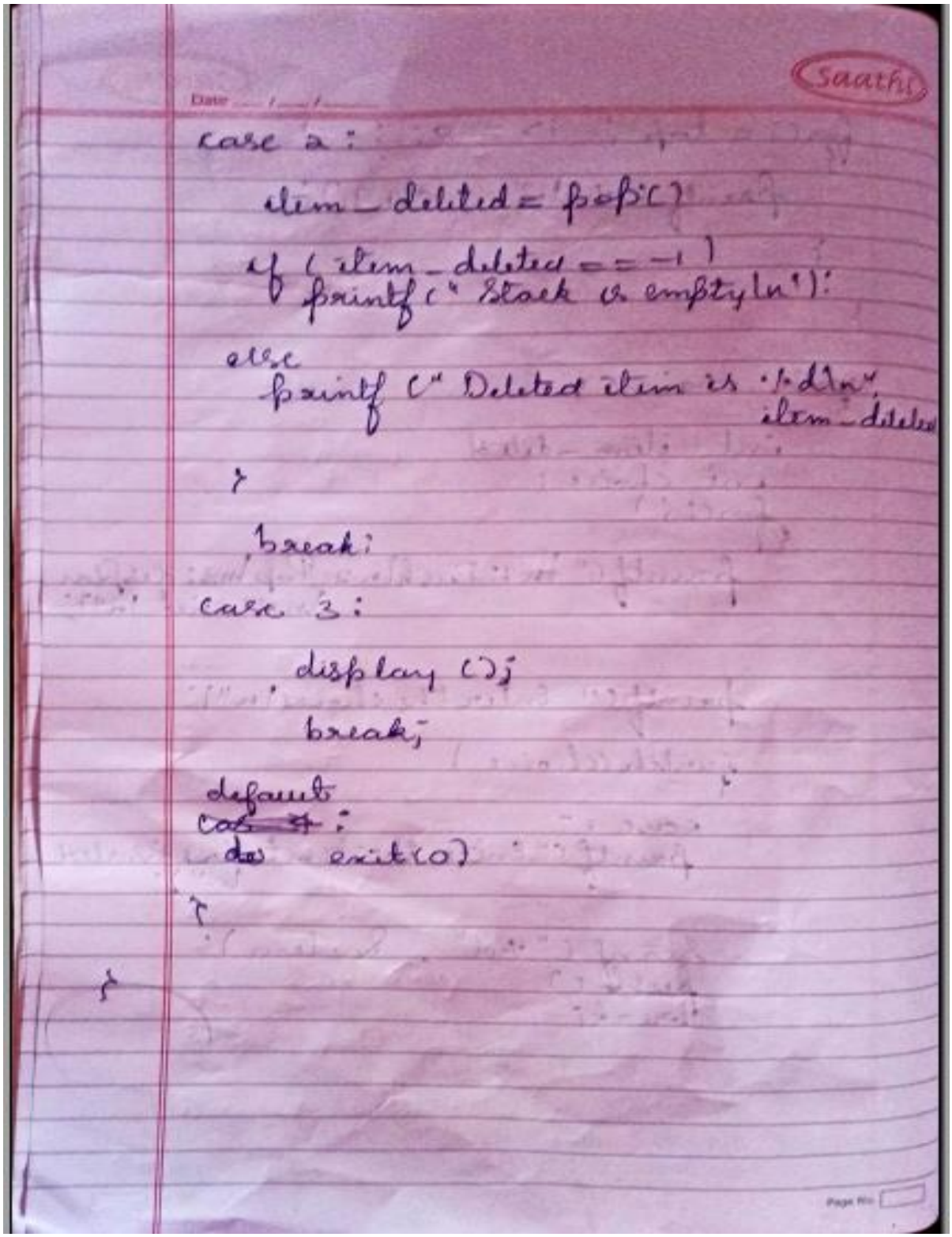
a) Push b) Pop c) Display

The program should print appropriate messages for stack overflow, stack underflow



```
#include<stdio.h>
#define STACK_SIZE 5
int top=-1
int s[10];
int item
void push()
{
    if(top == STACK_SIZE-1)
    {
      printf(" stack overflow\n");
      return;
    }
    top+ = 1
    s[top]=item
}

int pop()
{
    if (top==-1)
    return -1

  return s[top--]
}

void display()
{
    int ;
    if (top==-1)
    {
      printf(" Stack is empty\n");
      return;
    }
}
```

```
for(i = top ; i >= 0; i--)
{
    printf("%d\n", s[i]);
}
}

void main()
{
    int item_deleted
    int choice;
    for(;;)
    {
        printf("\n1: Push\n2: Pop\n3: display
                \n4: exit\n");

        printf(" Enter the choice\n");
        switch(choice)
        {
            case 1:
            printf(" Enter the item to be inserted
                    \n");

            scanf("%d", &item);
            push()
            break;
```



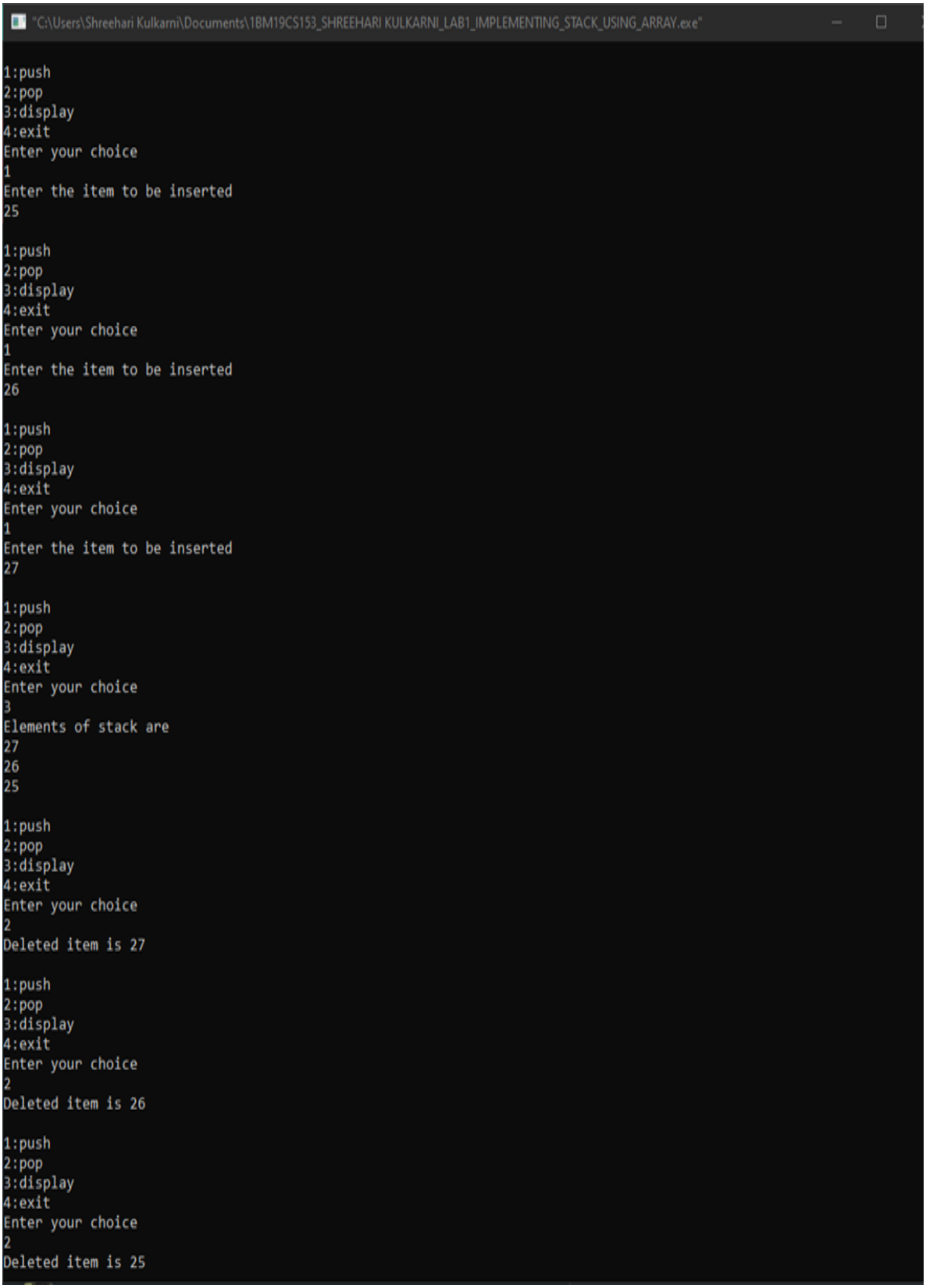
```
case 2:
    item_deleted = pop();
    if (item_deleted == -1)
    printf("Stack is empty\n");
    else
    printf("Deleted item is %d\n",
                                item_deleted);
    }
    break;
case 3:
    display();
    break;
default
case 4:
    exit(0)
}
}
```

OUTPUT:

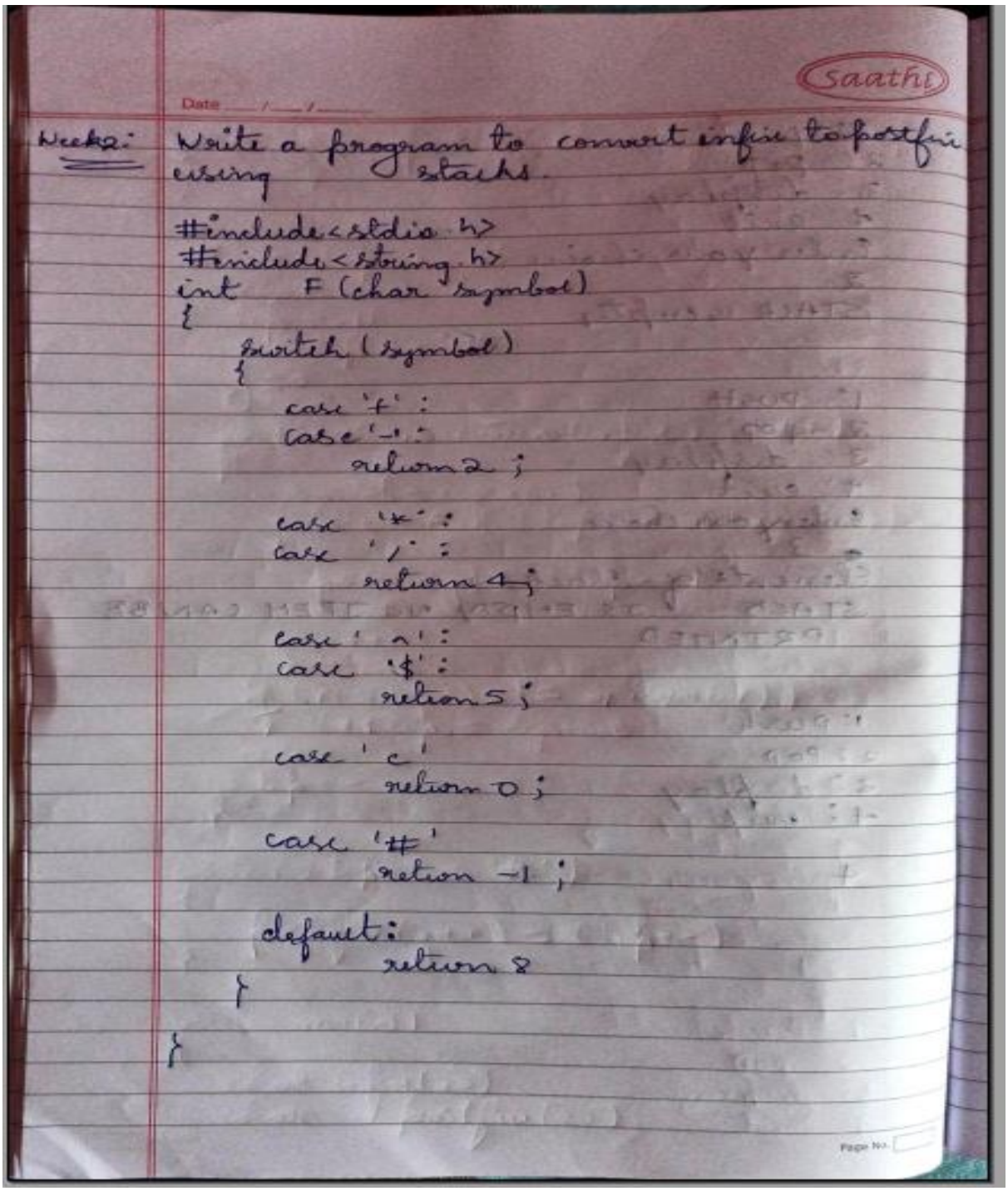
```
1: Push
2: POP
3: display
4: exit

Enter your choice
1
Enter the item to be inserted
25

1: Push
2: POP
3: display
4: exit
Enter your choice
1
enter the item to be insated
26

1: Push
2: POP
3: display
4: exit
Enter your choice
1
Enter the item to be inserted
27

1: Push
2: POP
3: display
4: exit
```

OUTPUT:


"C:\Users\Shreehari Kulkarni\Documents\1BM19CS153_SHREEHARI KULKARNI_LAB1_IMPLEMENTING_STACK_USING_ARRAY.exe"
1:push
2:pop
3:display
4:exit
Enter your choice
3
Stack is empty
1:push
2:pop
3:display
4:exit
Enter your choice
3
Elements of stack are
STACK IS EMPTY NO ITEM CAN BE PRINTED
1:push
2:pop
3:display
4:exit
Enter your choice
4
Process returned 0 (0x0) execution time : 664.350 s
Press any key to continue.


"C:\Users\Shreehari Kulkarni\Documents\1BM19CS153_SHREEHARI KULKARNI_LAB1_IMPLEMENTING_STACK_USING_ARRAY.exe"
1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
25
1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
26
1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
27
1:push
2:pop
3:display
4:exit
Enter your choice
3
Elements of stack are
27
26
25
1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 27
1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 26
1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 25

2: WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

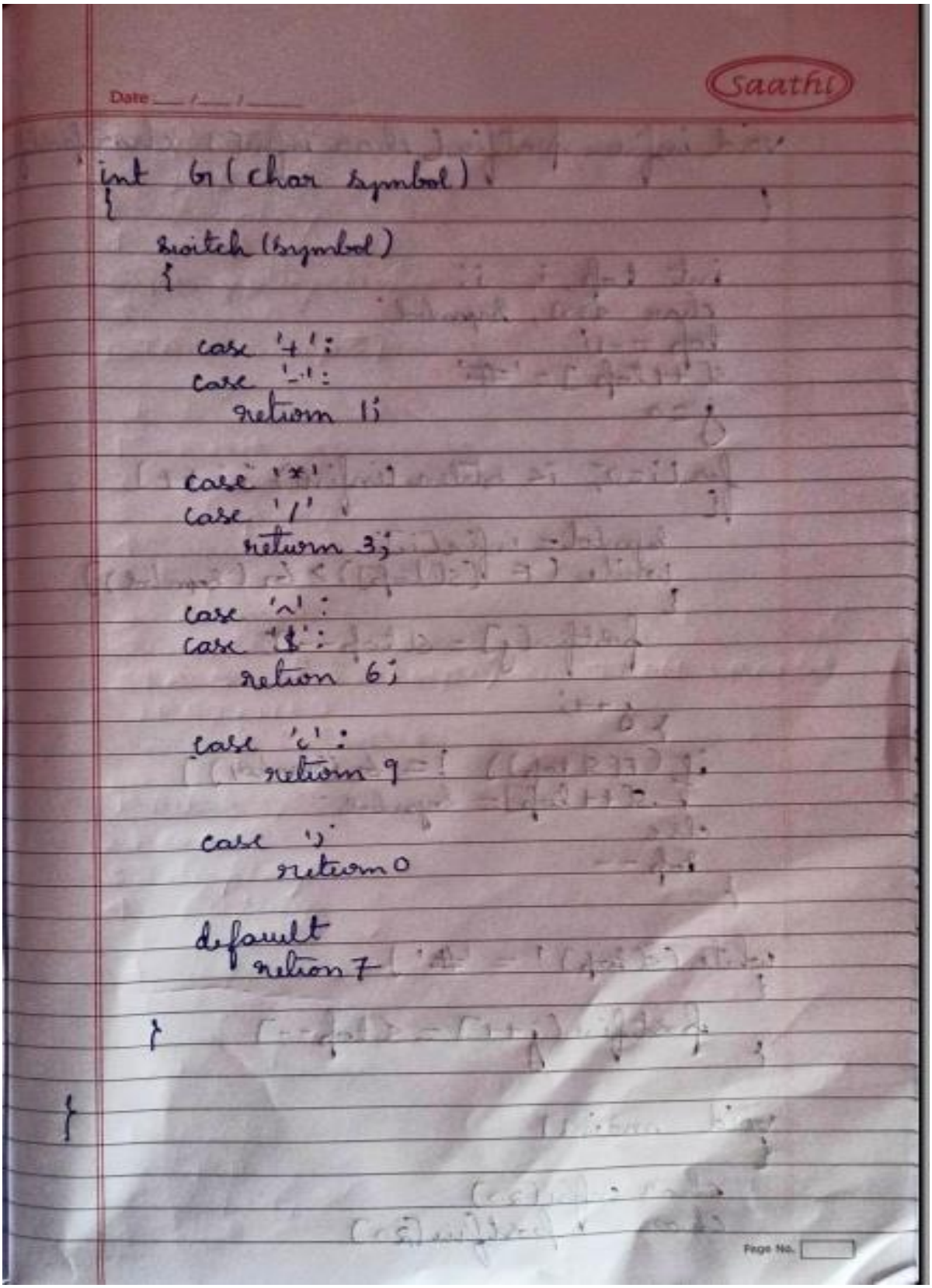


Saathi

Date

Aim: Write a program to convert infix to postfix using stacks.

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-':
            return 2;

        case '*':
        case '/':
            return 4;

        case '^':
        case '$':
            return 5;

        case '(':
            return 0;

        case '#'
            return -1;

        default:
            return 8
    }

}
```



```
int br(char symbol)
{
   switch(symbol)
   {
      case '+':
      case '-':
         return 1;

      case '*':
      case '/':
         return 3;

      case '^':
      case '$':
         return 6;

      case '(':
         return 9;

      case ')':
         return 0

      default
         return 7

   }
}
```

Saathi

Date / /

```
void infix postfix( char infix[], char postfix[])
{
    int top, i, j;
    char s[30], Symbol;
    top = -1;
    s[++top] = '#';
    j = 0;
    for(i=0; i< strlen(infix); i++)
    {
        Symbol = infix[i];
        while ( F[s[top]] > G[Symbol])
        {
            postfix[j] = s[top--];
            j++;
        }
        if ( F[s[top]] != G[Symbol])
            s[++top] = Symbol;
        else
            top--;
    }
    while (s[top] != '#')
    {
        postfix[j++] = s[top--];
    }

void main()
{
    char infix[20];
    char postfix[20];
```

Date ___/___/_____

Saathi

(1)

```
printf("Enter the valid expression\n");
scanf("%s", infix);
infix_postfix(infix, postfix);
printf("postfix is \n");
printf("%s\n", postfix);
}
```

OUTPUT

Enter the valid Expression

(A + (B - C) * D)

Post fix Expression is

ABC - D* + .

Page No.

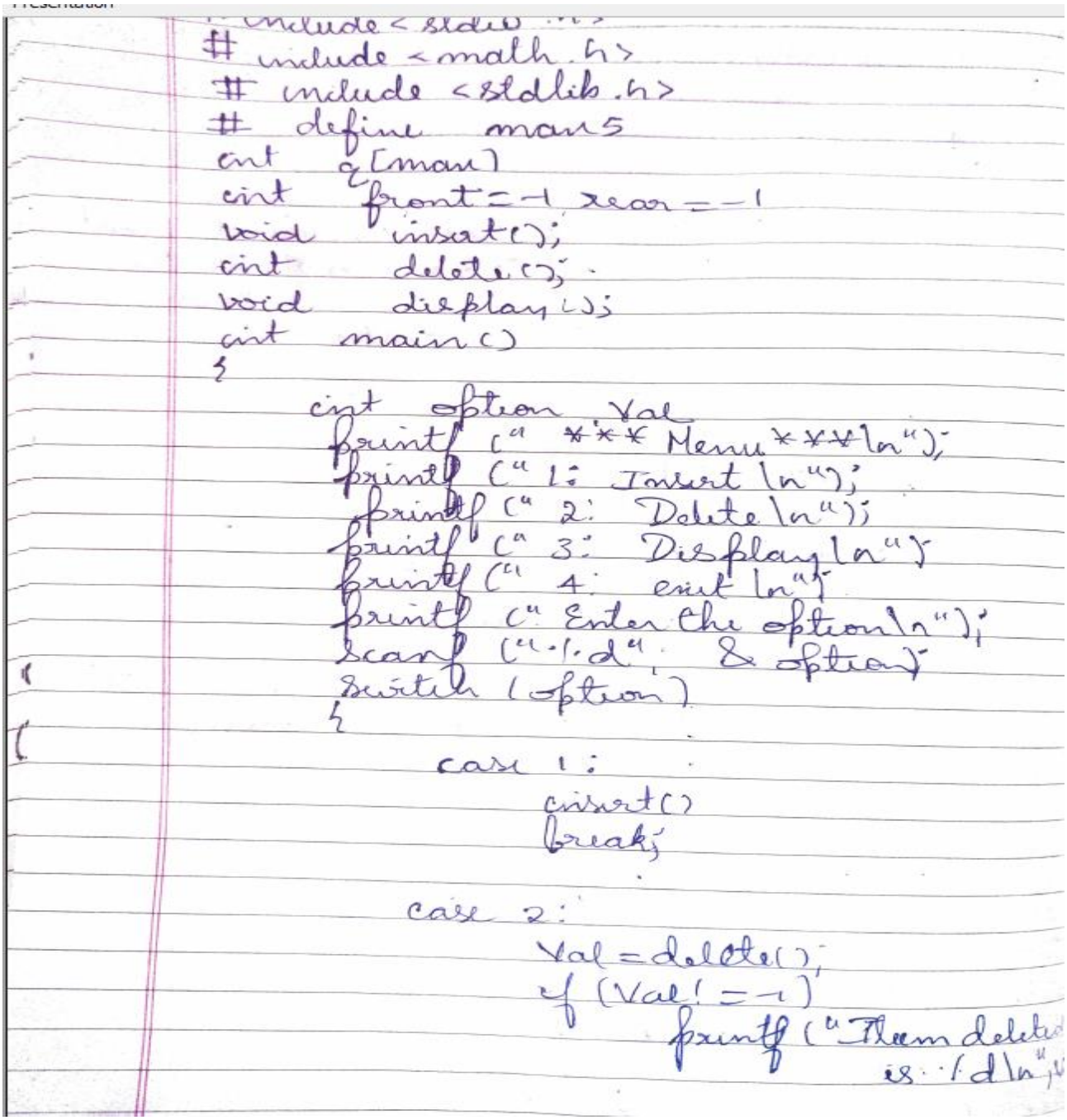
OUTPUT:


"C:\Users\Shreehari Kulkarni\Desktop\infix_postfix_modified.exe"
Enter the valid Expression
(a+b)*(c-d)
Valid Expression Continue
Postfix Expression is
ab+cd-*
Process returned 0 (0x0) execution time : 48.132 s
Press any key to continue.

3: WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue overflow Conditions



```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define max 5
int q[max]
int front=-1, rear=-1
void insert();
int delete();
void display();
int main()
{
    int option, val
    printf("*** Menu ***\n");
    printf("1: Insert\n");
    printf("2: Delete\n");
    printf("3: Display\n");
    printf("4: exit\n");
    printf("Enter the option\n");
    scanf("%d", &option);
    switch(option)
    {
        case 1:
            insert()
            break;

        case 2:
            val = delete();
            if(val != -1)
                printf("Item deleted
                is: %d\n",
```

```
        break;

    case 3:
            display();
            break;

    case 4:
            exit(0)
    }

} while(option!=5);
  return 0

}

void insert()
{
    int num;
    printf("Enter the item to be inserted \n");
    scanf("%d", &num);
    if (rear == max-1)
    {
        printf("overflow\n");
    }
    if (front==-1 && rear==-1)
    {
        front=0;
        rear=0;
        q[rear]=num;
    }
    else
    {
        rear++;
        q[rear]=num;
```

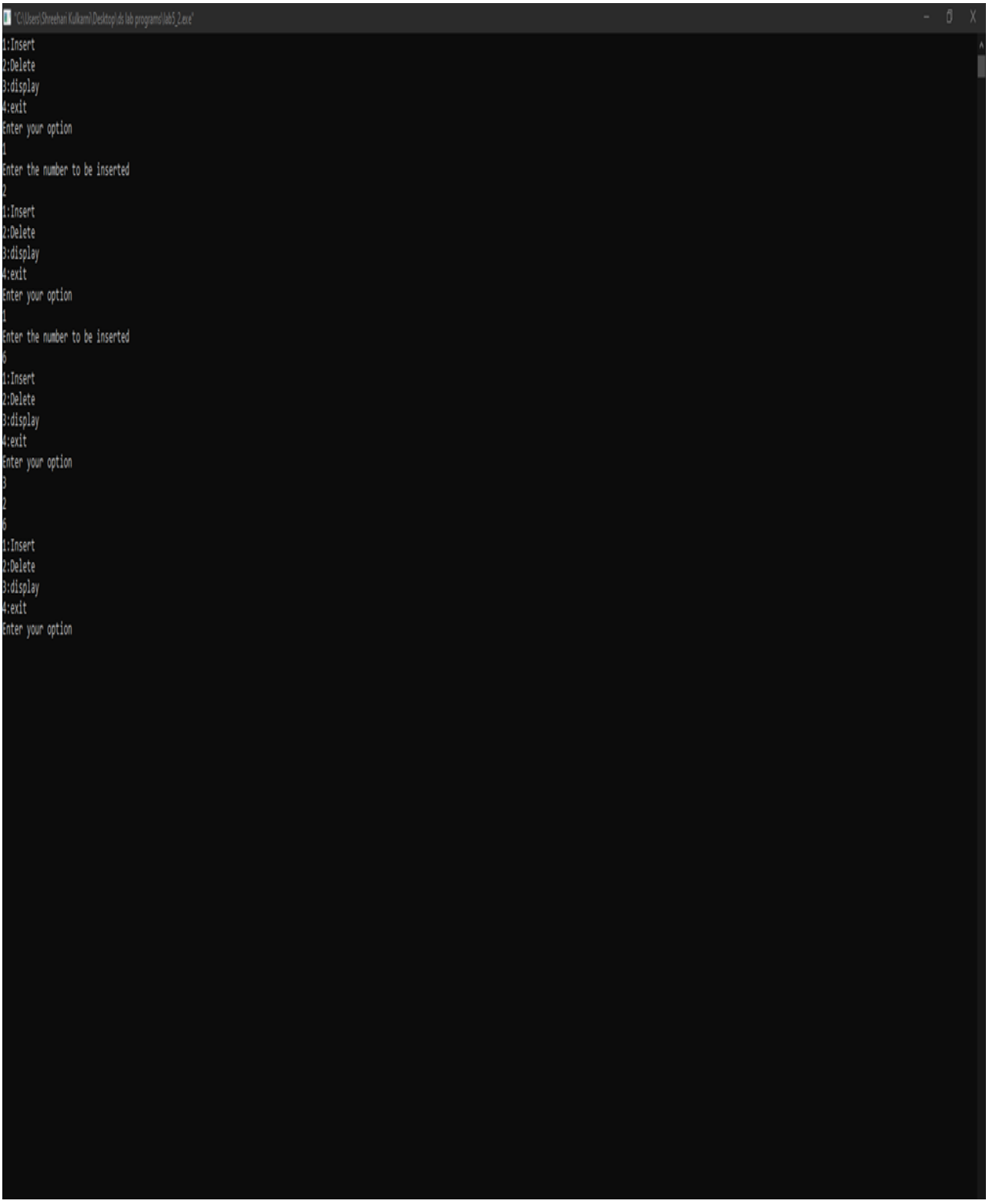
```
int delete()
{
    int val
    if( front==-1 || front>rear)
    {
        printf("UNDERFLOW\n");
        return -1;
    }
    else
    {
        val = q[front]
        front++;
        if( front>rear)
            front=-1;
            rear=-1;
        }
        return val
    }
}

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

OUTPUT

1: Insert
2: Delete
3: Display
Enter your option
1
Enter the number to inserted
2

1: insert
2: Delete
3: Display
4: exit
Enter your option
1
Enter the number to be inserted
6
1: insert
2: Delete
3: display
4: exit
Enter your option
3

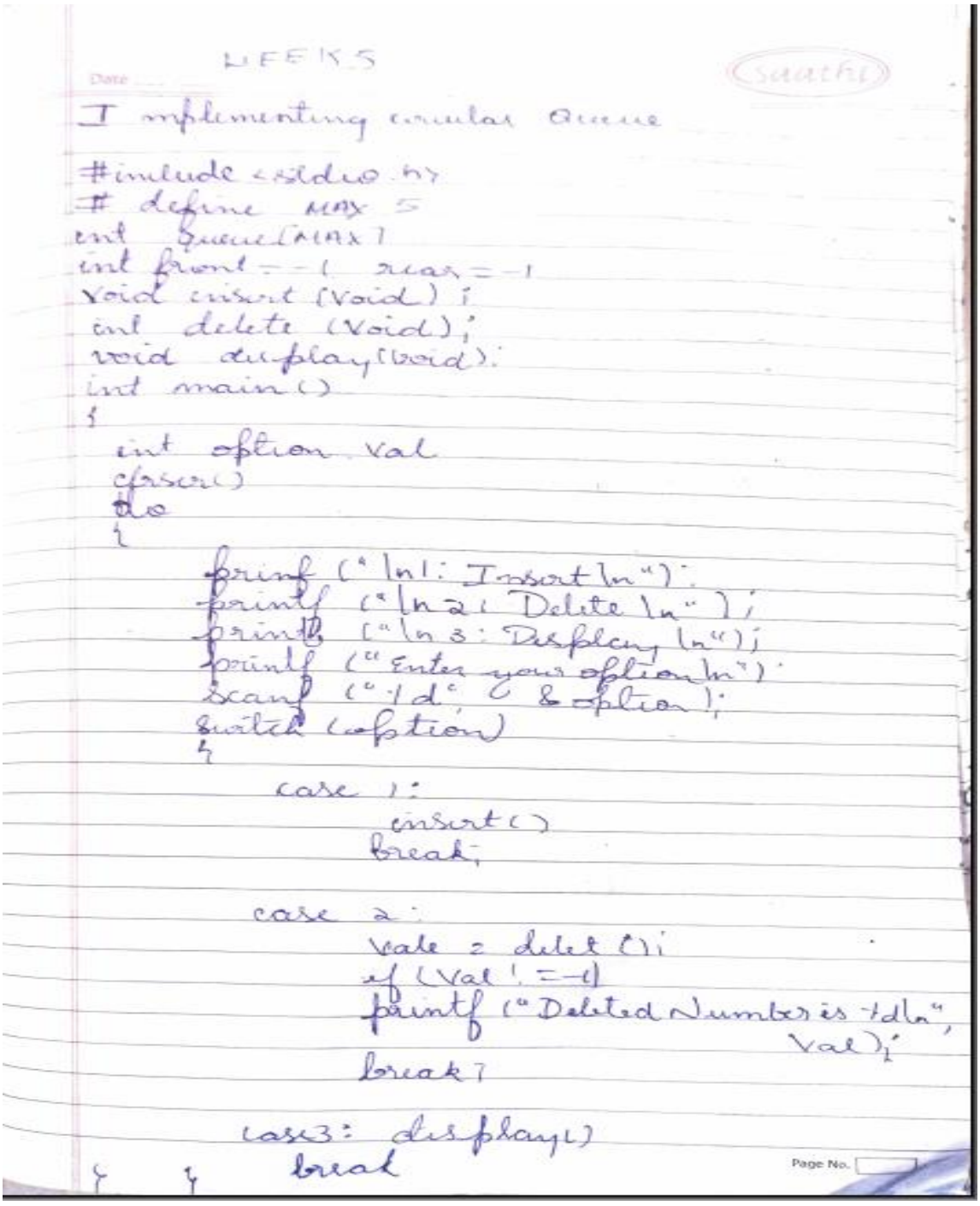
2


"C:\Users\Shreehari Kulkarni\Desktop\ds lab programs\lab5_2.exe"
1:Insert
2:Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
2
1:Insert
2:Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
5
1:Insert
2:Delete
3:display
4:exit
Enter your option
3
2
5
1:Insert
2:Delete
3:display
4:exit
Enter your option

4: WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue overflow Conditions



```
WEEK 5
Date
I mplementing circular Queue

#include <stdio.h>
# define MAX 5
int queue[MAX]
int front = -1 rear = -1
void insert (void);
int delete (void);
void display(void);
int main()
{
  int option val
  clrscr()
  do
  {
      printf ("\n1: Insert\n");
      printf ("\n2: Delete\n");
      printf ("\n3: Display\n");
      printf ("Enter your option\n");
      scanf ("%d", & option);
      switch (option)
      {
          case 1:
                insert()
                break;

          case 2:
                vale = delet();
                if (Val != -1)
                printf ("Deleted Number is %d\n",
                                    Val);
                break;

          case3: display()
                break
      }
  }
```

Saathi

Date ___/___/___

```
{ while(option != 5)
  return 0

void insert()
{
   int num;
   printf("Enter the number to be added\n");
   scanf("%d", &num);
   if (front == 0 && rear == MAX-1)
        printf("overflow\n");
   else if (front == -1 && rear == -1)
   {
        front = rear = 0
        queue[rear] = num
   }
   else if (rear == MAX-1 && front != 0)
   {
        rear = 0
        queue[rear] = num;
   }
   else
   {
      rear ++
      queue[rear] = num;
   }
}
```

```
int delete()
{
  int val
  if (front==-1 && rear==-1)
  {
     printf("\n Underflow");
     return -1
  }
  val = queue[front]
  if (front == rear)
  {
     front = rear = -1
  }
  else
  {
     if (front = Max-1)
        front = 0
     else
        front++
  }
return val
}
void display()
{
  int i
  printf("\n");
  if (front == -1 && rear == -1)
     printf("\n Queue is empty");
  else
  {
     if (front < rear)
```

Saathi

Date __/__/____

```
{
    for(int i = front; i<max; i++)
        printf("%d", queue[i]);

    for i=0;

}
else
{
    for(int i = front; i<Max; i++)
        printf("%d", queue[i]);

    for(int j=0; i<= rear; i++)
        printf("%d\n", queue[i]);

}
}
}
```

Saathi

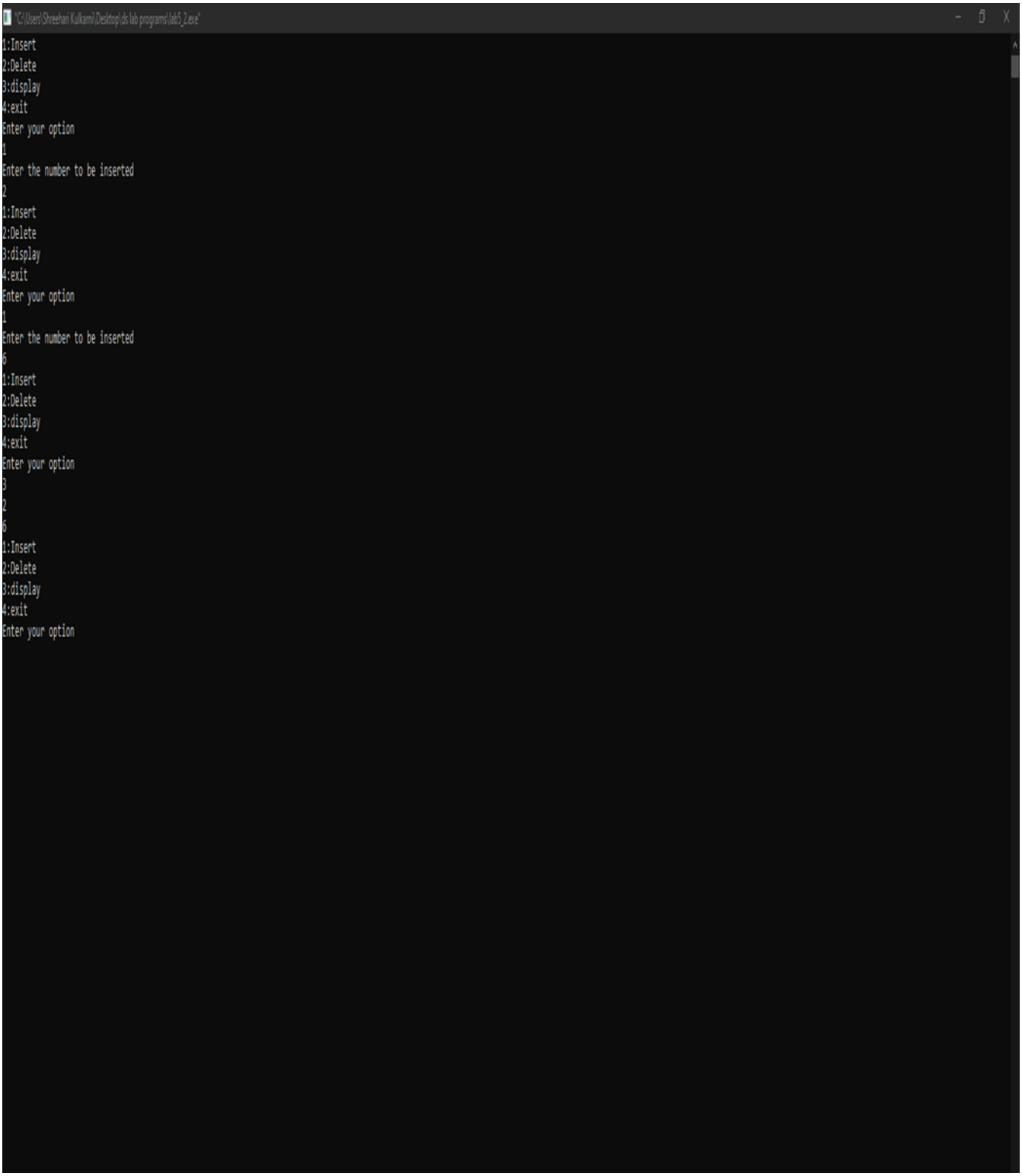
Date

OUTPUT

1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your option
1
Enter the number to be inserted
10

1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your option
3

OUTPUT


C:\Users\Shreehari Kulkarni\Desktop\ds lab programs\lab5_2.exe
1:Insert
2:Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
2
1:Insert
2:Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
6
1:Insert
2:Delete
3:display
4:exit
Enter your option
3
2
6
1:Insert
2:Delete
3:display
4:exit
Enter your option

5: WAP to Implement Singly Linked List with following operations

a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

6: WAP to Implement Singly Linked List with following operations

a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

7: WAP Implement Single Link List with following operations

a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct Node
{
    int item;
    struct Node *next;
}
Node getnode()
{
    Node x
    x = (Node) malloc( sizeof ( struct node));
    return x;

}
Node insert_front (Node first, int data)
{
    Node new_node;
    new_node = getNode();
    new_node -> item = data;
    new_node -> next = NULL;
    if (first == NULL)
        return new_node;
    new_node -> next = first;
    first = new_node;
    return first;

}
```

Date __/__/____ Saathi

```
Node insert_end(Node first, int data)
{
    Node last;
    Node new_node;
    new_node = getnode();
    new_node->item = data
    new_node->next = NULL;
    if (first == NULL)
        return new_node;
    last = first;
    while (last->next != NULL)
        last = last->next;
    last->next = new_node;
    return first;
}

Node insert_pos(Node first, int data, int pos)
{
    Node last;
    Node new_node;
    new_node = getnode();
    new_node->item = data;
    if (first == NULL && pos == 0)
    {
        new_node->next = NULL;
        return new_node;
    }
```

Page No.

```
else if (pos<1)
{
    printf("Invalid Position \n");
    return first;
}
else if (pos==1)
{
    new_node->next = first;
    first = new_node;
    return first;
}
else
{
    last = first;
    for(int i=1; i<pos-1; i++)
        last = last->next;
    new_node->next = last->next;
    last->next = new_node;
    return first;
}
}

void display(Node first)
{
    Node temp;
    if (first==NULL)
        printf("List Empty \n");
    for (temp = first; temp!=NULL; temp=temp
        printf("%d \n", temp->item);
}
```

```
Node delete_front(Node first)
{
    Node temp;
    if (first == NULL)
    {
        printf("list is Empty\n");
        return first;
    }
    temp = first;
    temp = temp -> next;
    free(first);
    return temp;
}

Node delete_end(Node first)
{
    Node temp;
    Node prev, cur;
    if (first == NULL)
    {
        printf("list Empty\n");
        break;
    }
    cur = first;
    while (cur -> next != NULL)
    {
        prev = cur;
        cur = cur -> next;
    }
    prev -> next = NULL;
    free(cur);
    return first;
}
```

Date __/__/____

Saathi

```
Node delete_pos(Node first, int pos)
{
    Node prev, cur
    if (first == NULL)
    {
        printf("list is Empty\n");
        return first;
    }
    if (pos == 1)
    {
        cur = first;
        first = first->next;
        free(cur);
        return first;
    }
    cur = first;
    for(int i = 1; i < pos; i++)
    {
        prev = cur;
        cur = cur->next;
    }
    prev->next = cur->next;
    free(cur);
    return first;
}
```

Page No.

```
Node reverse (Node first)
{
    Node cur, prev, temp;
    if (first == NULL)
    {
        printf ("list is empty\n");
        return first;
    }
    cur=NULL
    while (first != NULL)
    {
        temp = first;
        first = first->next
        temp->next = cur;
        cur = temp;
    }
    return cur;
}

Node sor

Void sort(Node *first)
{   int t;
    Node temp, t
    for (Node i = first; i != NULL; i=i->next
    {
        for (Node j = i->next; j != NULL; j
            temp-
            if (i->data > j->item)
            {
                ten t = i->data;
                j->tem = j->item
                j->item = t;
```

Date __/__/____

Saathi

```
Node merge_sort(Node a, Node b)
{
    Node result;
    if (b == NULL)
    {
        result = a;
        return result;
    }
else if (a == NULL)
{
    result = b;
    return result;
}
else
{
    Node cur = a
    while(cur->next != NUL)
        cur = cur->next;

    cur->next = b;
    return cur;
}
}
```

```
int main()
{
    Node head, a, b; = NULL;
    int data, pos, choice, in;
do{ printf("1: Insert front\n");
    printf("2: Insert End\n");
    printf("3: Insert Pos\n");
    printf("4: Delete front\n");
    printf("5: Delete End\n");
    printf("6: Delete Pos\n");
    printf("7: Sort\n");
    printf("8: Reverse\n");
    printf("9: Concat\n");
    printf("Enter Your choice\n");
    Scanf("%d", &choice);
    printf("10: Display\n");
    printf("11: exit\n");
  Switch(choice)
  {
    case 1:
      printf("Enter the data\n");
      Scanf("%d", &data);
      head = insert_front(head, data);
      break;
    case 2:
      printf("Enter the data\n");
      Scanf("%d", &data);
      head = insert_end(head, data);
      break;
    case 3:
      printf("Enter the position\n");
      Scanf("%d", &pos);
```

Date __/__/____

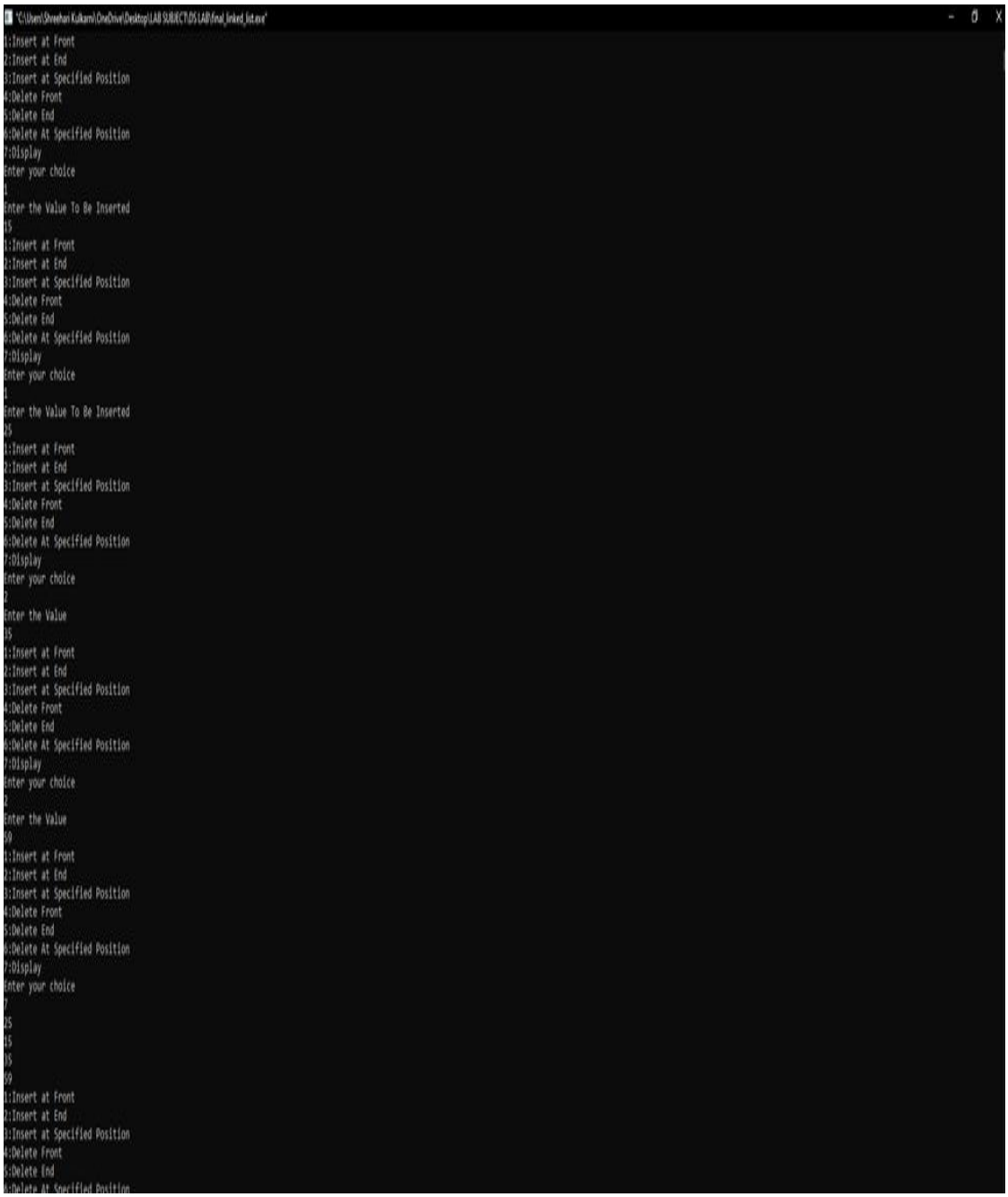
Saathi

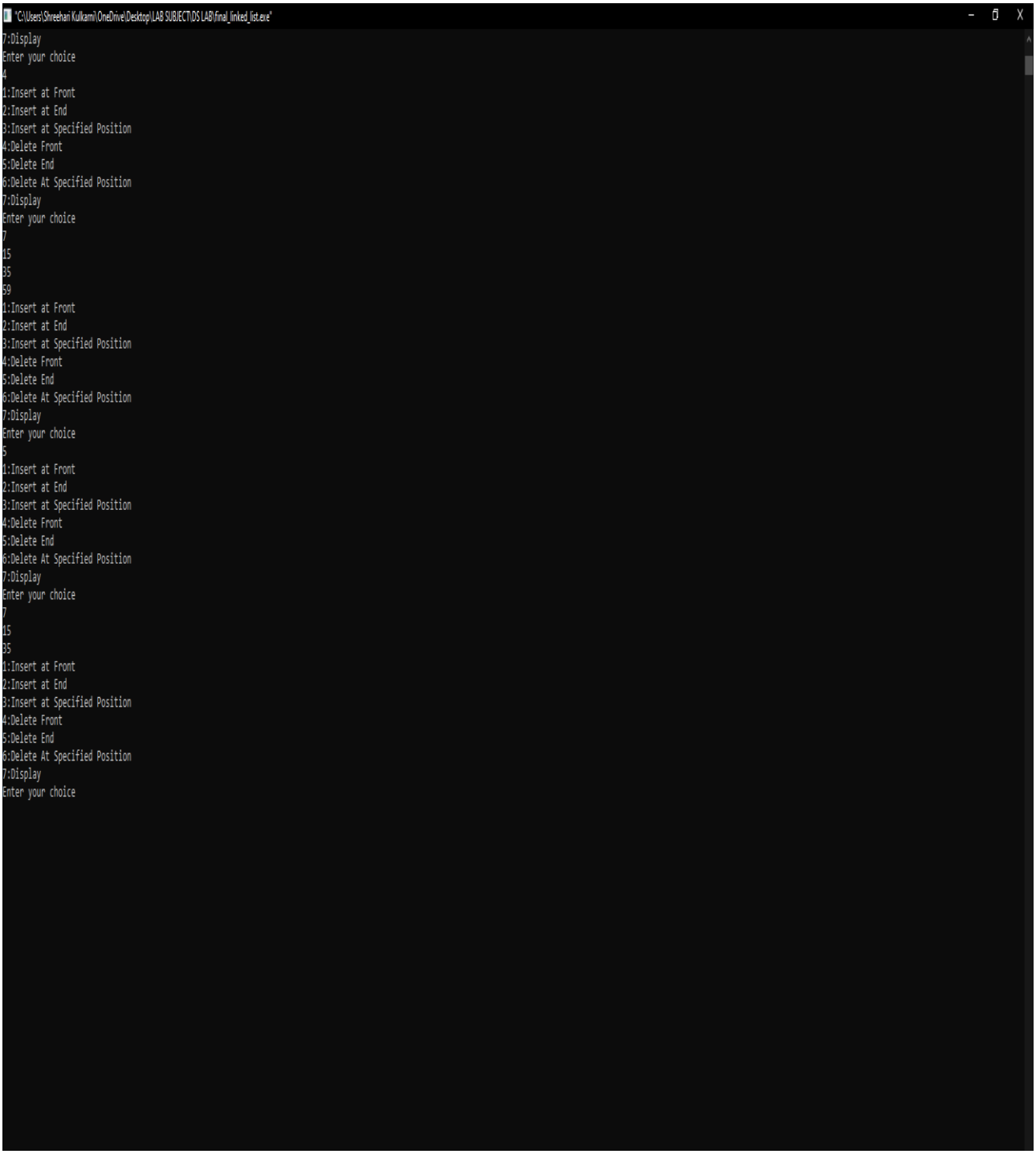
```
printf("Enter the data\n");
scanf("%d", &data);
head = insert_pos(head, data, pos);
break;

case 4:
    head = delete_front(head);
    break;

case 5:
    head = delete_end(head);
    break;

case 6:
    printf("Enter position\n");
    scanf("%d", &pos);
    head = delete_pos(head, pos);
    break;

case 7:
    sort(head);
    break;

case 8:
    head = reverse(head);
    break;

case 9:
    printf("Enter Number of nodes in list\n");
    scanf("%d", &n);
    for(i=1; i<n; i++)
    {
        printf("Enter data\n");
        scanf("%d", &data);
```

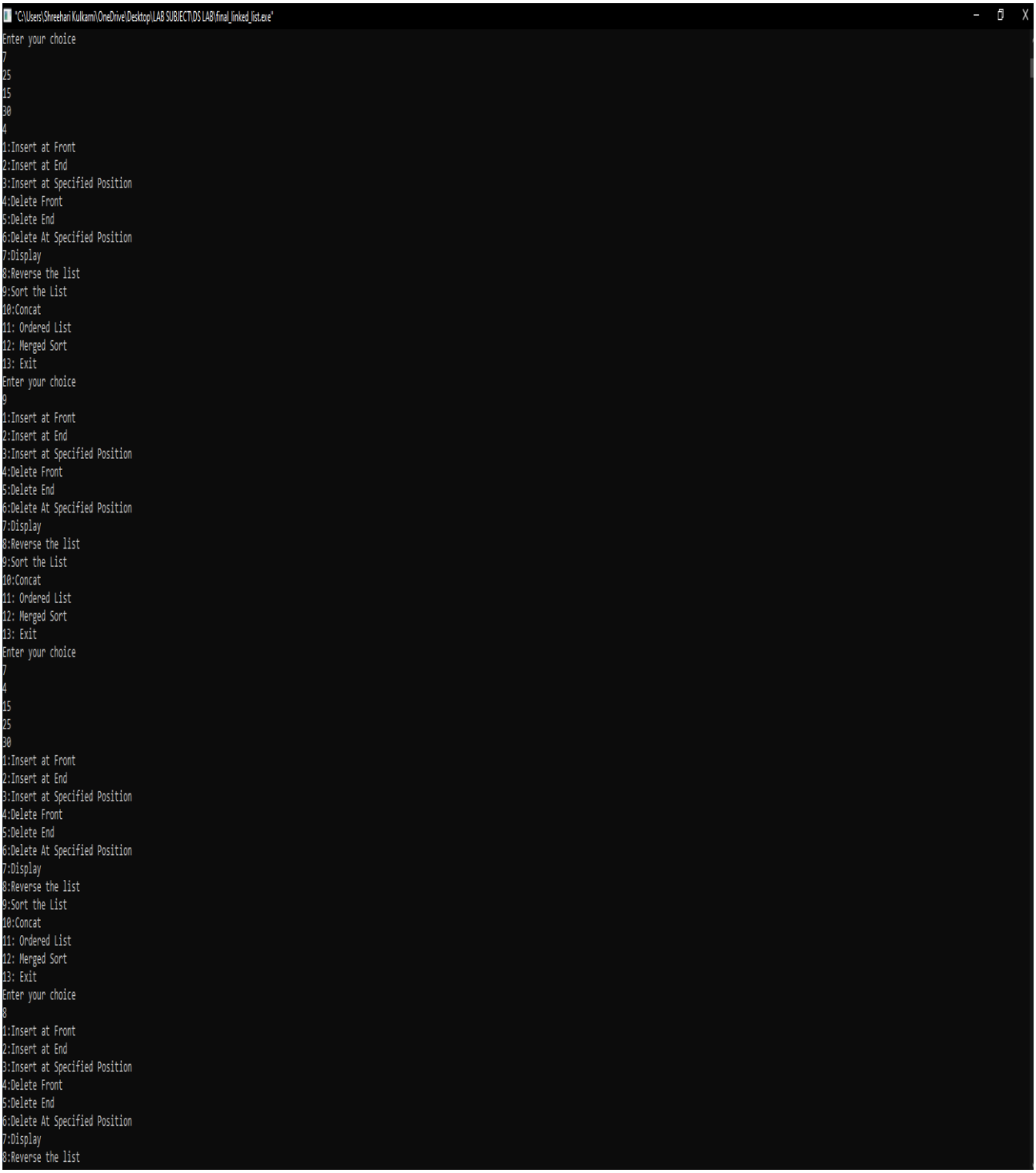
Page No.

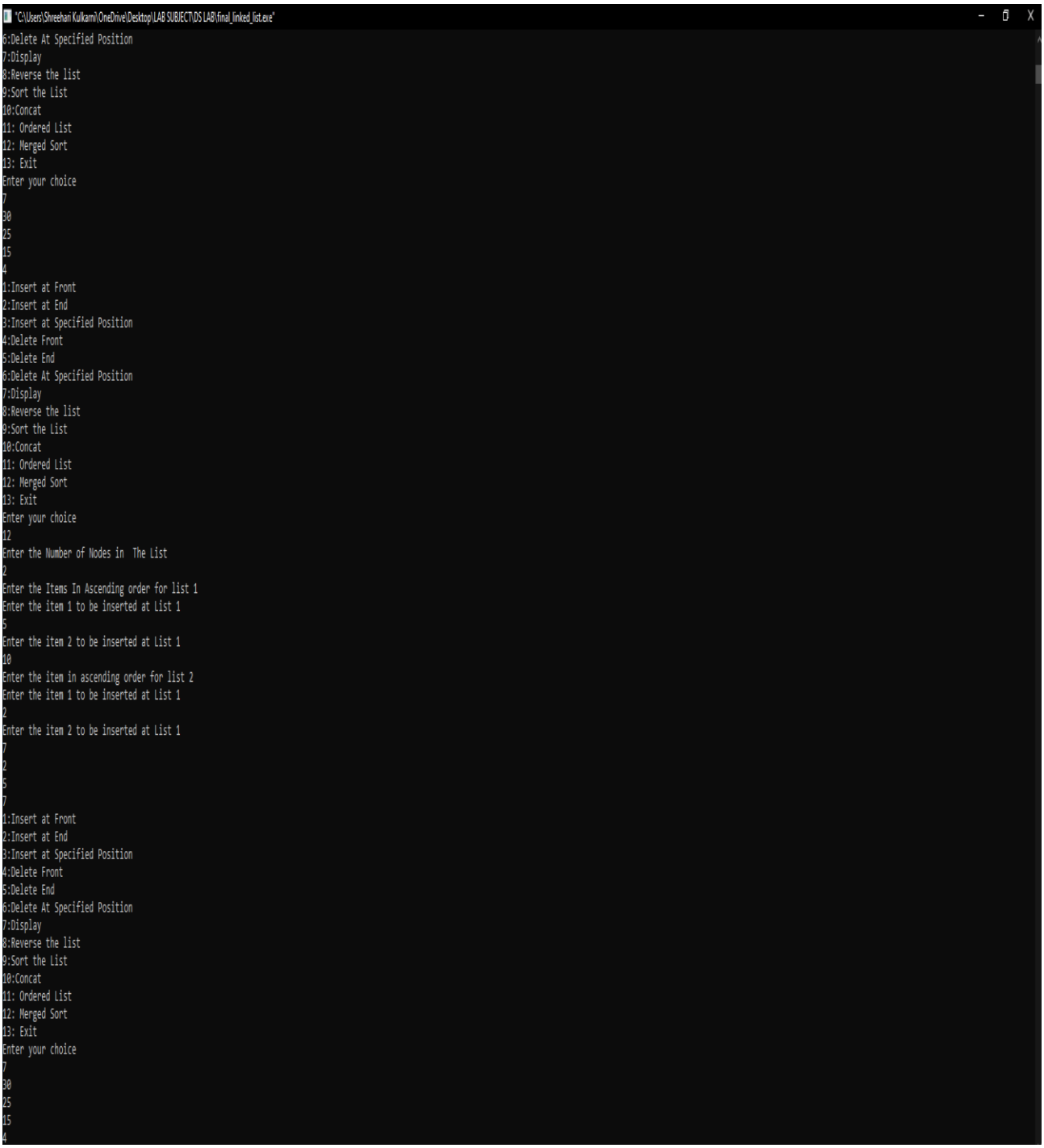
```
        a = insert_front(a, data);
    }
    printf("Enter nodes in linked list\n");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("Enter data\n");
        scanf("%d", &data);
        b = insert_front(b, data);
    }
    head = merged_sort(a, b);
    break;

case 10:
    display(head);
    break;
}
} while(option != 11);
}
```

OUTPUT

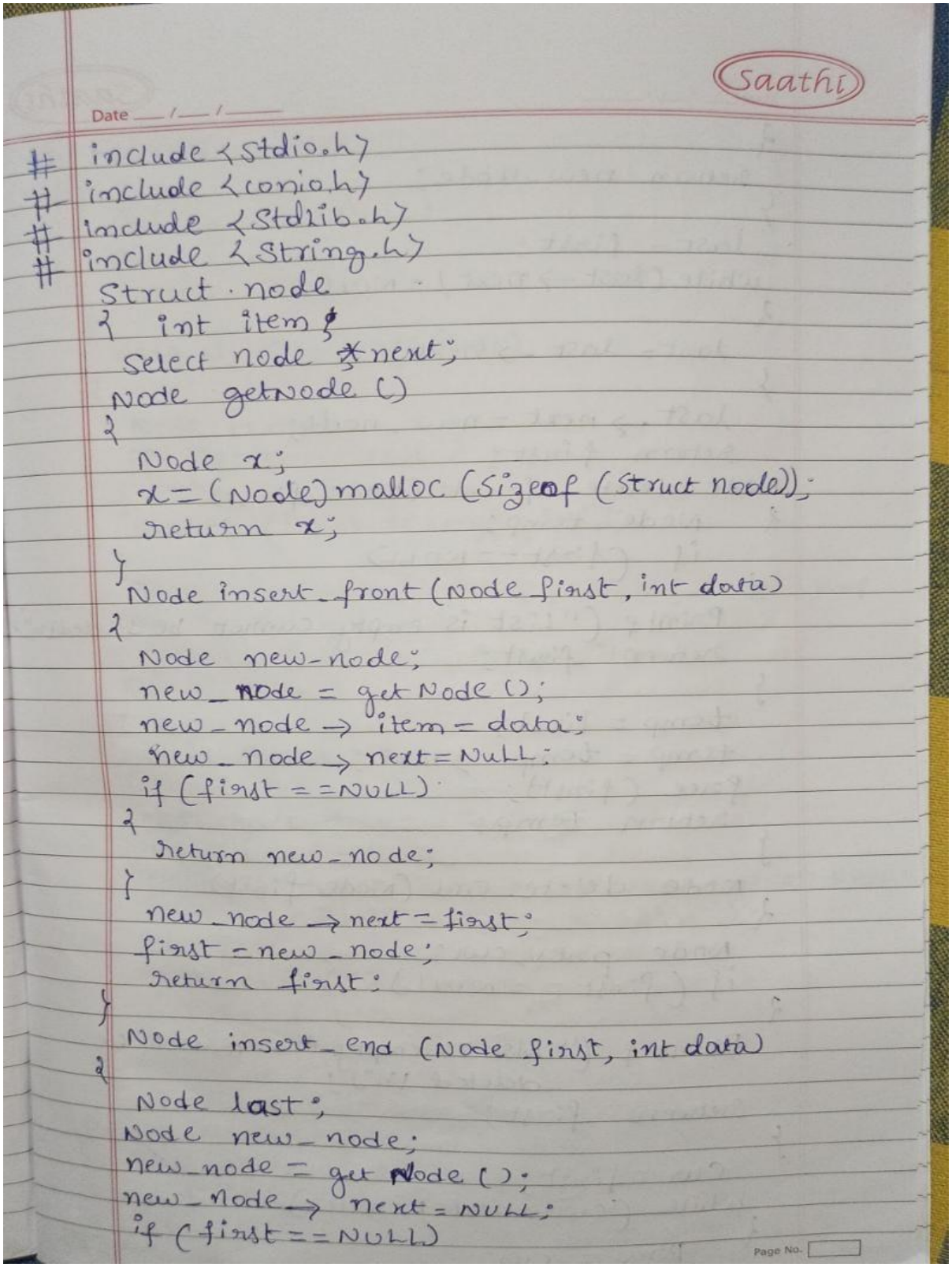

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe"
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
1
Enter the Value To Be Inserted
15
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
1
Enter the Value To Be Inserted
25
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
2
Enter the Value
35
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
2
Enter the Value
50
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
7
25
15
35
50
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe"
7:Display
Enter your choice
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
7
15
35
59
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
5
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
7
15
35
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe"
Enter your choice
7
25
15
50
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
9
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
4
15
25
50
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
8
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe"
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
30
25
15
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
12
Enter the Number of Nodes in The List
2
Enter the Items In Ascending order for list 1
Enter the item 1 to be inserted at List 1
5
Enter the item 2 to be inserted at List 1
10
Enter the item in ascending order for list 2
Enter the item 1 to be inserted at List 1
2
Enter the item 2 to be inserted at List 1
7
2
5
7
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
30
25
15
4

8: WAP to implement Stack & Queues using Linked Representation



```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
Struct node
{ int item;
 Select node *next;
Node getNode()
{
  Node x;
  x=(Node)malloc(sizeof(Struct node));
  return x;
}
Node insert_front(Node first, int data)
{
  Node new_node;
  new_node = getNode();
  new_node -> item = data;
  new_node -> next = NULL;
  if(first==NULL)
{
  return new_node;
}
  new_node -> next = first;
  first = new_node;
  return first;
}
Node insert_end(Node first, int data)
{
 Node last;
Node new_node;
new_node = getNode();
new_node -> next = NULL;
if(first==NULL)
```

```
{
  return new_node;
}
  last = first;
 while (last -> next != NULL)
{
   last = last -> next;
}
   last -> next = new_node;
   return first;
Node delete_front (Node first)
{   Node temp;
    if (first == NULL)
{
   Printf ("List is empty cannot be deleted\n");
   return first;
}
   temp = first;
   temp = temp -> next;
  free (first);
   return temp;
}
  Node delete_end (Node first)
{
   Node prev, cur;
   if (first == NULL).
{
  Print f (" List Is empty And cannot be
            deleted \n");
  return first;
}
   cur = first;
 while (cur -> next != NULL)
{   prev = cur;
```

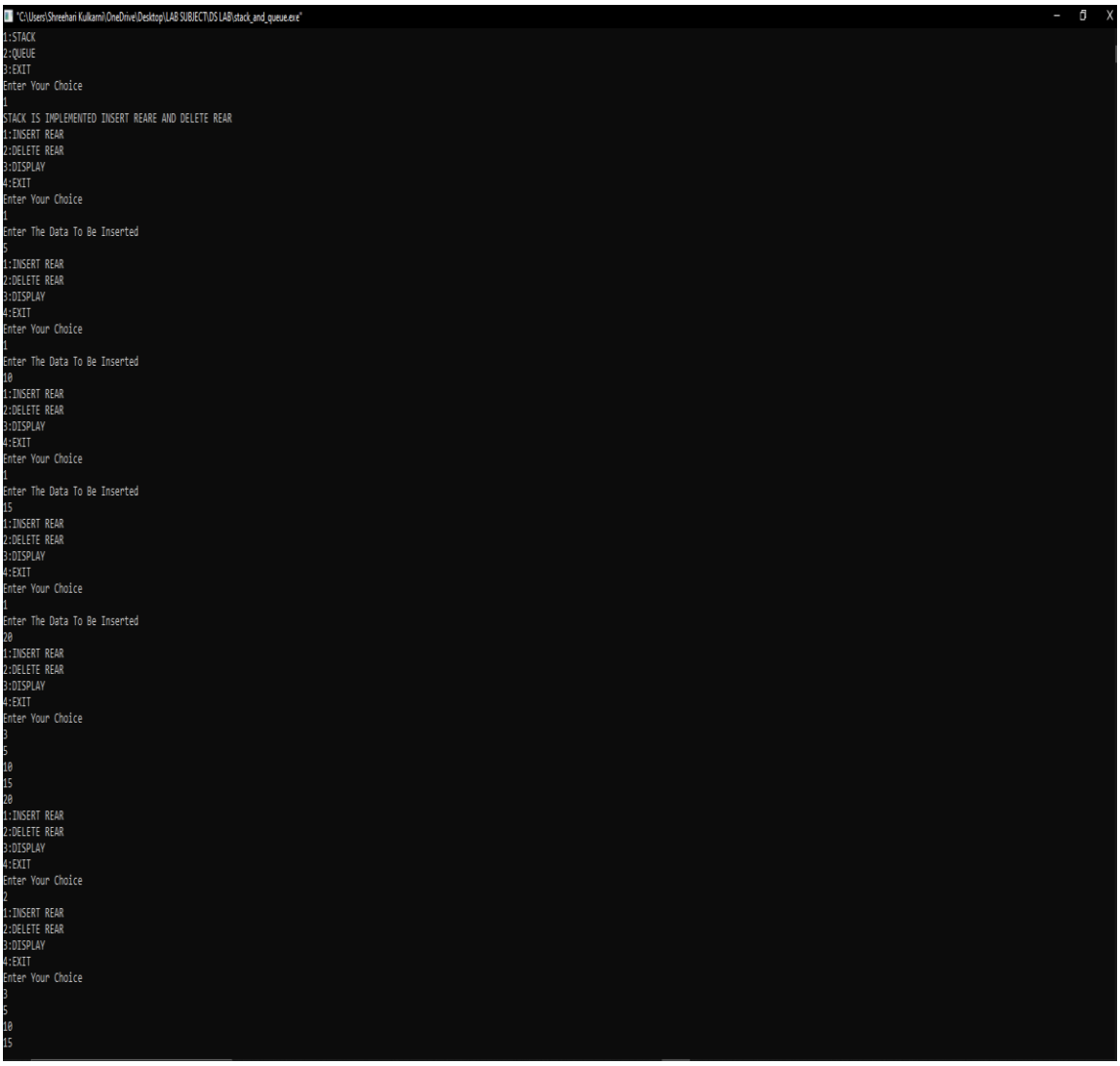
```
        cur = cur -> next;
    }
    prev -> next = NULL;
    free (cur);
    return first;
}
Void display (Node first)
{
    Node temp;
    if (first == NULL)
    {
    printf ("List is empty \n");
    }
    for (temp = first; temp != NULL; temp = temp
                -> next)
    { printf ("%d \n", temp -> item);
    }
}
    Void Stack ()
    {
    int choice, data;
    Node head = NULL;
    printf ("STACK IS IMPLEMENTED INSERT REARE
            AND DELETE REAR \n");
    do {
      printf ("1: INSERT REAR \n");
      Print f ("2: DELETE REAR \n");
      Print f ("3: DISPLAY \n");
      Print f ("4: EXIT \n");
      print f ("Enter your choice \n");
Print  Scan f ("%d", & choice \n");
      Scanf ("%d" & choice);
      switch (choice)
      {
```

```
Case 1 ;
Printf (" Enter The Data To Be Inserted\n");
Scanf ("%d", &data);
head = insert_end (head, data);
break;
Case 2 ;
    head = delete_end (head);
    break;
Case 3 ;
    display (head);
    break;
}
} while (choice != 4);
}
  void q()
{ int choice, data;
  Node head = NULL;
printf (" QUEUE IS IMPLEMENTED INSERT
        FRONT AND DELETE ~~FRONT~~ REAR \n");
 do {
    printf (" 1: INSERT FRONT\n");
    printf (" 2: DELETE FRONT\n");
    Printf (" 3: DISPLAY \n");
    Printf (" 4: EXIT \n");
    Printf (" Enter your choice \n");
    Scanf ("%d", & choice);
    Switch (choice)
    {
      case 1 :
        Printf ("Enter The data To Be
            Inserted\n");
      Scanf ("%d", & data);
      head = insert_front (head, data);
      break;
```

```
case 2:
    head = delete_front (head);
    break;
case 3:
    display (head);
    break;
}
  while (choice != 4);
}

case 4:
  head = delete_End (head);
  break;
};
  while (choice != 5);
}
int main ()
{
   int option;
Printf ("1: STACK\n");
Printf ("2: QUEUE\n");
Printf ("3: EXIT\n");
Printf ("Enter your choice\n");
Scanf ("%d", &option);
Switch (option)
{ case 1:
      stack ();
      break;
  case 2:
      q()    break;
  case 3:
     exit (0);
     break;
}}
```

OUTPUT:

STACK-OUTPUT



QUEUE-OUTPUT:

```
1:STACK
2:QUEUE
3:EXIT
Enter Your Choice
2
QUEUE IS IMPLEMENTED INSERT FRONT AND DELETE REAR
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
25
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
15
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
35
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
3
35
15
25
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
2
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
3
35
15
1:INSERT FRONT
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
```

9: WAP Implement doubly link list with primitive operations

a) Create a doubly linked list. b) Insert a new node to the left of the node.

c) Delete the node based on a specific value. c) Display the contents of the list

Name: Shreehasi Kulkarni
USN: 1BM19CS153

```
#include<stdio.h>
#include<malloc.h>
#include<stdio.h>
struct node
{
    int info
    struct node *link;
    struct node *rlink;
};
typedef struct node *NODE
NODE getnode()
{
    NODE x;
    x = (NODE) malloc( sizeof( struct node));
    if (x==NULL)
    {
        printf ("memory full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_front(int item, NODE head)
{
```

```
NODE x;
NODE temp, cur;
temp = getnode();
temp -> info = item;
cur = head -> rlink;
head -> rlink = temp;
temp -> llink = head;
temp -> rlink = cur;
cur -> llink = temp;
return head
}
NODE insert_end(int item, NODE head)
{
    NODE temp, cur;
    temp = getnode();
    temp -> info = item;
    cur = head -> rlink;
    head -> rlink = temp;
    temp -> llink = head;
    temp -> rlink = cur;
    cur -> llink = temp;
    return head
}

NODE delete_front(NODE head)
{
    NODE cur, next;
    if (head -> rlink == head)
    {
        printf("Empty list\n");
        return head;
    }
```

```
    cur = head->rlink;
    next = cur->rlink;
    head->rlink = next;
    next->llink = head;
    printf("The node deleted is %d\n", cur->info);

    free(cur);
    return head
}

NODE delete_rear(NODE head)
{
    NODE cur, prev;
    if(head->rlink == head)
    {
        printf("Empty");
        return head;
    }
    cur = head->llink;
    prev = cur->llink;
    head->llink = prev;
    prev->rlink = head;
    printf("Node deleted is %d\n", cur->info)
    free(cur);
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head->rlink == head)
    {
        printf("Empty\n");
        return;
    }
```

```
}
printf ("Contents \n");
temp = head->rlink;
while (temp != head)
{
    printf ("%d \n", temp->info);
    temp = temp->rlink;
}
printf ("\n");
}

NODE insert_leftpos(int item, NODE head)
{
    NODE temp, cur, prev;
    if (head->rlink == head)
    {
        printf ("empty \n");
        return head;
    }
    cur = head->rlink;
    while (cur != head)
    {
        if (item == cur->info)
            break;
        cur = cur->rlink;
    }
    if (cur == head)
        printf ("key not found \n");
        return head;
}
```

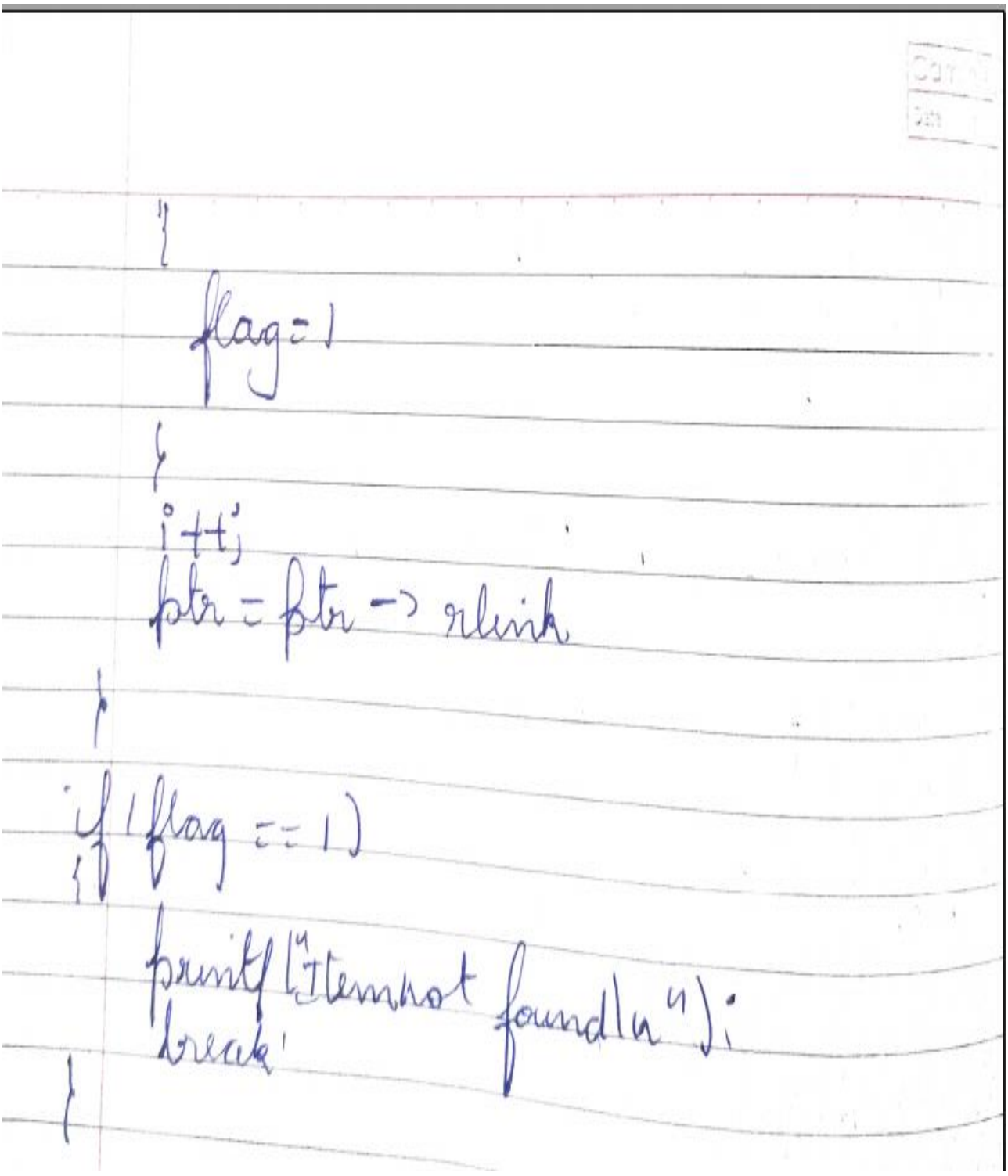
```
prev = cur -> llink;
printf(" Enter toward left of %d = ", item
temp = getnode();
scanf("%d", &temp->info);
prev -> rlink = temp;
temp -> llink = prev;
cur -> llink = temp;
temp -> rlink = cur;
return head
}
NODE insert_right_pos(int item, NODE head
```

15 x 16 + 15 x 16⁰

240 + 15

```
{
  NODE temp, cur, prev
  if (head -> rlink == head)
  {
      printf("empty\n");
      return head
  }
  cur = head -> rlink;
  while (cur != head)
  {
      if (item == cur -> info)
          break;
      cur = cur -> rlink.
  }
  if (cur == head)
      printf("key not found\n");
      return head
}
```

```
    fprev = cur -> llink
    printf("Enter toward right of %d = ", item)
    temp = getnode();
    scanf("%d", &temp -> info);
    temp -> rlink = cur -> rlink
    temp -> llink = cur
    cur -> rlink = temp
    return head
}

void search(NODE head)
{
    NODE fptr
    int item, i=0, flag;
    fptr = head;
    if (fptr == NULL)
    {
        fprintf("Empty\n");
    }
    else
    {
        fprintf("Enter item to search\n");
        scanf("%d", &item);
        while (fptr != NULL)
        {
            if (fptr -> info == item)
            {
                fprintf("Item found at %d\n", i);
                flag = 0;
                break
            }
            else
```

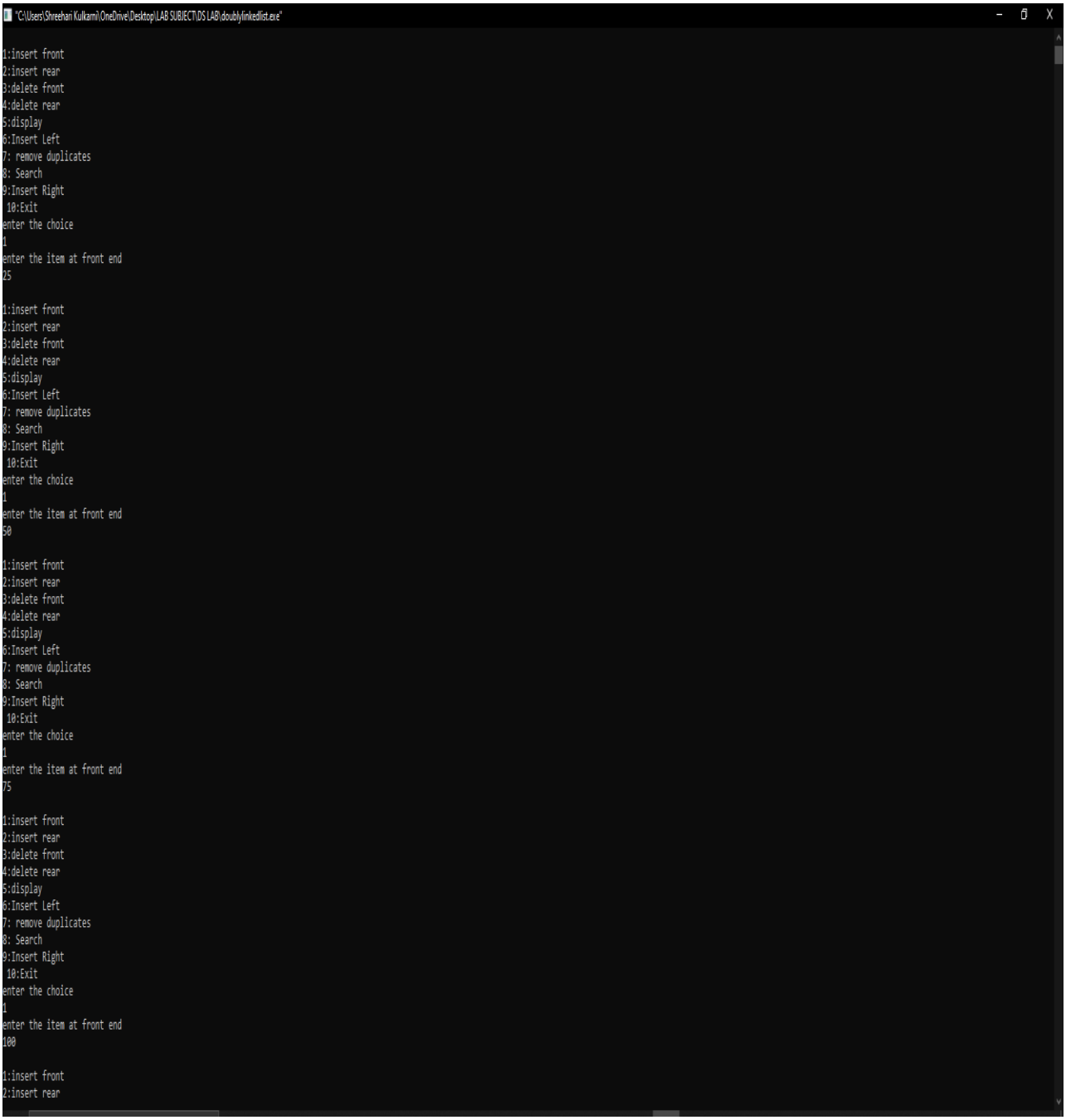

}
flag=1
{
i++;
ptr=ptr->rlink
}
if(flag==1)
{
printf("Item not found\n");
break;
}

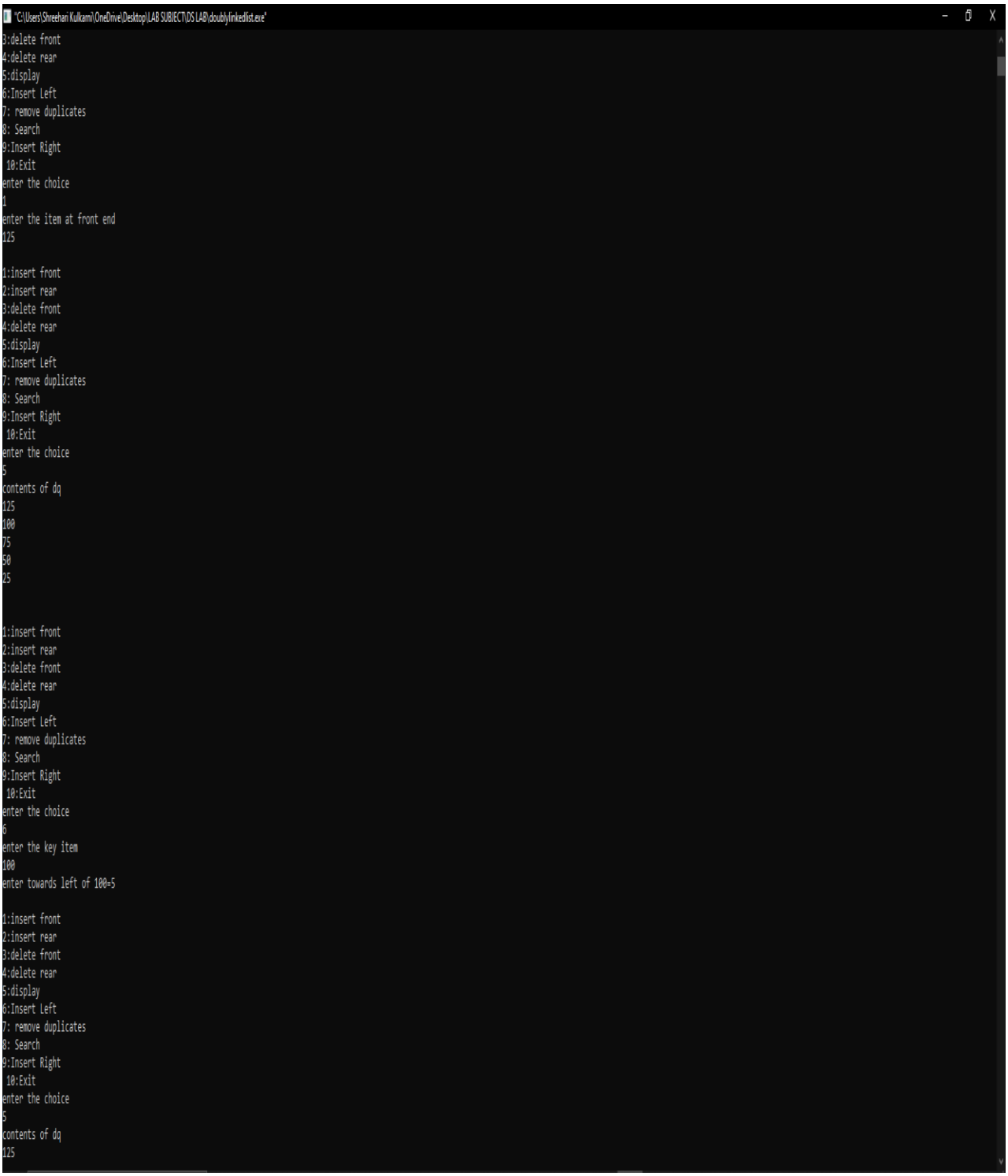
```
NODE delete_all key(int item, NODE head)
{
    NODE prev, cur, next;
    int count;
    if( head -> rlink == head)
    {
        printf("Empty list\n");
        return head;
    }
    count = 0;
    cur = head -> rlink;
    while(cur != head)
    {
        if(item != cur -> info)
            cur = cur -> rlink;
        else
        {
            count++;
            prev = cur -> llink;
            next = cur -> rlink;
            prev -> rlink = next;
            next -> llink = prev;
            free(cur);
            cur = next;
        }
    }
    if(count == 0)
        printf("Not found\n");
    else
        printf("Key found at %d position\n", count);
    return head;
}
```

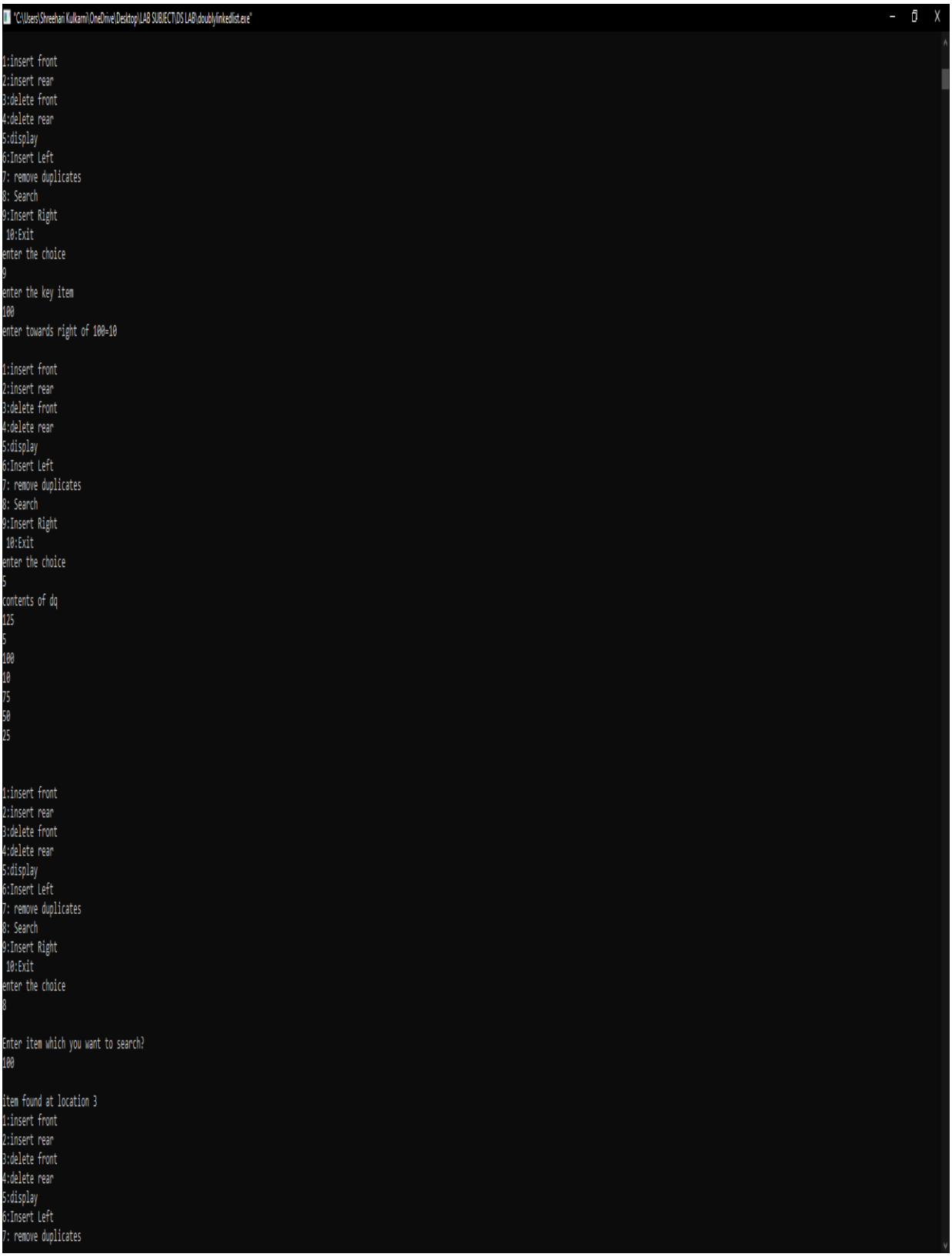
```
int main()
{
    NODE head, last
    int item, choice
    head = getnode()
    head->rlink = head
    head->llink = head
    for(;;)
    {
        printf("1: Insert front\n");
        printf("2: Insert rear\n");
        printf("3: Delete front\n");
        printf("4: Delete rear\n");
        printf("5: display\n");
        printf("6: insert left\n");
        printf("7: Remove Duplicates\n");
        printf("8: search\n");
        printf("9: insert right\n");
        printf("10: exit\n");
        printf("Enter your choice\n");
        scanf("%d", &choice)
        switch(choice)
        {
            case 1: printf("Enter item\n");
                    scanf("%d", &item);
                    last = insert_front(item, head);
                    break

            case 2: printf("Enter item\n");
                    scanf("%d", &item);
                    last = insert_end(item, head);
                    break;
```

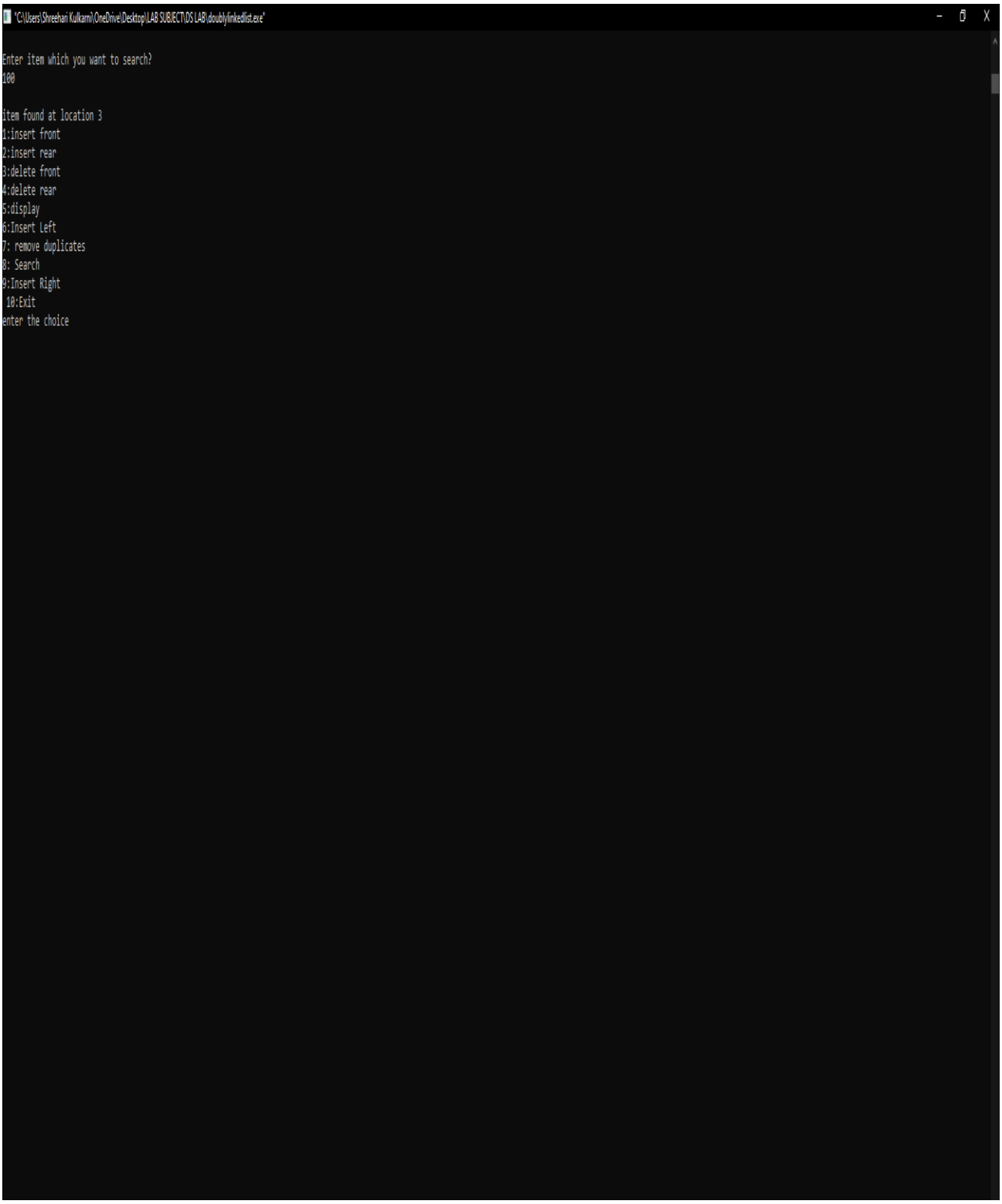
```
case 3: last = delete_front (head);
        break;

case 4: last = delete_rear(head);
        break;

case 5: display(head)
        break;

case 6: printf("Enter the item to");
        scanf("%d", &item);
        head = insert_leftpos(item, head);
        break;

case 7: printf("Enter the item to");
        scanf("%d", &item);
        head = delete_all_key(item, head);
        break;

case 8: search(head);
        break;

case 9: printf("Enter item to");
        scanf("%d", &item);
        head = insert_rightpos(item, head);
        break;

    default: exit(0);

}
```

OUTPUT


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doublylinkedlist.exe"
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
25
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
50
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
75
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
100
1:insert front
2:insert rear


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doubly\linkedlist.exe"
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
125
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125
100
75
50
25
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
6
enter the key item
100
enter towards left of 100=5
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doublylinkedlist.exe"
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
9
enter the key item
100
enter towards right of 100=10
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125
5
100
10
75
50
25
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
8
Enter item which you want to search?
100
item found at location 3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates

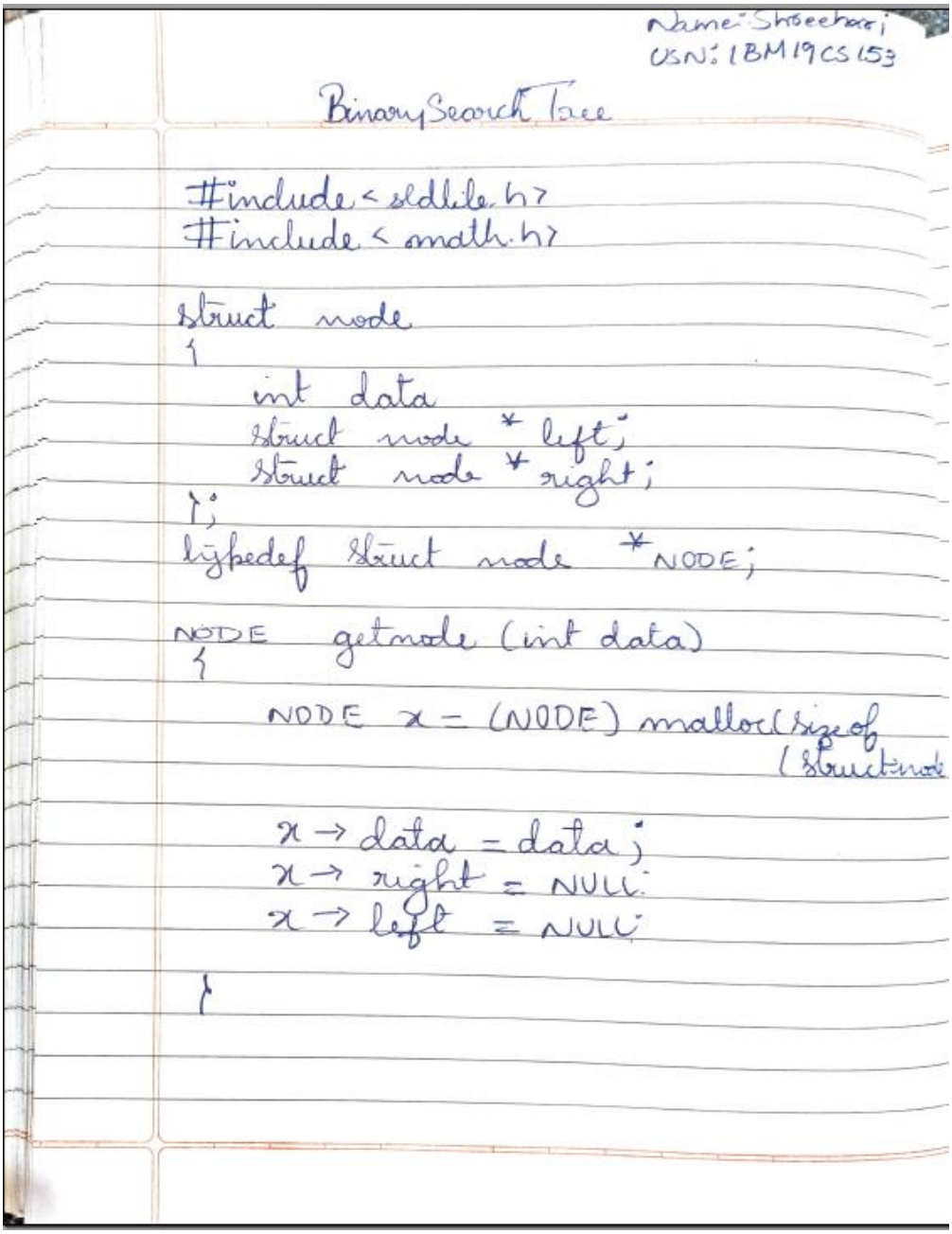

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doublylinkedlist.exe"
Enter item which you want to search?
100
item found at location 3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice

10: Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.



Name: Shreehari
USN: 1BM19CS153

Binary Search Tree

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

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

NODE getnode (int data)
{
    NODE x = (NODE) malloc(sizeof
                (struct node
    x -> data = data;
    x -> right = NULL;
    x -> left = NULL;
}
```

```
NODE insert (NODE root, int info)
{
    NODE new_node = getnode(info)
    if (new
    if
    if ( root == NULL)
        NODE new_node = getnode(info)
    else if (root -> data <= info)
        root -> right = insert(root
                               -> right, info
    else
        root -> left = insert(root -> left, inf
};
}
```

```
Void preorder(NODE root)
} if (root != NULL) {
    printf("%d\t", root->data);
    preorder(root->left);
    preorder(root->right);
  }
}

Void postorder(NODE root)
{
    if (root == NULL)
        return

    postorder(root->left);
    postorder(root->right);
}
```

```
void inorder (NODE root)
{
    if (root == NULL)
        return

    inorder (root -> left);
    printf ("%d ", root -> data);
    inorder (root -> right);

}
NODE delete_newnode (NODE root, data)
{
    if (root == NULL)
        return root
else if (info < root -> data)
    root -> left = delete_newnode
                   (root -> left, data)

else if (info > root -> data
    root -> right = delete_newnode
                    (root -> right,
                     data);

else
```

```
if (root -> right == NULL &&
{
    NODE temp = root;
    free(temp);
    root == NULL;
    return true;
}
else if (root -> right == NULL)
{
    NODE temp = root;
    free(temp)
    root = root -> right;
    return root;
}
else if (left -> right == NULL)
{
    NODE temp = root;
    free(temp)
    root = root -> right;
    return root
}
```

```
root -> left == NULL)

else
{
    NODE temp = findmin(root->right)

    root -> data = temp -> data;
    root -> right = delete
                    (root->right-
                    ,temp->d

}
return root
}
```

```
void display(NODE root, int i)
{
    if (root == NULL)
        return

    display(root->right, i);
    for(int j=1; j<=i; j++)
        printf(" ");
    printf("%d\n", root->data);
    display(root->left, i+1);
    display(root->left, i+1);

}

int height(NODE root)
{
    if (root == NULL)
        return -1;
    int left_height = height(root->left);
    int right_height = height(root->right);
    int val = max(left_height, right_height);
    return val + 1;
}
```

get

```
int leafCount(NODE root)
{
    if (root == NULL)
        return 0;
    if (root->right == NULL &&
        root->left
        == NULL)
        return 1
    else
        return (getCount(root->right)
                + getCount(root->left));
}
```

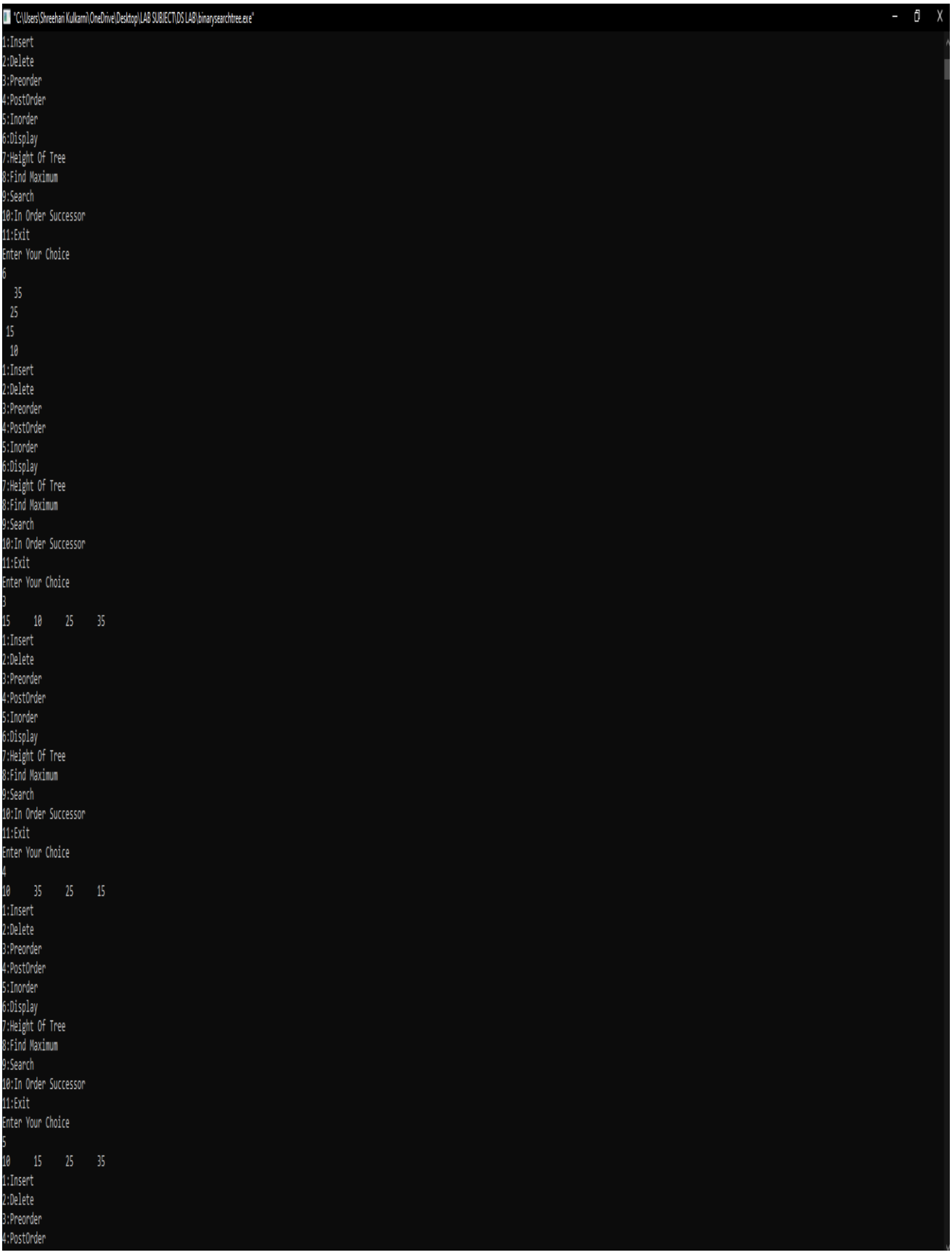
```
int main()
{
    printf("1: insert
do {
    NODE root = NULL;
    int option, data;
    printf("1: insert \n");
    printf("2: Delete \n");
    printf("3: Preorder \n");
    printf("4: Postorder \n");
    printf("5: inorder \n");
    printf("6: display \n");
    printf("Enter your option");
    scanf("%d", &option);
    switch(option)
    {
        case 1:
            printf("Enter data \n");
            scanf("%d", &data);
            root = insert(root, data
            break;

        case 2:
            printf("enter data \n");
            scanf("%d", &data);
```

```
        root = delete_node(root, data);
        break;

    case 3:
        preorder(root);
        break;

    case 4:
        postorder(root);
        break;

    case 5:
        inorder(root);
        break;

    case 6:
        display(root);
        break;
    }
} while(option != 7);
}
```

OUT

OUTPUT:


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\binarysearchtree.exe"
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
15
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
10
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
25
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
35
1:Insert
2:Delete
3:Preorder

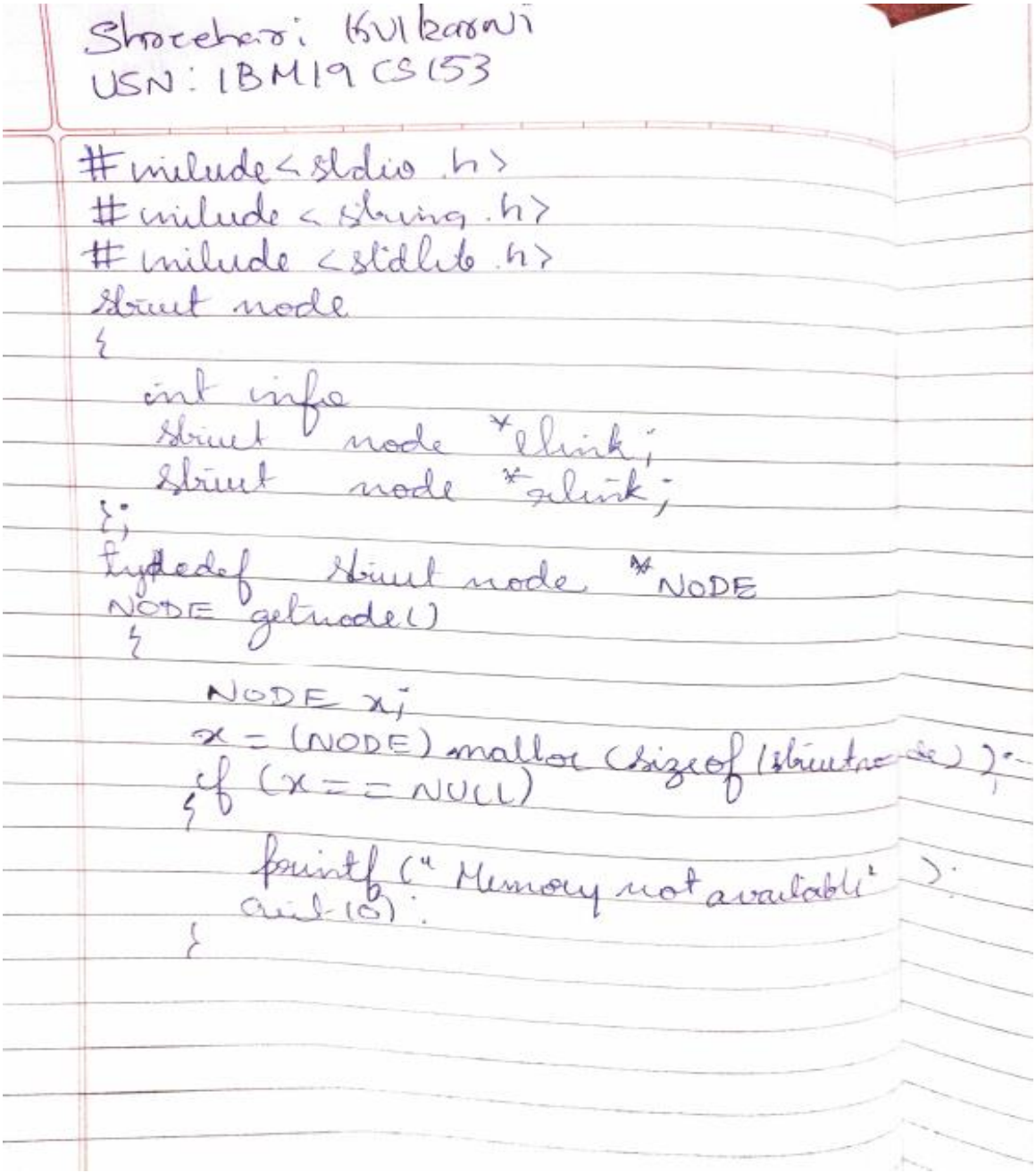

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\binarysearchtree.exe"
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
6
35
25
15
10
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
3
15 10 25 35
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
4
10 35 25 15
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
5
10 15 25 35
1:Insert
2:Delete
3:Preorder
4:PostOrder

11: Write a program

a) To construct a binary tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.



```
Shreehari KULKARNI
USN: 1BM19CS153

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node
{
   int info
   struct node *llink;
   struct node *rlink;
};
typedef struct node *NODE
NODE getnode()
{
   NODE x;
   x = (NODE) malloc (sizeof(struct node));
   if (x==NULL)
   {
      printf("Memory not available");
      exit(0);
   }
```

```
return x;

NODE insert(int item, NODE root)
{
    NODE temp, cur, prev;
    char direction[10];
    int i;
    temp = getnode();
    temp->info = item;
    temp->llink = NULL;
    temp->rlink = NULL;
    if(root == NULL)
        return temp
);--  printf("Give direction to insert\n");
    scanf("%s", direction);
    prev = NULL;
    cur = root;
    for(i=0; i< strlen(direction) &&
                cur != NULL;
                i++)
    {
        prev = cur;
        if(direction[i] == 'l')
            cur = cur->llink;
        else
```

```
        cur = cur -> rlink
    }
    if ( cur != NULL || i != strlen(direction)
    {
        printf("insertion not possible");
        free(temp);
        return(root);
    }
    if (cur == NULL)
    {
        if (direction[i-1] == 'l')
            prev -> llink = temp
        else
            prev -> rlink = temp
    }
    return(root);
}
```

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

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

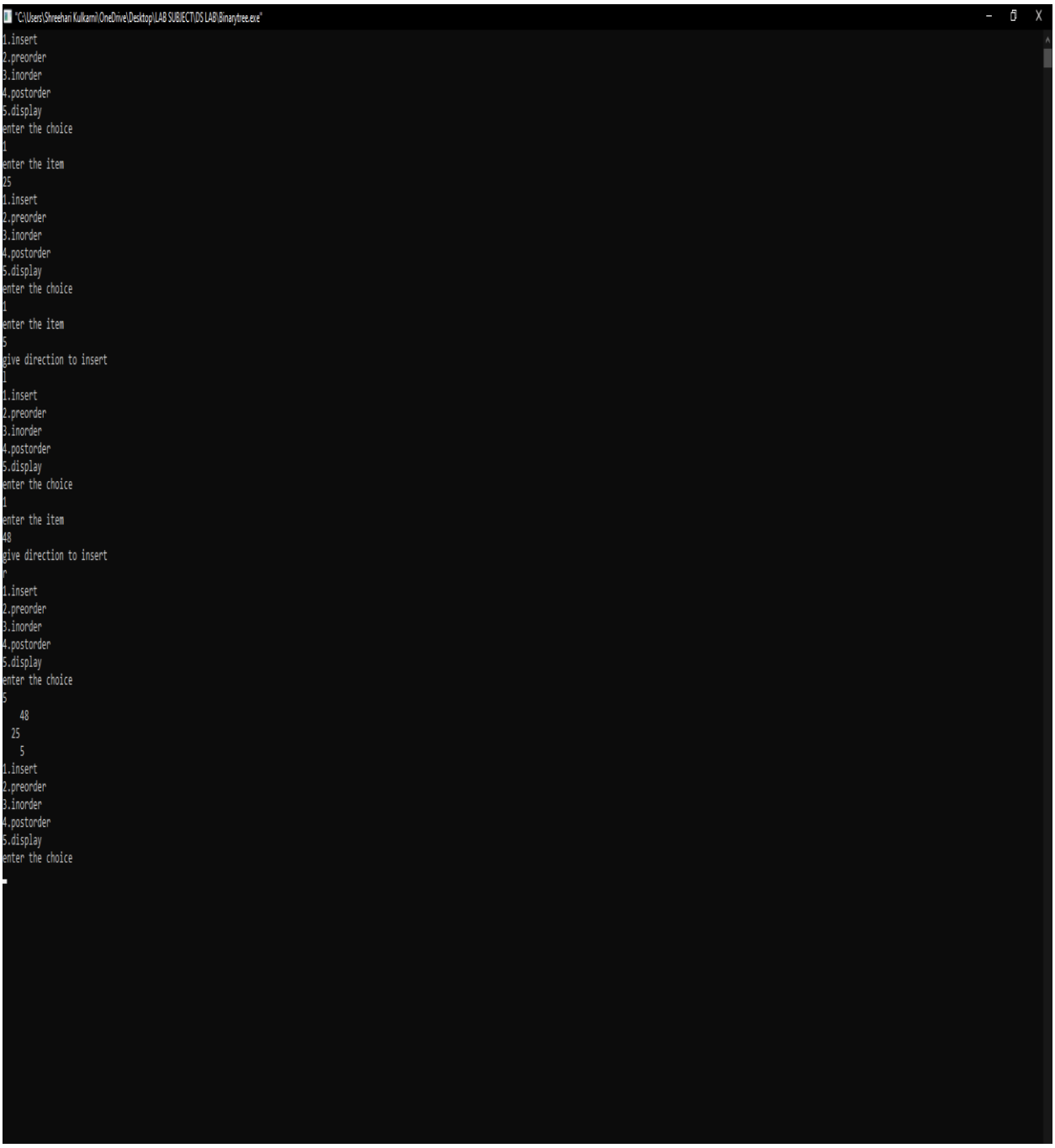
```
void postorder(NODE root)
{
  if(root != NULL)
  {
    postorder(root->llink);
    postorder(root->rlink);
    printf("%d\n", root->info);
  }
}

void display(NODE root, int i)
{
  int j;
  if(root != NULL)
  {
    display(root->rlink, i+1);
    for(j=1; j<=i; j++)
      printf("  ");
    printf("%d\n", root->info);
    display(root->llink, i+1);
  }
}
```

```
void main()
{
    NODE root = NULL
    int choice, i, item
    for(;;)
    {
        printf("1: insert \n");
        printf("2: preorder \n");
        printf("3: inorder\n");
        printf("4: postorder\n");
        printf("5: Display\n");
        scanf("%d", &choice);
switch(choice)
{
    case1: printf("Enter item\n")
           scanf("%d", &item);
           root = insert(item, root);
           break;

    case 2: if(root == NULL)
            {
              printf("tree empty\n");
            }
```

```
else
{
    printf("Given tree is");
    display(root, 1);
    printf("Preorder traversal\n");
    preorder(root);
}

break;

case 3: if(root==NULL)
{
    printf("Tree empty\n");
}
else
{
    printf("Given Tree is");
    display(root, 1);
    printf("in order traversal\n");
    inorder(root);
}
break;
```

```
case 4: if (root == NULL)
        {
            printf("Tree empty\n");
        }
        else
        {
            printf("Enumerics");
            display(root, 1);
            printf("postorder\n");
            postorder(root);
        }
        break;

case 5: display(root, 1);
        break;

default: exit(0);
    }
  }
}
```

OUTPUT:


"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\Binarytree.exe"
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
25
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
5
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
48
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
48
25
5
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice