VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

MAHANTESH GATTINA [1BM19EE025]

Under the Guidance of

Prof. POOJA S
Assistant Professor,
Dept. of CSE
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 **Mahantesh Gattina (1BM19EE025)** who is the bonafide student 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
Dept. of CSE
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. _____
   _____

Q1

LAB2

WAP, to simulate the working of stack using an array with the following:

a.] Push
b.] Pop
c.] Display.

The programs should print appropriate messages for stack overflow, stack underflow.

```
#include <stdio.h>
# define SIZE 3
int top = -1;
void push (int [], int);
int pop (int []);
void display (int []);
int main ()
{
    int stack [SIZE];
    int choice, element;
    char ch;
do
{
    printf (" Enter your choice \n");
    printf (" 1. Push \n");
    printf (" 2. Pop \n");
    printf (" 3. Display \n");
    scanf ("%d", &choice);
```

```
switch (choice).
{
        case 1: printf("Enter the element to be pushed \n");
                scanf("%d", &element);
                push(stack, element);
                break;
case 2 : element = pop(stack);
         if(element == -1)
            printf("Stack Underflow \n");
         else
            printf("Poped element is %d \n", element);
         break;
case 3: display(stack);
        break;
        default : printf("Invalid choice \n");
}
printf("Do you want to continue : ");
fflush(stdin);
scanf("%c", &ch);
}while(ch=='y' || ch=='Y');
return 0;
}
```

```
void push (int stack[], int ele)
{
    if (top == SIZE-1)
    {
        printf ("stack overflow \n");
    }
    else
    {
        top++;
        stack[top] = ele;
    }
}

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

```
void display(int stack[])
{
    int i;
    printf("The stack elements \n");
    for (i=top; i>=0; i--)
    {
     printf ("%d\n", stack[i]);
    }
}.
```

Output:

Enter your choice:

1. Push
2. Pop
3. Display

1
Enter the element to be pushed: 1
Do you want to continue: y
Enter your choice:

1. Push
2. Pop
3. Display

1.
Enter the element to pushed: 2

Do you want to continue : y

Enter your choice:

1. Push
2. Pop
3. Display

3

stack elements

2

1

Do you want to continue: y

Enter your choice:

1. Push
2. Pop
3. Display

1.

Enter the element to be pushed: 3

Do you want to continue: y

Enter your choice:

1. Push
2. Pop
3. Display

3

stack elements

3

2

1

Do you want to continue: y

Enter your choice:

1. Push
2. Pop
3. Display

2.

Poped element is 3

Do you want to continue: y

Enter your choice:

1. Push
2. Pop
3. Display.

2. Poped element is 2.

Do you want to continue: y

Enter your choice:

1. Push
2. Pop
3. Display

2. .

Poped element is 1

Do you want to continue: y

Enter your choice:

1. Push

2. Pop

3. Display

2.

Stack Under flow

Do you want to continue: n

Expected output:

```
~/structures/lab2_stack_implementation/ $ ./stack_implement
Enter your choice:
1. Push
2. Pop
3. Display
1
Enter the element to be pushed: 1
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
1
Enter the element to be pushed: 2
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
3
Stack elements
2
1
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
1
Enter the element to be pushed: 3
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
3
Stack elements
3
2
1
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
2
Poped element is 3
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
3
Stack elements
2
1
Do you want to continue: y
Enter your choice:1. Push
```

2. Pop
3. Display
2
Poped element is 2
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
2
Poped element is 1
Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
2
Stack Underflow!Do you want to continue: y
Enter your choice:
1. Push
2. Pop
3. Display
3
Stack elements
Do you want to continue: n

Q 2

LAB 3 - Infix to Postfix.

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators

+, -, *, / .

```
#include <stdio.h>
#define MAX 100
char stack[MAX];
int top = -1;

void push(char ch);
char pop();
int stackempty();
char stacktop();
int priority(char ch);

int main()
{
    char infix[100];
    int i, item;
    printf("Enter the infix expression : ");
    scanf("%s", infix);
    printf("\n postfix : ");
```

```
i=0;
while(infix[i]!='\0')
{
    switch(infix[i])
    {
        case '(': push(infix[i]);
                  break;
        case ')': while((item=pop())!='(')
                      printf("%c",item);
                  break;
        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
                  while(!stackempty() &&
                        priority(infix[i])<=
                            priority(stacktop())
                  {
                      item=pop();
                      printf("%c",item);
                  }
                  push(infix[i]);
                  break;
```

```
        default : printf("%c", infix[i]);
                  break;
    }
    i++;
}

while (!stackempty())
{
    char item;
    item = pop();
    printf("%c", item);
}
printf("\n");
return 0;
}

void push(char ch)
{
    if (top = MAX-1)
        printf("Stack is full\n");
    else
    {
        top++;
        stack[top] = ch;
    }
}
```

```
char pop()
{
    char item;
    if (top == -1)
        printf("\n stack is empty !");
    else
    {
        item = stack[top];
        top--;
        return item;
    }
}

int stackempty()
{
    if (top == -1)
    return 1;
    else
    return 0;
}

char stacktop()
{
    if (top == -1)
    printf("\n stack is empty !");
    return '\0';
    else
    return stack[top];
}
```

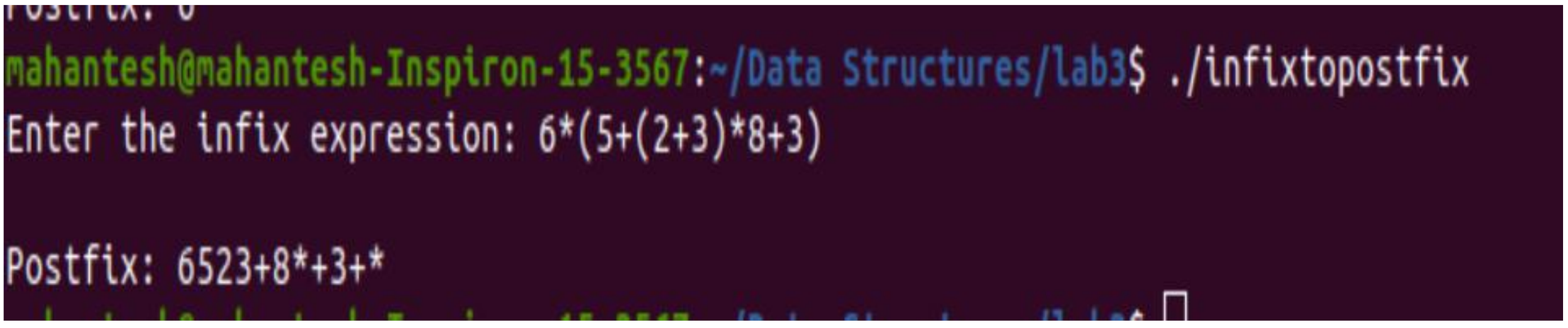
```
int priority (char ch)
{
    switch (ch)
    {
        case '+':
        case '-': return (1);
        case '*':
        case '/': return (2);
        case '^': return (3);
        default : return (0);
    }
}
```

expected output.

Enter the infix expression : 6*(5+(2+3)*8+3)

Postfix : 65234+8*+3+*

Expected output:



Q 3

LAB - 5 Linear Queue Implementation.

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 <stdlib.h>
#define MAX 5
int front =0;
int rear =-1;

int queue[MAX];

void Enqueue(int);

int Deque();
void display();
int main()
{
    int option;
    int item;
```

```
do {
    printf ("\n 1. Insert to Queue (EnQueue)");
    printf ("\n 2. Delete from the Queue (DeQueue)");
    printf ("\n 3. Display the content ");
    printf ("\n 4. Exit \n");
    printf ("Enter the option : ");
    scanf ("%d", & option);
    switch (option)
    {
        case 1 : printf ("Enter the elements \n");
                 scanf ("%d, &item);
                 Enque ( item );
                 break;
        case 2 : item = Deque();
                 if ( item = -1)
                 printf("Queue is empty \n");
                 else
                 printf (" Removed element from the
                                 queue %d", item);
                 break;
        case 3 : display ();
                 break;
        case 4 : exit(0);
    }
} while(option != 4); return 0; }
```

```
void Enque(int ele)
{
    if (rear == MAX-1)
     printf("Queue is full \n");
     else
     {
       rear++;
       queue[rear] = ele;
     }
}
int Deque()
{
     int item;
     if (front == 0) && (rear == -1)
       return -1;
     else
     {
       item = queue[front];
       front++;
       if (front > rear)
       {  front = 0;
          rear = -1;
       }
       return item;
     }
}
```

```
void display()
{
    int i;
    if (front == 0)  && rear = -1
      printf(" Queue is empty.\n");

    else
    { printf("\n Que contents: ");
      for( i = front ; i <= rear ; i++)
          printf(" %d ", queue[i]);
    }
}
```

expected output:

1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit

Enter the options: 1
Enter the element
1

1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit

Enter the option: 1

Enter the element
2

1. Insert to Queue (EnQueue)
2. delete from the queue (DeQueue)
3. Display the content
4 - Exit

Enter the option: 3

Que contents : 12

1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit

Enter the option: 2
Removed element from the Que 1

1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. display the content
4. Exit
enter the option: 2

Removed element from the Queue 2

1. Insert to Queue (EnQueue)
2. delete from the Queue (De Queue)
3. Display the content
4 - Exit

Enter the option: 2

Queue is Empty

1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit

Enter the option : 3

Expected output:

```
mahantesh@mahantesh-Inspiron-15-3567:~/Data Structures/lab/lab5/linear
queue$ ./linearqueue

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
1

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
2

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
3

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
4

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
5

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :1
Enter the element
6
Queue is full

  1. Insert to Queue (EnQueue)
  2. delete from the Queue (DeQueue)
  3. Display the content
  4. Exit
Enter the option :3
```

Queue contents:1 2 3 4 5
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 1
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 2
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3

Queue contents:3 4 5
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 3
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 4
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 5
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Queue is empty
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :4

Q 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

LAB-6

5] Circular queue.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int front = -1;
int rear = -1;
int queue[MAX];
void Enqueue(int);
int Dequeue();
void display();
int main()
{
    int option;
    int item;
    do {
        printf("Circular Queue\n");
        printf("\n1. Insert to Queue (Enqueue)");
        printf("\n2. delete from the Queue (Dequeue)");
        printf("\n3 Display the content");
        printf("\n4. Exit\n");
        printf("Enter the option :");
```

```
scanf("%d", &option);
switch(option)
{
        case 1: printf("Enter the element:\n");
                scanf("%d", &item);
                Enque(item);
                break;
        case 2: item = Deque();
                if(item == -1)
                  printf("Queue is empty\n");
                else
                printf("Removed element from
                                the que %d", item);
                break;
        case 3: display();
                break;
        case 4: exit(0);
}
} while(option != 4);
return 0;
}
```

```
void Enque( int ele)
{
    if ((rear+1) %. MAX == front)
    printf("Queue is full\n");
    else
    {
        rear = (rear+1) %. MAX;
        queue[rear] = ele;
        if (front == -1)
            front = 0;
    }
}
int Deque()
{
    int item;
    if ((front == -1) && (rear == -1))
    return -1;
    else
    {
        item = queue[front];
        front = (front+1) %. MAX;
        if (front > rear)
        {
            front = -1;
            rear = -1;
        }
        return item;
```

```
    }
}
void display()
{
    int i;
if ((front == 0) && (rear == -1))
  printf("Queue is empty \n");

else
{
    printf("\n Queue contents : ");
    for(i = front ; i <= rear; i++)
        printf("%d ", queue[i]);
  } printf("\n");
}
```

expected output:

Circular Queue

1. Insert to Queue (Enqueue)
2. Delete from the Queue (Dequeue)
3. Display the Content
4. Exit

Enter the option: 2

Queue is empty

Circular Queue

1. Insert to queue (enQueue)
2. Delete from the queue (DeQueue)
3. Display the content
4. Exit

Enter the option : 1.

Enter the element

2.

Circular Queue.

1. Insert to queue (enQueue)
2. delete from the queue (DeQueue)
3. Display the content
4. exit.

Enter the option : 4.

Priority Queue.

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

int queue[3][N];
int front[3] = {0, 0, 0};
int rear[3] = {-1, -1, -1};

int item, pr;
void pqinsert(int pr)
{
    if(rear[pr] == N-1)
      printf("\n Que overflow \n");
else
{
    printf("\nenter the item \n");
    scanf("%d", &item);
    rear[pr]++;
    queue[pr][rear[pr]] = item;
}
    return;
}
void pqdelete()
{
    int i;
    for(i=0; i<3; i++)
```

```
{
    if (rear[i] == front[i] - 1)
     printf("\n queue %d empty \n", i+1);
    else
    {
     printf("\n deleted item is %d of queue %d \n",
                                    queue[i][front[i]]);
     front[i]++;
     return;
    }
  }
}
void display()
{
   int i, j;
   for(i=0; i<3; i++)
   {
     if (rear[i] == front[i] -1)
     printf("\n queue %d empty \n", i+1);
    else
    {
     printf("\n QUEUE %d: ", i+1);
     for (j = front[i]; j<= rear[i]; j++)
      printf("%d\t", queue[i][j]);
```

```
    }
    return;
}
int main()
{
    int ch;
    while(1)
    {
        printf("\n\t 1: PQ insert \n");
        printf("\n\t 2: PQdelete \n");
        printf("\n\t 3: PQ display \n");
        printf("\n\t 4: Exit \n");
        printf("\n Enter the choice\n");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: printf("\n enter the priority number\n");
                    scanf("%d", &pr);
                    if(pr>0 && pr<4)
                    pq insq(pr-1);
                    else
                    printf("\n only 3 priority exists 123\n");
                    break;
            case 2: pq deletet();
                    break;
```

```
        case 3: display();
                break;
        case 4: exit(0);
        }
}
    return 0;
}
```

Executed Output.

1: PQ insert
2: PQdelete
3: PQ display
4: Exit

enter the choice

1

enter the priority number

2

enter the item

45

1: PQ insert
2: PQdelete
3: PQdisplay
4: Exit

enter the choice

1

enter the priority number

1

enter the item

67

1 : PQ insert
2 : PQ delete
3 : PQ display
4 : Exit

enter the choice

3

QUEUE 1 : 67

QUEUE 2 : 45

queue 3 empty

1 : PQ insert

2 : PQ delete

3 : PQ display

4 : Exit

enter the choice

2

deleted item is 67 of queue 1

1: PO insert

2: PO delete

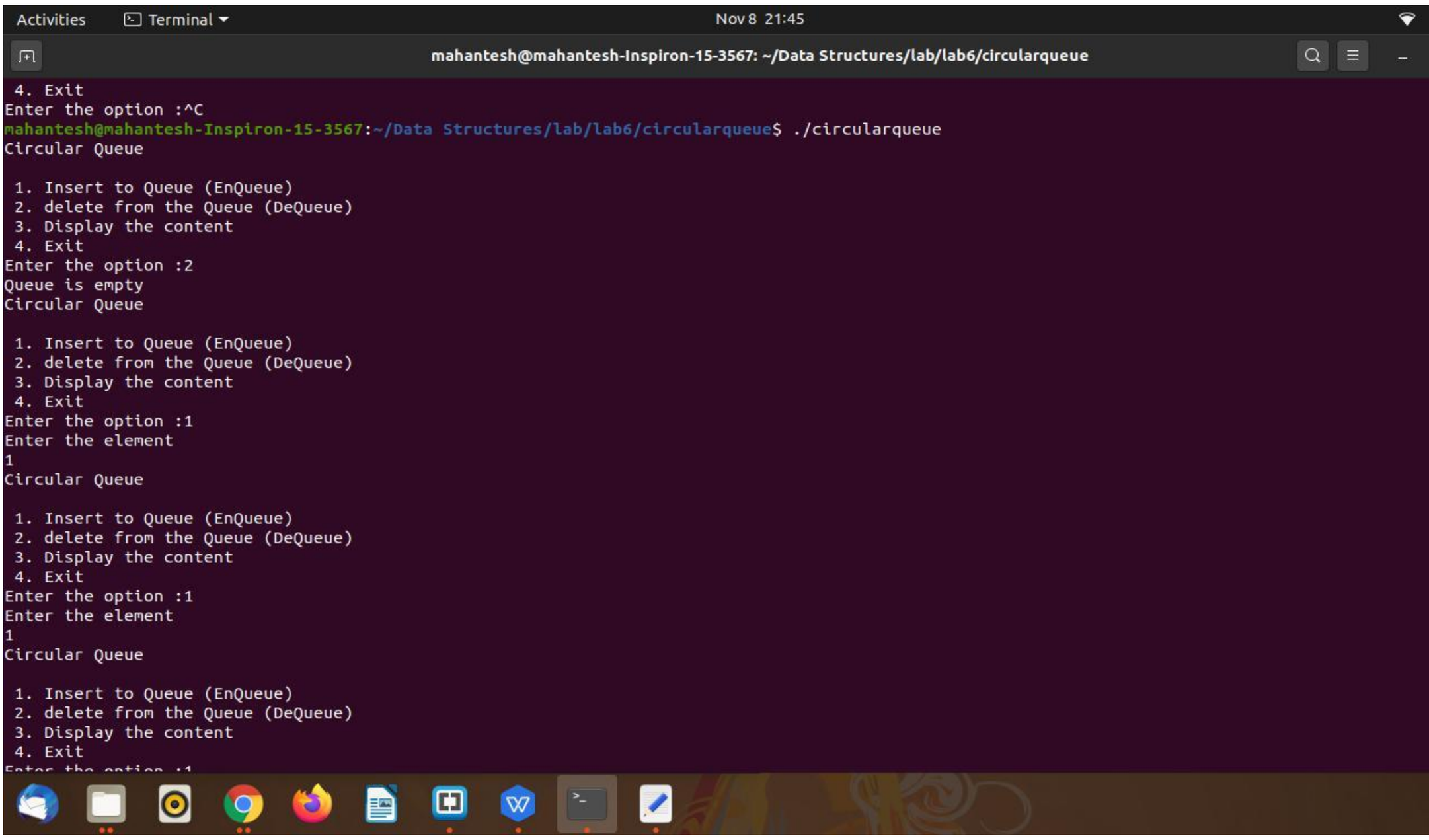
3: PO display

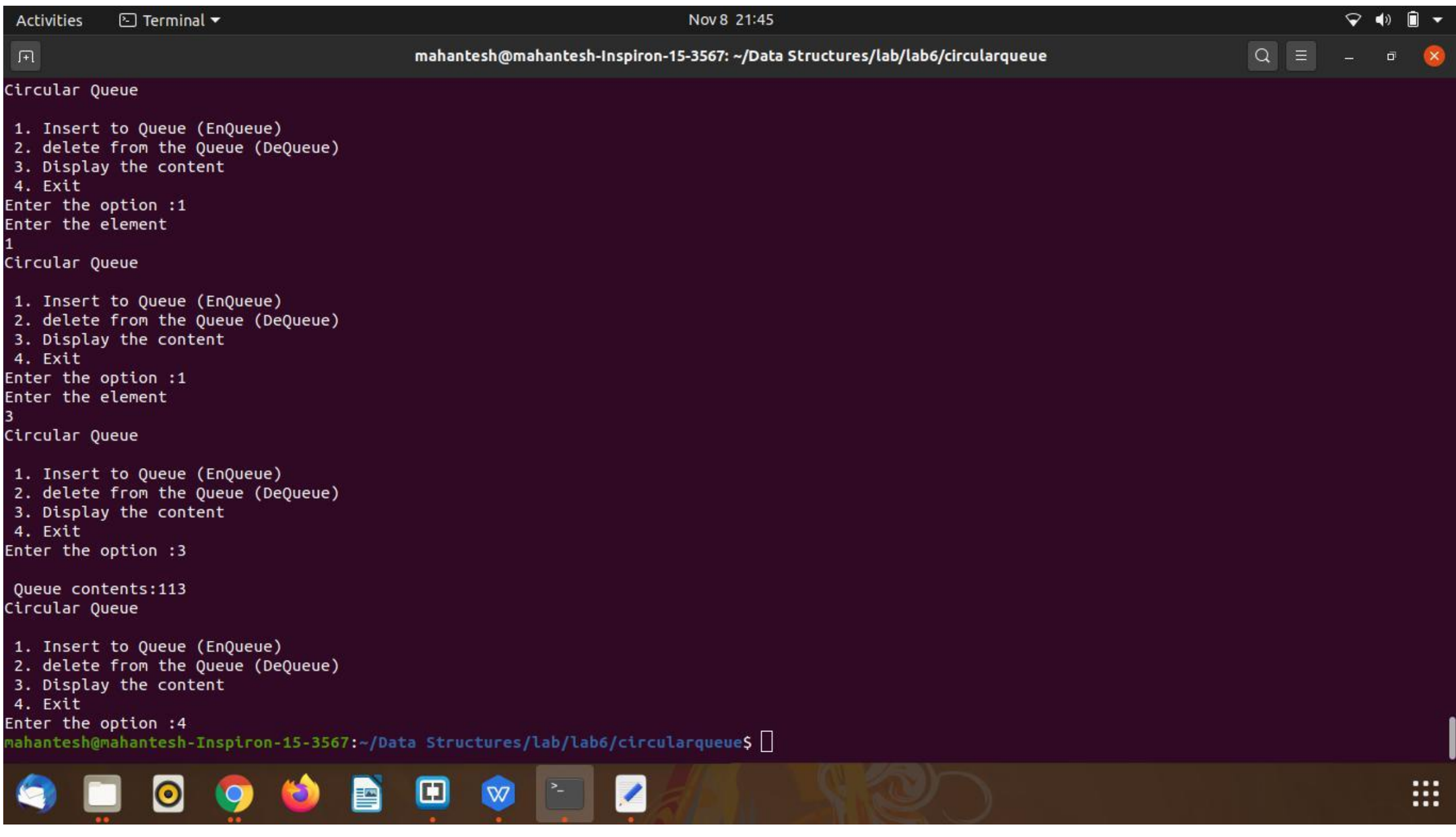
4: Exit

enter the choice

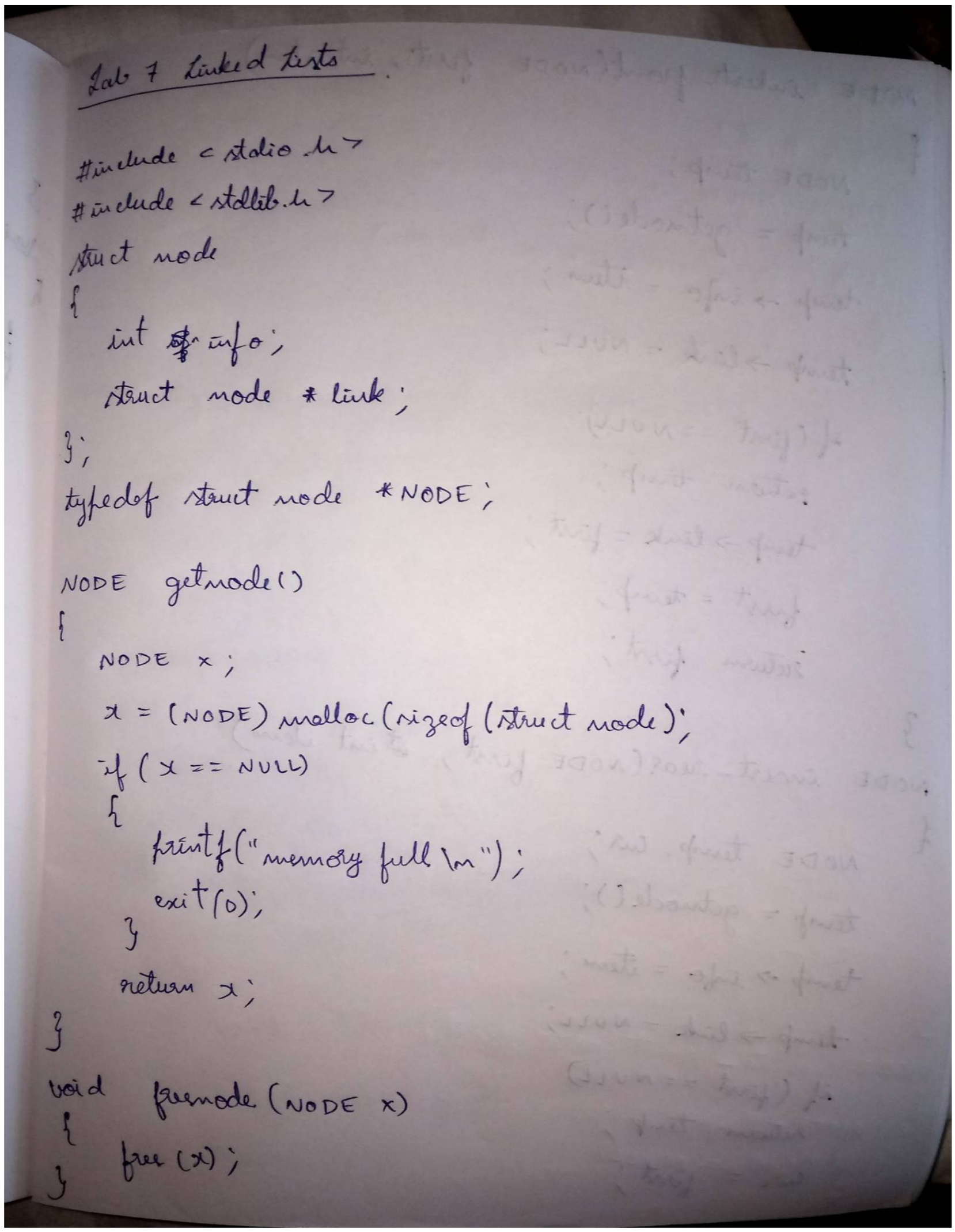
4

Circular queues expected output:





Q 5.*WAP to Implement Singly Linked List with following operations a) 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 d) Deletion of first element, specified element and last element in the list. e)Sort the linked list. f) Reverse the linked list. g) Concatenation of two linked lists*



Lab 7 Linked Lists

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

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

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

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

```
while (cur -> link != NULL)
    cur = cur -> link;
    cur -> link = temp;
    return first;
}
NODE insert_pos(int item, int pos, NODE first)
{
    NODE temp, prev, cur;
    temp = getnode();
    temp -> info = item;
    temp -> link = NULL;
    if ( pos == 1 && first == NULL)
      return temp;
    if (first == NULL)
    {  printf("Invalid Position\n");
       return first;
    }
    if (pos == 1)
    {
       temp -> link = first;
       return temp;
```

```
int count = 1;
cur = first;
prev = NULL;
while (cur != NULL && count != pos)
{
    prev = cur;
    cur = cur -> link;
    count ++;
}
if (count == pos)
{
    prev -> link = temp;
    temp -> link = cur;
    return first;
}
printf ("Invalid position\n");
return first;
}
NODE delete_front (NODE first)
{
    NODE temp;
    if (first == NULL)
    {
```

```
    printf("list is empty cannot delete \n");
    return first;
}
    temp = first;
    temp = temp -> link;
    printf(" item deleted at front-end is %d \n", first->info);
    free(first);
    return temp;
}
NODE delete_rear(NODE first)
{
    NODE cur, prev;
    if (first == NULL)
    {
        printf(" list is empty cannot delete \n");
        return first;
    }
    if ( first -> link == NULL)
    {
        printf(" item deleted is %d \n", first -> info);
        free( first );
        return NULL;
    }
```

```
prev = NULL;
cur = first;
while (cur -> link != NULL)
{
    prev = cur;
    cur = cur -> link;
}
printf("item deleted at rear-end is %d", cur->info);
free(cur);
prev -> link = NULL;
return first;
}
NODE delete_pos(int pos, NODE first)
{
    NODE cur;
    NODE prev;
    int count;
    if(first == NULL || pos <= 0)
    {
        printf("invalid position\n");
        return NULL;
    }
```

```
if (pos == 1)
{
    cur = first;
    first = first -> link;
    freenode(cur);
    printf("Node deleted succesfully \n");
    return first;
}
prev = NULL;
cur = first;
count = 1;
while (cur != NULL)
{   if (count == pos) { break; }
    prev = cur;
    cur = cur -> link;
    count ++;
}
if (count != pos)
{
    printf("Invalid position\n");
    return first;
}
prev -> link = cur -> link;
freenode(cur);
return first;
}
```

```
void swap(NODE a, NODE b)
{
    int temp = a->info;
    a->info = b->info;
    b->info = temp;
}
void bubblesort(NODE first)
{
    int swapped;
    NODE cur;
    NODE prev = NULL;
    if (first == NULL)
    {
        printf("Empty linked list \n");
        return;
    }
    do
    {
        swapped = 0;
        cur = first;
        while (cur->link != prev)
        {
            if (cur->info > cur->link->info)
            {
                swap(cur, cur->link);
                swapped = 1;
            }
            cur = cur->link;
```

```
    } prev = cur;
  } while (swapped);
}
NODE concat(NODE first, NODE second)
{  NODE cur;
  if (first == NULL)
   return second;
  if (second == NULL)
   return first;
  cur = first;
  while (cur -> link != NULL)
    cur = cur -> link;
    cur -> link = second;
    return first;
}
NODE reverse (NODE first)
{
    NODE cur, temp;
  cur = NULL;
  while (first != NULL)
  {  temp = first;
     first = first -> link;
     temp -> link = cur; cur = temp; } return cur; }
```

```
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
    printf ("list cannot display items \n");
for (temp = first ; temp != NULL; temp = temp -> link)
    {
        printf ("%d \n", temp -> info);
    }
}

int main( )
{   int item, choice, pos, i, n;
    NODE first = NULL, a, b;
    for (;;) {
    printf ("\n 1: Insert_front \n 2: Insert_rear \n 3.
Insert at specified position \n 4: Delete front \n 5. Delete_rear
\n 6. Delete at specified position \n 7. sort \n 8. Concatenate
two lists \n 9. Reverse the list \n 10. Display the list \n
11. Exit \n");
printf ("Enter the choice \n");
scanf ("%d", &choice);
```

```
switch (choice)
{
    case 1: printf("enter the item at the front-end \n");
            scanf("%d", &item);
            first = insert_front(first,item);
            break;
    case 2: printf("Enter the item at rear_end \n");
            scanf("%d", &item);
            first = insert_rear(first, item);
            break;
    case 3: printf("enter the item\n");
            scanf("%d", &item);
            printf("enter the position \n");
            scanf("%d", &pos);
            first = insert_pos(item, pos, first);
            break;
    case 4: first = delete_front(first);
            break;
    case 5: first = delete_rear(first);
            break;
    case 6: printf("Enter the position \n");
            scanf("%d", &pos);
            first = delete_pos(pos, first); break;
```

```
case 7: bubblesort(first);
        printf("Items in sorted order are \n");
        display(first);
        break;
case 8: printf("enter the no. of nodes in 1 \n");
        scanf("%d", &n);
        a = NULL;
        for(i=0; i<n; i++)
        {
          printf("enter the item \n");
          scanf("%d", &item);
          a = insert_rear(a, item);
        }
        printf("enter the no. of nodes in 2 \n");
        scanf("%d", &n);
        b = NULL;
        for(i=0; i<n; i++)
        {
          printf("enter the item \n");
          scanf("%d", &item);
          b = insert_rear(b, item);
        }
        a = concat(a, b);
        printf("Concatenated list : \n");
        display(a);
        break;
case 9: first = reverse(first);
        display(first);
        break;
case 10: printf("List : \n");
         display(first);
         break;
case 11: exit(0);
default: printf("Enter correct instruction !!!!");
         break;
}
}
return 0;
}
```

Output :

1. Insert front
2. Insert Rear
3. Insert at specified position
4. Delete front
5. Delete Rear

6. Delete at specified pos

7. Sort

8. Concatenate

9. Reverse

10. Display

11. Exit

Enter choice : 11

Expected Output:

```
mahantesh@mahantesh-Inspiron-15-3567:~/Data Structures/lab/lab7$ make
llist
cc
llist.c
-o llist
mahantesh@mahantesh-Inspiron-15-3567:~/Data Structures/lab/lab7$ ./llist
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
1
enter the item at front-end
1
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
2
enter the item at rear-end
10
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
3
enter the item
12
enter the position
3
1:Insert_front
2:Insert_rear
3.Insert at specified position
```

4:Delete_front
5.Delete_rear6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
10
List :
1
10
12
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
1
enter the item at front-end
10
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
1
enter the item at front-end
19
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
4
item deleted at front-end is=191:Insert_front
2:Insert_rear
3.Insert at specified position

4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
5
item deleted at rear-end is 12
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
6
enter the position
1
Node deleted successfully
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
10
List :
1
10
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list11.Exit
enter the choice
1
enter the item at front-end
-1
1:Insert_front
2:Insert_rear

3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
1
enter the item at front-end
100
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
7
Items In Sorted Order are
-1
1
10
100
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
9
Reverse list :
100
10
1
-11:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit

enter the choice
8
enter the no of nodes in 1
4
enter the item
1
enter the item
2
enter the item
3
enter the item
4
enter the no of nodes in 2
2
enter the item
1
enter the item
2
Concatenated list :
1
2
3
4
1
2
1:Insert_front
2:Insert_rear
3.Insert at specified position
4:Delete_front
5.Delete_rear
6.Delete at specified position
7.Sort
8.Concatenate two lists
9.Reverse the list
10.Display_list
11.Exit
enter the choice
11

Q 6. WAP to implement Stack & Queues using Linked Representation

Implementation of stacks & queues using link lists.

1BM19EE025

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node * link;
};
typedef struct node *NODE;
NODE first = NULL;
NODE second = NULL;
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_rear (NODE first, int item)
{
   NODE temp, cur;
   temp = getnode ();
```

```
   temp->
   temp-> link = NU
   if (first == NU
      return temp
   cur = first;
   while( cur -> li
      cur = cur ->
   cur -> link =
   return first
}
NODE invert_s(N
{
   NODE temp, cu
   temp = getnod
   temp -> info =
   temp -> link =
   if (second =
      retur
   cur = second
   while (cur =
      cur = cur
   cur -> link
   return sec
}
```

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

```
NODE delete_front (NODE record)
{
  NODE temp;
  if (record == NULL)
  {
    printf (" Queue is empty cannot delete \n");
     return record;
  }
  temp = record;
  temp = temp -> link;
  printf (" Item deleted is = %d \n", record -> info);
   free (record);
   return temp;
}
NODE delete_rear (NODE first)
{
    NODE cur, prev;
if (first == NULL)
{
  printf (" Stack is empty cannot delete \n");
   return first;
}
 if (first -> link == NULL)
{ printf (" Item deleted is = %d \n", first -> info);
   free (first);
   return NULL;
}
```

```
NODE delete_front (NODE record)
{
  NODE temp;
  if (record == NULL)
  {
    printf (" Queue is empty cannot delete\n");
     return record;
  }
  temp = record;
  temp = temp -> link;
  printf (" Item deleted is = %d\n", record -> info);
   free (record);
   return temp;
}
NODE delete_rear (NODE first)
{
    NODE cur, prev;
if (first == NULL)
{
  printf (" Stack is empty cannot delete\n");
   return first;
}
 if (first -> link == NULL)
{ printf (" Item deleted is %d\n", first -> info);
   free (first);
   return NULL;
}
```

```
int main()
{
   int item, choice, f;
   for(;;)
   { printf("\n1. Stack\n2. Queue\n3. exit");
     scanf("%d",&f);
   if(f==1)
   {
     printf("\n Stack:-");
   for(;;)
   {
     printf("\n1. Push\n2. Pop\n3- Display\n4. Return to
   Main menu\n5. Exit\n");
     printf("Enter the choice:");
     scanf("%d", &choice);
     switch(choice)
   {
   Case 1: printf("Enter the item: ");
           scanf("%d", &item);
           first = insert_rear(first, item);
           break;
   Case 2: first = delete_rear(first);
           break;
   Case 3: printf("Stack: \n");
           display(first);
           break;
   Case 4: main();
   Case 5: exit(0);
```

```
default : printf ("Enter valid i/p \n");
 break;

 }

}
}if (f==2){
  printf ("\n Queue :- ");
for(;;){
printf("\n1. Push \n2. Pop \n3. Display \n4. Return to
      Main Menu \n5. Exit\n");
printf("Enter the choice \n");
  scanf ("%d", &choice);
  switch (choice)
{
      case 1 : printf ("Enter the item: ");
        scanf("%d", &item);
       rcond = insert_rear [rcond, item);
        break;
case 2: rcond = delete_front (rcond);
        break;
case 3: printf ("Queue: \n");
        disp (rcond);
       break;
Case 4: main();
case 5 : exit(0);
default: printf ("Enter correct instruction \n");
   break;
}
}
```

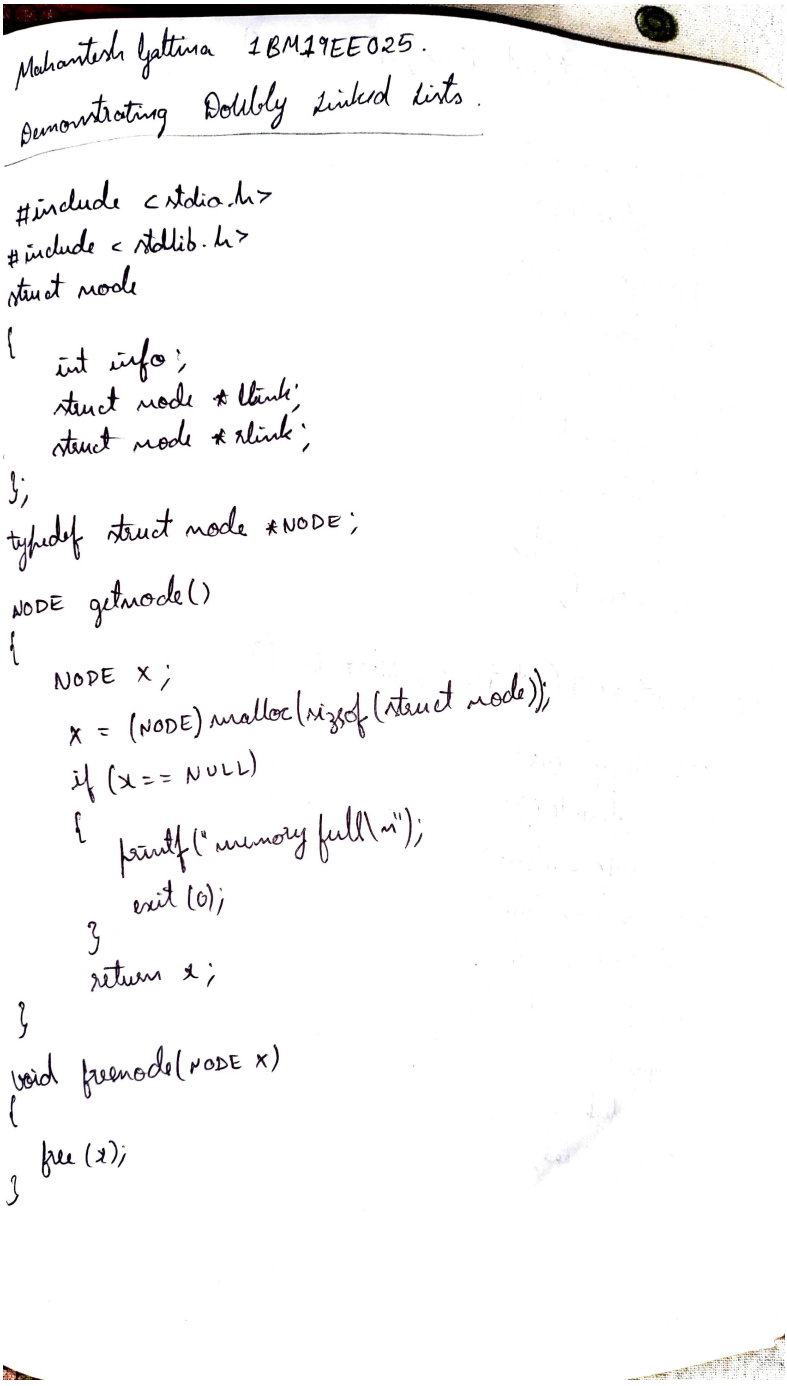
```
else if(f==3)
{
  exit(0);
}
else {
    printf ("Enter proper choice...");
    }
}
return 0;
}
```

Expected output:

```
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data
Structures/lab/lab10/Stack n Ques$ gedit list.c&
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data
Structures/lab/lab10/Stack n Ques$ make list
cc
list.c
-o list
[1]+ Done
gedit list.c
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data
Structures/lab/lab10/Stack n Ques$ ./list
1.Stack
2.Queue
3.exit
1
Stack:-
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice1
Enter the item : 1
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice1
Enter the item : 2
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice1
Enter the item : 3
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice2
Item deleted is 3
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice3
Stack :
1
2
1.Push
2.Pop3.Display
```

4.Return to Main Menu
5.Exit
Enter the choice4
1.Stack
2.Queue
3.exit
2
Queue:-
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
1
Enter the item : 1
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
1
Enter the item : 3
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
1
Enter the item : 5
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
3
Queue :
1
3
5
1.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
2
Item deleted is=11.Push
2.Pop
3.Display
4.Return to Main Menu
5.Exit
Enter the choice
5
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data Structures/lab/lab10/Stack n Ques$

Q 7.WAP Implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list



Mahantesh Gattina 1BM19EE025.

Demonstrating Doubly Linked Lists.

```
#include <stdio.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 full\n");
      exit(0);
   }
   return x;
}
void freenode(NODE x)
{
  free(x);
}
```

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

```
NODE ddelete_front(NODE head)
{
    NODE cur, next;
    if (head->rlink == head)
    {
        printf("dll empty\n");
        return head;
    }
    cur = head->rlink;
    next = cur->rlink;
    head->rlink = next;
    next->llink = head;
    printf("the node deleted is %d\n", cur->info);
    freenode(cur);
    return head;
}
NODE ddelete_rear(NODE head)
{
    NODE cur, prev;
    if(head->rlink == head)
    {
        printf("dll empty\n");
        return head;
    }
    cur = head->llink;
    prev = cur->llink;
    head->llink = prev;
    prev->rlink = head;
    printf("the node deleted is %d", cur->info);
    freenode(cur);
}   return head;
```

```
o  insert_leftpos( int item, NODE
{
   NODE temp, cur, prev;
   if (head->rlink == head)
   {
     printf("list 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 towards left of                item);
   temp = getnode();
   scanf("%d", &temp->info);
   prev->rlink = temp;
   temp->llink = prev;
   cur->llink = temp;
   temp->rlink = cur;
   return head;
}
```

```
NODE insert_rightpos(int item, NODE head)
{
  NODE temp, cur, next;
  if(head->rlink == head)
  {
    printf("list empty\n");
    return head;
  }
  cur = head->rlink;
  while(cur != head)
  {
    if(item == cur->info) break;
    cur = cur->link;
  }
  if(cur == head)
  {
    printf("key not found\n");
    return head;
  }
  next = cur->rlink;
  printf("enter towards the right of %d = ", item);
  temp = getnode();
  scanf("%d", &item->info);
  cur->rlink = temp;
  temp->llink = cur;
  temp->rlink = next;
  next->llink = temp;
  return head;
}
```

```
search                                  head)
{
    NODE temp;
    if(head->rlink = head)
    {
        printf("dll empty\n");
        return;
    }
    temp = head->rlink;
    while(temp != head)
    {
        if (temp->info == item)
        { printf("Element found\n");
            return;
        }
        temp = temp->rlink;
    }
    printf("Element not found !!!
}
Delete_duplicates(int item,                    head)
{
    NODE prev, cur, next;
    int count;
    if (head->rlink == head)
    {
        printf("List empty\n");
        return;
    }
```

```
count = 0;
cur = head -> rlink;
while (cur != head)
{   if (item != cur -> info)
      cur = cur -> rlink
      else
      {
          count++;
          if (count > 1)
          {
              prev = cur -> llink;
              next = cur -> rlink;
              prev -> rlink = next;
              next -> llink = prev;
              freenode (cur);
              cur = next;
          }
          else
          {
            cur = cur -> rlink;
          }
      }
}
 if (count == 0)
    printf(" key not found");
 else if(count == 1)
    printf(" duplicates not found\n");
 else
 printf (" duplicates found at %d positions and are
deleted\n", count-1);
}
```

```
void display (NODE head)
{
    NODE temp;
    if (head->rlink == head)
    {
        printf("dll empty \n");
        return;
    }
    printf("contents of dll \n");
    temp = head->rlink;
    while (temp != head)
    {
        printf("%d", temp->info);
        temp = temp->rlink;
    }
    printf("\n");
}
void main()
{
    NODE head, last;
    int item, choice;
    head = getnode();
    head->rlink = head;
    head->llink = head;
    for (;;)
    {   printf("\n1: insert front \n2: insert rear \n3: delete
front \n4 delete rear \n5: insert to the left of the key \n6: insert
to the right of the key \n7: search \n8: delete duplicates
```

```
\n9: display \n10: exit \n");
printf(" enter the choice \n");
scanf("%d", &choice);
switch (choice)
{   case 1: printf(" Enter the item at the front end \n");
            scanf("%d", &item);
            last = dinsert_front(item, head);
            break;
    case 2: printf(" enter the item at rear end \n");
            scanf("%d", &item);
            last = dinsert_rear(item, head);
            break;
    case 3: last = ddelete_front(head);
            break;
    case 4: last = ddelete_rear(head);
            break;
    case 5: printf(" enter the key item \n");
            scanf("%d", &item);
            last = insert_leftpos(item, head);
            break;
    case 6: printf(" enter the key item \n");
            scanf("%d", &item);
            search(item, head);
            break;
    case 8: printf(" enter the item \n");
            scanf("%d", &item);
            delete_duplicates(item, head);
            break;
    case 9: display(head);
            break;
```

```
        case 10: exit (0);
        default: exit (0);
    }
  }
}
```

Expected output:

```
1: insert front
2: insert rear
3: delete front
4: delete rear
5: insert to the left of the key
6: insert to the right of the key
7: search
8: Delete duplicates
9: display
10: exit
enter the choice
1
enter the item at the front end
1
1: insert front
2: insert rear
3: delete front
4: delete rear
```

5 : insert to the left of the key
6 : insert to the right of the key
7 : search
8 : Delete duplicates
9 : display
10 : exit
enter the choice
10.

Expected output:

mahantesh@mahantesh-Inspiron-15-3567:~/Data Structures/lab/lab9$ ./dlist
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
1
enter the item at front end

1
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
1
enter the item at front end
1
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
1
enter the item at front end
1
1:insert front
2:insert rear
3:delete front

4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
1enter the item at front end
2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
1
enter the item at front end
3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates

9:display
10:exit
enter the choice
1
enter the item at front end
4
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
2
enter the item at rear end
3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice2
enter the item at rear end

5
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
2
enter the item at rear end
8
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
9
contents of dll
432111358
1:insert front
2:insert rear
3:delete front

4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
8
enter the item
1
dupliactes found at 2 positions and are deleted
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display10:exit
enter the choice
9
contents of dll
4321358
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key

7:search
8:Delete duplicates
9:display
10:exit
enter the choice
3
the node deleted is 4
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
4
the node deleted is 8
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
5

enter the key item
12
key not found
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice5
enter the key item
5
enter towards left of 5=10
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
6
enter to the key item
5
enter towards right of 5 = 12

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
9
contents of dll
3 2 1 3 10 5 12
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
7
enter the key item to be searched
100
Element not found!!!!
1:insert front
2:insert rear
3:delete front

4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates9:display
10:exit
enter the choice
7
enter the key item to be searched
3
Element found
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert to the left of the key
6:insert to the right of the key
7:search
8:Delete duplicates
9:display
10:exit
enter the choice
10
mahantesh@mahantesh-Inspiron-15-3567:~/Data Structures/lab/lab9$

Q 8. Write a c program to demonstrate binary tree.

```
Demonstrating binary Tree ~ Mahantesh Gattina 1BM19EE0
#include <stdio.h>
#include <stdlib.h>
#include <string.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\n");
        exit(0);
    }
    return x;
}
void freenode(Node x)
{
    free(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 the 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\n");
    freenode(temp); return root;
  }
```

```
void display(NODE root, int i)
{ int j;
  if (root != NULL)
  { display (root->rlink, i+1);
  }
  return(root);
}
void preorder(NODE root)
{
   if(root != NULL)
   {
      for(j=1; j<=i; j++)
         printf(" ");
      printf("%d\n", root->info);
   }
   display(root->llink, i+1);
}
void main()
{
  for(;;)
  { printf("1. insert \n2. preorder \n3. inorder \n4. postorder
\n5. display\n");
    printf("enter the choice\n");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1: printf("enter the item\n");
                scanf("%d",&item);
                root = insert(item, root);
                break;
```

```
case 2 : if(root==NULL)
         {
           printf("tree is empty\n");
         }
         else
         {
           printf("given tree is");
           display(root, 1);
           printf("the preorder traversal is\n");
           preorder(root);
         }
         break;
case 3 : if(root==NULL)
         {
           printf("tree is empty\n");
         }
         else
         {
           printf("given tree is");
           display(root, 1);
           printf("the inorder-traversal is\n");
           inorder(root);
         }
         break;
case 4 : if(root==NULL)
         {
           printf("tree is empty\n");
         }
         else
         {
```

```
        printf("given tree is");
        display(root, 1);
        printf("the postorder traversal is \n");
        postorder(root);

    }
    break;
case 5: display(root, 1);
        break;
default: exit(0);
    }
  }
}
```

Expected output -

1. insert
2. preorder
3. inorder
4. postorder
5. display

enter the choice

1

enter the item

1

1. insert
2. preorder
3. inorder
4. postorder

5. display
enter the choice
1
enter the item
2
give the direction to insert
l
1. insert
2- preorder
3. inorder
4. postorder
5. display
enter the choice
1
enter the item
3
give the direction to insert
1
1. insert
2. preorder
3. inorder
4. postorder
5. display
enter the choice
2
given tree is
3
1
2
the preorder traversal is

1
2
3

1. insert
2. preorder
3. inorder
4. postorder
5. display
enter the choice
3
given tree is

3
1
2

the inorder traversal is
2
1
3

1. insert
2- preorder
3- inorder
4- postorder
5. display
enter the choice
4
given tree is

3
1.
2

the post order traversal is
2
3
1

Expected output:

```
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data
Structures/lab/lab10/binary_tree$ ./binarytree
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
2
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
3
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
4
give direction to insert
ll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
5
give direction to insert
lr
1.insert
2.preorder
3.inorder
4.postorder
```

```
5.display
enter the choice
1
enter the item
6
give direction to insert
rl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
7
give direction to insert
rr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
given tree is
        7
    3
        6
  1
        5
    2
        4
the preorder traversal is
1
2
4
5
3
6
7
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
given tree is
        7
    3
        6
  1
        5
    2
        4
the inorder traversal is
4
2
```

```
5
1
6
3
7
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
given tree is
        7
    3
        6
  1
        5
    2
        4
the postorder traversal is
4
5
2
6
7
3
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
        7
    3
        6
  1
        5
    2
        4
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
0
mahantesh@mahantesh-Inspiron-15-3567:~/Desktop/Notes/Data
Structures/lab/lab10/binary_tree$
```