DS LAB PROGRAM

Page No.
Date

[HB-]

NAME: ANSHUL H. S
USN: 1BM19CS020

1) Write a program to stimulate 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 size 3
int top=-1;
void push (int [], int);
int pop (int []);
void display (int []);
int main (int argc, char ** argv)
{
    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 in);
```

Anshul H. S

```
scanf( "%d", &element);
push( stack, element);
break;
case 2 : element= pop(stack);
        if (element == -1).
        printf( " Stack Underflow").
        else
        printf(" poped element is %d \n", element);
        break;
case 3: display (stack);
        break;
  default : printf (" Invalid choice ");
}

printf(" Do you want to continue: \n");
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");
    }
  else
  {
    top++;
    stack [top]=ele;
  }
}
```

Anshul H. Surana

NAME: ANCHAL H SUPANA
USN: 1BM19CS050

```
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 \t", stack[i]);
    }
}
```


Anchal H Supana

input

```
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):1

Enter element to push:10

** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):1

Enter element to push:20

** Stack Menu **

1.Push
2.Pop
3.Display
```

input

```
Enter your choice(1-4):1

Enter element to push:20

** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

 Enter your choice(1-4):1

Enter element to push:30

** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

 Enter your choice(1-4):3

 Stack is...
 30
 20
 10
```

input

```
30
20
10

** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):2

Deleted element is 30
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):2

Deleted element is 20
** Stack Menu **

1.Push
2.Pop
```

input

```
Deleted element is 30
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):2

Deleted element is 20
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):2

Deleted element is 10
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit
```

input

```
Deleted element is 10
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):2

Stack is empty!!
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):3

Stack is empty!!
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit
```

input

```
3.Display
4.Exit

Enter your choice(1-4):2

Stack is empty!!
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):3

Stack is empty!!
** Stack Menu **

1.Push
2.Pop
3.Display
4.Exit

Enter your choice(1-4):4


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

LAB-2

Name: [illegible]
USN: [illegible]

1) 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)

```
#include <stdio.h>
#define MAX 100
char stack [MAX];
int top=-1;
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(" Stack is empty \n");
  else
  {  item= stack [top];
     top--;
     return item;
  }
}
```

Arjun H Suresh

NAME: ANCHUL H SURANA
USN: 1BM19CS020

```
int stackempty()
{ if (top==-1)    return 1;
        else      return 0;
}
char stacktop()
{ if (top==-1)
  printf("\n Stack is empty \n");
  else
  return stack[top];
}
int priority (char ch)
{   switch (ch)
    { case '+';
      case '-';   return (1);
      case '*';
      case '/';   return (2);
      Case '^';   return (3);
      default : return(0);
    }
}
int main (int argc, char ** argv)
{    char infix[100];
     int i, item;
     printf("Enter the infix expression:");
     scanf("%s", infix);
     printf("Expression: %s", infix);
     printf("\n Postfix: ");
     i=0;
     while (infix[i]!='\0')
{    cap'(' push (infix[i]);
     break;
```

Anchul H. Surana

Name: Anshul H Saxena
Uni: 1BM19CS020

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

Anshul H Saxena

Scanned by TapScanner

input

```
Enter the infix expression :a+b-c-d*e+f
Expression : a+b-c-d*e+f
 Postfix: ab+c-de*-f+


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

input

```
Enter the infix expression :a*b^c+d^e-f+g
Expression : a*b^c+d^e-f+g
 Postfix: abc^*de^+f-g+


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

NAME: ANSHUL H. SUPANA
USN: 1BM19CS020

LAB-3

Q-3) WAP to stimulate 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 & queue full condition.

→
```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int queue[N];
void enq (int a, int *front, int *rear)
{   if (rear == N-1)
        printf("\n QUEUE IS FULL");
    else
    { if (*front == -1)
        *front = 0;
      (*rear)++;
      queue[*rear] = a;
      printf("\n Insertion Done !"); }}

void deq (int *front, int *rear)
{   if (*front == *rear)
        printf("\n Queue is empty");
    else
    { printf("\n deleted element %d", queue[*front]);
      *front = *front + 1;
    }}
void display (int *front, int *rear)
{   if (*rear == -1)
        printf("\n QUEUE IS EMPTY");
    else { int i;
```


Anshul H. Supana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
printf ("\n QUEUE ELEMENTS ARE:\t");
for (i=*front ; i<=*rear; i++)
printf (" %d \t", queue[i]); }}

int main()
{ int a, choice;
 int *front=-1, *rear=-1;
while (1)
{ printf ("\n ENTER \n");
  printf ("\n To INSERT ");
  printf ("\n To DELETE ");
  printf ("\n To DISPLAY");
  printf ("\n To EXIT);
  printf ("\n Enter your choice");
  scanf ("%d", &choice);
  switch (choice)
{ case 1:
  printf ("Enter the value to inserted");
  scanf ("%d", &a);
  enq (a, &front, &rear);
  break;
case 2:
  deq (&front, &rear);
  break;
case 3:
  display (&front, &rear);
  break;
case 4:
  return 0;
default:
  printf ("\n INVALID");
} } return 0;
```

Anshul H. Surana

input

```
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
1
Enter the value to insert
20

 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
1
Enter the value to insert
40

 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
```

input

```
 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
1
Enter the value to insert
60

 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
1
Enter the value to insert
80

 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
```

input

```
 Insertion Done!
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
3

 QUEUE elements are :   20      40      60      80
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 20
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 40
```

input

```
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 20
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 40
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 60
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
```

input

```
2

 deleted element 20
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 40
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 deleted element 60
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
3
```

input

```
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
3

 QUEUE elements are :   80
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
2

 QUEUE is empty
 ENTER
 1.TO INSERT
 2.TO DELETE
 3.TO DISPLAY
 4.TO EXIT
 Enter your choice
4


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

NAME: ANSHUL H
USN: 1BM19CS020

LAB-4

4) WAP to simulate the working of a circular queue of ~~integers~~ 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 size 3
int front = -1;
int rear = -1;
void insert (int, int[]);
int delete (int[]);
void display (int[]);
int main()
{
    int ch, queue[size];
    int item;
    do{
printf("\n 1. Insert to Queue:");
printf("\n 2. delete from Queue:");
printf("\n 3 Display Queue");
printf("\n 4. Exit\n");
printf("Enter the option:");
scanf("%d", &ch);
switch(ch)
{ case 1: printf("Enter the element\n");
          scanf("%d", &item);
```


Anshul H

NAME: ANJALI H SURANA
USN: 1BM19CS020

```
        insert(item, queue);
        break;
case 2: item = delete(queue);
        if (item == -9999)
        printf("Queue Underflow\n");
        else
        printf("Removed element: %d", item);
        break;
        else
case 3: display(queue);
        break;
case 4: exit(0);
    }
} while (ch!=4);
    return 0;
}
void insert(int ele, int queue[])
{   if ((rear+1) % size == front)
        printf("Queue Overflow\n");
    else
{   rear = (rear+1) % size;
    queue[rear] = ele;
    if((rear+1) % size == front)
    printf("Queue is full\n");
    if (front == -1)
        front=0;
}
}
int delete(int queue[])
{   int item;
    if(front == -1)
    return -9999;
```

Anjali H Surana

```
else
{ item = queue[front];
    if (front == rear)
{ front = -1;
    rear=-1;
  } else {
{ front = (front+1) %size; }
    return item;
  }
}
}
void display (int queue[7])
{ int i;
 if (front==-1) && (rear==-1))
 printf(" Queue is empty\n");
 else
 { if (front <= rear)
   for(i=front; i<=rear; i++)
   printf("%d", queue[i]);
else
{
   for( i=front ; i<size; i++)
   printf("%d", queue[i]);
   for (i=0; i<=rear; i++){
   printf("%d", queue[i]);
    }

  }
}
```


Anshul H Surana

input

```
 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :1
Enter the element
10


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :1
Enter the element
20


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :1
Enter the element
30
Queue is full
```

input

```
30
Queue is full


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :3
102030

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Removed element: 10

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Removed element: 20

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
```

input

```
 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Removed element: 20

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Removed element: 30

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Queue undreflow


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :3
```

input

```
Enter the option :2
Removed element: 30

 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :2
Queue undreflow


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :3
Queue is empty


 1. Insert to Queue:
 2. delete from the Queue:
 3. Display Queue
 4. Exit
Enter the option :4


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

LAB-5

NAME: ANSHUL H SUBANA
USN: 1BM19CS026

Q) WAP to Implement Singly Link list with following operations
a) Create a linked list b) Insertion of a node at first postion, at any position and at the end of list c) Display the contents of the linked list.

→
```
#include <stdio.h>
#include <stdlib.h>
struct node
{ int id;
  char name [20];
   int sem;
   struct node * next;
};
struct node * head = NULL;
void linkedlist ();
void insertNodeAtBegn();
void insertNodeAtEnd();
void insertNodeAtAny();
void displayList();

int size=0;
int main()
{
   linkedlist();
   return 0;
}
```

Anshul H. Subana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

```
void linkedList()
{  int choice1, choice2;
   printf("\n Enter the operation");
   printf("\n 1. Insert Node  \n 2. Display List \n Enter
                \n Choice : ");
   scanf("%d", &choice1);
   switch (choice1)
{ case 1: printf ("\n 1. At the First Position \t
                2. At End of List \t 3. At any Specified
   Location\n Choice :");
   scanf("%d", &choice2);
   switch (choice2)
{  case 1: insertNodeAtBegin();
           break;
   case 2: insertNodeAtEnd();
           break;
   case 3: insertNodeAtAny();
           break;
   default: printf ("\n Input Error, try Again!\n");
           linkedlist();
} break;
case 2: displaylist();
        break;
case 3: exit(0);
default: printf("\n Input error, Try again!\n");
        linkedlist();
}
}
void insertNodeAtBegin()
{ struct node* newnode;
  newnode = (struct node*) malloc (sizeof(struct node));
```


Anshul H. Surana

```
printf("\n Enter the Details");
printf("\n ID ");
scanf("%d", &newnode->data);
printf("\n Name ");
scanf("%s", newnode->name);
printf("\n Sem ");
scanf("%d", &newnode->sem);

newnode->next=head;
head=newnode;
cnt++;
printf("\n Node Created \n");
linkedlist();
}
void insertatend()
{ struct node *newnode, *temp;
newnode=(struct node*) malloc(sizeof(struct node));
printf("\n Enter the Details");
printf("\n ID ");
scanf("%d", &newnode->data);
printf("\n Name ");
scanf("%s", newnode->name);
printf("\n Sem ");
scanf("%d", &newnode->sem);

if (head==NULL)
{ newnode->next=NULL;
head=newnode;
printf("\n Node Created \n");
linkedlist();
}
for(temp=head; temp->next!=NULL; temp=temp->next);
newnode->next=NULL;
temp->next=newnode;
cnt++;
printf("\n Node Created");
linkedlist();
}
```


Assignment

NAME: ANCHAL H. SHARMA
USN: 1BM19CS020

```
void insertNodeAtAny()
{ struct node * newnode, * temp=head;
  newnode = (struct node*) malloc (size of (struct node));
  printf(" Enter the Details");
   printf("\n ID: ");          scanf("%d", &(newnode->id));
   printf(" Name:");           scanf("%s", (newnode->name));
   printf(" Sem: ");           scanf("%d", (newnode->sem));

  int pos=0, i=0;
   printf("\n Enter the position (pos>=1 and pos<%d)",
                                          size);
   scanf("%d", &pos);
     if(pos==0)
  { printf("\n Error position, check the operation once");
       LinkedList();
  }
  for(temp; temp->next!=NULL; temp= temp->next)
  { if(i==(pos-1))
    { newnode->next= (temp->next);
      temp->next = newnode;
      size++;
      printf("\n Node Created\n");
       LinkedList();
    }
     i++;
     temp= temp->next;
  }
}

void displayList()
{ if (head==NULL)
  { printf("\n Empty List\n");
      LinkedList();
  }
```


Anchal H Sharma

NAME : ANSHUL H. SURANA
USN : 1BM19CS020

Page No
Date

```
printf("\n The List is:");
for(struct node * temp = head; temp != NULL;
                    temp=temp->next)
{ printf("\n Student Details ");
printf("\n\nID : %d", temp->id);
printf("\n Name : %s", temp->name);
printf("\n Sem : %d", temp->sem);
}
linkedlist();
}
```

Anshul H. Surana

```
<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 1

1.At the First Position          2.At End of list          3.At Any Specified Location
Choice:1

<--Enter the Details-->
ID: 10001
Name: Rohan
Sem: 3

Node created

<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 1

1.At the First Position          2.At End of list          3.At Any Specified Location
Choice:2

<--Enter the Details-->
ID: 10002
```

```
<--Enter the Details-->
ID: 10002
Name: Ketan
Sem: 5

Node created

<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 1

1.At the First Position         2.At End of list         3.At Any Specified Location
Choice:1

<--Enter the Details-->
ID: 10003
Name: Yash
Sem: 2

Node created

<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
```

```
1.At the First Position         2.At End of list        3.At Any Specified Location
Choice:1

<--Enter the Details-->
ID: 10003
Name: Yash
Sem: 2

Node created

<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 2

The List is :
<---Student Details--->
ID: 10003
Name: Yash
Sem: 2
<---Student Details--->
ID: 10001
Name: Rohan
Sem: 3
<---Student Details--->
ID: 10002
```

input

```
Sem: 2

Node created

<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 2

The List is :
<---Student Details--->
ID: 10003
Name: Yash
Sem: 2
<---Student Details--->
ID: 10001
Name: Rohan
Sem: 3
<---Student Details--->
ID: 10002
Name: Ketan
Sem: 5
<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 1
```

input

```
Sem: 2
<---Student Details--->
ID: 10001
Name: Rohan
Sem: 3
<---Student Details--->
ID: 10002
Name: Ketan
Sem: 5
<---Enter the Operation--->
1.Insert Node.
2.Display List
3.Exit.
Choice: 1

1.At the First Position         2.At End of list        3.At Any Specified Location
Choice:3

<--Enter the Details-->
ID: 10004
Name: Mohan
Sem: 8

Enter the position(pos>=1 and pos <3) : 2


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

Page No.
Date / /

LAB 6

NAME: ANSHUL H. SUBHRA
USN: 1BM19CS020

Q 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.

→
```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int id;
    char name[20];
    int sem;
    struct node * next;
};
struct node * head = NULL;

void linkedList();
void insertNode();
void deleteNode(int);
void deleteNodeAtBegin();
void deleteNodeOfGiven();
void displayList();

int size=0;
int main()
{   linkedList();
    return 0;
}
```


Anshul H. Subhra

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
void linkedList()
{ int choice1, choice2;
  printf("\n Enter the Operation");
  printf("\n Insert Node \n 2. Delete node \n
    3. Display list \n 4. Exit. \n Choice");
  scanf("%d", &choice1);
  switch (choice1)
{ case 1 : insertNode();
           break;
  case 2: printf("\n 1. At the First Position
  \n 2. At End of List \t 3. At Given
  Element Position\n Choice:");
  scanf("%d", &choice2);
  switch (choice2)
{ case 1: deleteNode(1);
          break;
  case 2: deleteNode(2);
          break;
  case 3: deleteNode(3);
          break.
default: Printf("\n Input Error, try Again \n");
         linkedList();
} break;
case 3: displayList();
        break;
case 4: exit(0);
default : printf("\n Input error, Try again \n");
          linkedList();
}
}
```

Anshul H. Surana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
void insertNode()
{ struct node *newnode *temp;
  newnode =(struct node*) malloc (sizeof (struct node));
  printf("\n Enter the Details");
  printf("\n ID:");
  printf("Name:");
  printf("Sem:");
  scanf("%d", &(newnode->id));
  scanf("%s", (newnode->name));
  scanf("%d", &(newnode->sem));

  if (head==NULL)
{ newnode->next=NULL;
    head = newnode;
    printf("\n Node Created");
    linkedList();
    size++;
}
for(temp=head; (temp->next)!=NULL ; temp=(temp->next));
  newnode->next=NULL;
{
    temp->next= newnode;
    size++;
    printf("\n Node Created");
    linkedlist();
  }
}
void deleteNode (int flag)
{  if (head==NULL)
   {
    printf("\n The list is Empty\n");
    linkedlist();
}
else
if (head->next==NULL)
```


Anshul H. Surana

NAME: ANSHUL H. SUBANNA
USN: 1BM19CS020

```
{ printf("\n Node Deleted. \n Now List is empty");
    free(head);
    head = NULL;
    linkedlist();
}
else
{ switch (flag)
  { case 1: deleteNodeAtBegin();
            break;
    case 2: deleteNodeAtEnd();
            break;
    case 3: deleteNodeOfGiven();
            break;
  }
  linkedlist();
}
void deleteNodeAtBegin()
{ struct node* temp = head;
  head = head->next;
  free(temp);
  printf("\n Node Deleted");
}
void deleteNodeAtEnd()
{ struct node* temp = head;
  head = head->next;
  free(temp->next); temp->next = NULL;
  printf("\n Node Deleted");
}
void deleteNodeOfGiven()
{ struct node* temp1 = head, * temp2 = temp1;
```


Anshul H. Subanna

NAME: ANSHUL H. SUBANNA
USN: 1BM19CS020

```
int ele;
printf("\n Enter the element");
scanf("%d", &ele);
for(temp1; temp1->next!=NULL; temp1=temp1->next)
{   temp2 = temp1->next;
    if(temp2->id==ele)
    {  temp1->next=temp2->next;
       free(temp2);
       printf("\n Node Deleted");
       return;
    }
if(temp1->id==ele)
{   head= temp2;
    free(temp1);
    printf("\n Node Deleted");
    return;
}
   }
   printf("\n Element Not present in the List");
    linkedList();
}
void displayList()
{ if(head==NULL)
   { printf("\n Empty List \n");
      linkedList();  }
    printf("\n The List is:");
    for(struct node * temp=head; temp!=NULL; temp=temp->next
{ printf("\n Student Details");
  printf("\n ID: %d", temp->id);
  printf("\n Name: %s", temp->name);
  printf("\n Sem: %d", temp->sem);
}  linkedList();
}
```


Anshul H. Subanna

```
<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 1

<--Enter the Details-->
ID: 10001
Name: Anil
Sem: 2

Node created


<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 1

<--Enter the Details-->
ID: 10002
Name: Rohit
Sem: 3
```

```
<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 1

<--Enter the Details-->
ID: 10003
Name: Mayank
Sem: 8

Node created


<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 3

The List is :

<---Student Details--->
ID: 10001
Name: Anil
Sem: 2
```

```
<---------x--------->

<---Student Details--->
ID: 10002
Name: Rohit
Sem: 3
<---------x--------->

<---Student Details--->
ID: 10003
Name: Mayank
Sem: 8
<---------x--------->

<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 2

1.At the First Position         2.At End of list        3.At Given Element Position
Choice:1

Node Deleted.

<---Enter the Operation--->
1.Insert Node.
```

input

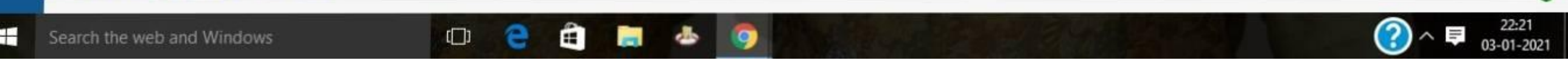
```
<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 3

The List is :

<---Student Details--->
ID: 10002
Name: Rohit
Sem: 3
<----------x---------->

<---Student Details--->
ID: 10003
Name: Mayank
Sem: 8
<----------x---------->

<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 2
```

input

```
4.Exit.
Choice: 2

1.At the First Position         2.At End of list        3.At Given Element Position
Choice:2

Node Deleted.

<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 3

The List is :

<---Student Details--->
ID: 10002
Name: Rohit
Sem: 3
<---------x---------->

<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
```

input

```
4.Exit.
Choice: 2

1.At the First Position          2.At End of list          3.At Given Element Position
Choice:1

Node Deleted.
Now List Is empty!

<---Enter the Operation--->
1.Insert Node.
2.Delete Node
3.Display List
4.Exit.
Choice: 3

Empty List!


 <---Enter the Operation--->
 1.Insert Node.
 2.Delete Node
 3.Display List
 4.Exit.
 Choice:

...Program finished with exit code 9
Press ENTER to exit console.
```



Page No.
Date

LAB-7.

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Q.7) WAP Implement Single Link list with following operations
a) Sort the linkedlist b) Reverse the linked list c) Concatenation of two linked lists

```
#include <stdio.h>
# include <stdlib.h>
 void sort()
 void create();
 void reverse();
 void create_second ();
  void concatenate();
  void display();
  struct node
{ int data;
    Struct node * next;
};
Struct node *head=NULL;
Struct node *head2 = NULL;
 int c;
 Int main()
{    int choice;
     do
{ printf("\n1.Create \n2. Sort \n3. Reverse \n4.
    Enter second list \n5. Concatenation \n6 Display
 \n7. Exit");
 scanf ("%d", &choice);
 Switch (choice)
{ case 1: create();
          break;
```

Anshul H. Surana

NAME : ANSHUL H. SUPANA
USN : 1BM19CS020

```
        case 1: sort();
                break;
        case 2: reverse();
                break;
        case 4: create_second();
                break;
        case 5: concatenate();
                break;
        case 6: display();
                break;
        case 7: exit(0);
    }
}while (choice!=7);
    Return 0;
}
void create()
{   struct node *newnode;
    struct node *temp;
    int s;
    printf("Enter integer");
    scanf("%d", &s);
  newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=s;
    if(head == NULL)
{   newnode->next=NULL;
    head=newnode;
    printf("First node created\n");
    ctf;
} else
{   temp=head;
    while(temp->next!=NULL)
{   temp=temp->next; }}
```

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

```
        temp -> next = newnode;
        newnode -> next = NULL;
        ctt;
        printf("Node Created\n");
    }
}
void reverse()
{ struct node * prev=NULL, * current = head, * next;
  while (current != NULL)
  { next = current -> next;
    current -> next = prev;
    prev = current;
    current = next;
  }
  head = prev;
  printf("The list is reversed\n");
}
void display()
{ struct node * ptr = NULL;
  ptr = head;
  if (ptr == NULL)
  { printf("List is empty\n");
  } else
  { printf("\n contents of the linked list
    while (ptr != NULL)
    {
      printf("\t %d", ptr -> data);
      ptr = ptr -> next;
    }
  }
  printf("\n");
}
```

Anshul H Surana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
void create_second()
{ struct node * newnode;
  struct node * temp;
    int s, y;

  printf ("Enter integer:");
  scanf ("%d", &s);
  newnode = (struct node*) malloc (size of (struct node
  newnode -> data = s;
   if (head2 == NULL)
{ newnode -> next = NULL;
   head2 = newnode;
   printf ("First node created \n");
   ct++;
} else
 { temp=head2;
  while (temp -> next!=NULL)
  { temp=temp -> next; }
  temp -> next = newnode;
  newnode -> next = NULL;
  ct++;
  printf (" Node created \n"); }
}
}
void concatenate()
{ struct node * ptr;
  if (head == NULL)
  { head2=head; }
  ptr=head;
while (ptr -> next != NULL)
  ptr = ptr -> next;
```

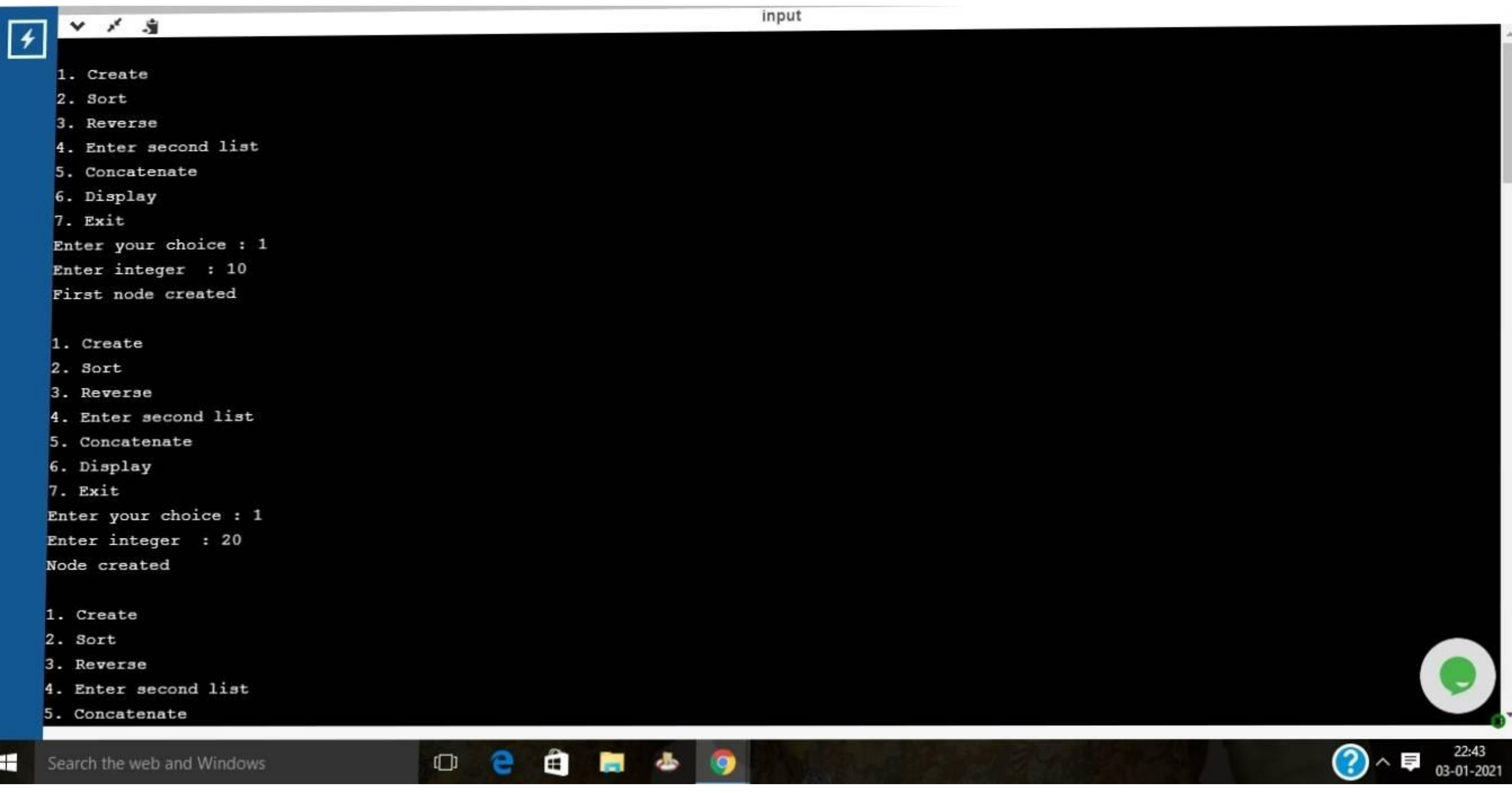

Anshul H. Surana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
        ptr -> next = head2;
        printf("The list is concatenated\n");
}
void sort()
{   int swap, i;
    struct node *ptr1;
    struct node *lptr = NULL;

    if (head == NULL)
        return;
do {
        swap = 0;
        ptr1 = head;
        while (ptr1 -> next != lptr)
    {   if (ptr1 -> data > ptr1 -> next -> data)
        {
            int temp = ptr1 -> data;
            ptr1 -> data = ptr1 -> next -> data;
            ptr1 -> next -> data = temp;
            swap = 1;
        }
        ptr1 = ptr1 -> next;
    }  lptr = ptr1;
}
    while (swap);
    printf("The list is sorted\n");
}
```

Anshul H. Surana



input

```
1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 1
Enter integer  : 30
Node created

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 2
The list is sorted

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
```

input

```
1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 6

Contents of the Linked List:    10      20      30

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 3
The list is reversed

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
```

input

```
1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 6

Contents of the Linked List:    30      20      10

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 4
Enter integer: 50
First node created

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
```

input

```
1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 1
Enter integer  : 60
Node created

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 3
The list is reversed

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
```

input

```
6. Display
7. Exit
Enter your choice : 3
The list is reversed

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 6

Contents of the Linked List:    60      10      20      30

1. Create
2. Sort
3. Reverse
4. Enter second list
5. Concatenate
6. Display
7. Exit
Enter your choice : 7


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

Search the web and Windows

22:46
03-01-2021

NAME: ANSHUL H. SURAN
USN: 1BM19CS020

LAB-8

Q-8) WAP to implement Stack & Queues using Linked Representation.

→ Stack

```
#include <stdio.h>
# include <stdlib.h>
 struct node * top= NULL;
  struct node
{     int data;
      struct node * next;
};
void stack();
void push(int);
int pop();
 void display();
 int main()
{     stack();
      return 1;
}
void stack()
{ int choice=0, ele=0;
  do
{ printf("\n Enter the choice: \n 1. Push.
  \n 2. Pop \n 3. Display \n 4. Exit \n Ch
scanf("%d", &choice);
switch(choice)
{ case 1: printf("\n Enter the element to Pu
          scanf("%d", &ele);
          push(ele);
```


Anshul H. Suran

```
        break;
case 2: ele = pop();
        printf("\n %d was deleted from Queue", ele);
        break;
case 3: display();
        break;
case 4: exit(0);
default:
        printf("\n Input Error Try Again");
        stack();
    }
}while(1);
}
void push (int ele)
{ struct node* newnode;
  newnode = (struct node*) malloc (sizeof (struct node));
  newnode -> data = ele;
  newnode -> next = NULL;
  if (top == NULL)
{ top = newnode; }
  else
{ newnode -> next = top;
    top = newnode;
}
printf ("\n element %d was inserted \n", ele);
}
int pop ()
{ int ele;
  struct node * temp;
  if (top == NULL)
{ printf("\n Stack Underflow, Stack is empty");
    stack(); }
```


Anshul H. Surana

```
    ele = top -> data;
    temp = top;

if (top -> next == NULL)
{   top = NULL; }
else
{   top = top -> next; }
    free (temp);
    return ele;
}
void display()
{   struct node * i;
    if (top == NULL)
  { printf("\n Stack is empty\n");
    stack();
  }
    Printf("\n Stack Contains: \n Top ->");
    for (i = top; i != NULL; i = i -> next)
      printf("\t %d", i -> data);
  }
}
```

Queue.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{ int data;
   struct node * next; };
void insert();
void display();
void del();
```


Anshul H. Surana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

Page No.
Date

```
struct node * rear = NULL, * front = NULL;

int main (int argc, char ** argv)
{   int choice;
    char ch = 'Y';
    do
{   printf("\n Queue implementation using linked list\n"
    printf("\n1. Create \n2. Display \n3. Delete
    \n4. Exit \n");
printf("\n Enter your choice");
scanf("%d", &choice);
switch (choice)
{ case 1: insert(); break;
    case 2: display(); break;
    case 3: del(); break;
    case 4:
        ch = 'n';
        break;
    }
}while (ch == 'y' || ch == 'Y');
}
void insert()
{   struct node* newnode;
    newnode = (struct node*) malloc (size of(struct node));
    printf("Enter the element: \n");
    scanf("%d", &newnode -> data);
    newnode -> next = NULL;

    if (rear == NULL)
{   rear = newnode;
    front = newnode;
}
```

Anshul H. Surana

```
else
{  rear->next = newnode;
    rear=newnode;
}
}
void del()
{ if (front==NULL)
  { printf("Queue is empty\n"); return;
  }
else
{ printf("Deleted ele is %d", front->data);
   if (front==rear)
   { printf("Queue is empty\n");
      front=NULL;    rear=NULL;
   }
else
   front= front->next;
  }
}
void display()
{
   struct node *temp;
   if (front== NULL)
{    printf("Queue is empty");
      return;
}
   temp= front;
   while (temp!=NULL)
{  printf("%d", temp->data);
   temp = temp->next;
}
}
```


Anshul H. Surana

input

```
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:1

Enter the element to Push: 10

Element 10 was inserted!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:1

Enter the element to Push: 20

Element 20 was inserted!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
```

input

```
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:1

Enter the element to Push: 30

Element 30 was inserted!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:1

Enter the element to Push: 40

Element 40 was inserted!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:3
```

input

```
3.Display.
4.Exit
Choice:3

The Stack Contains:
TOP->  40      30      20      10
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2

40 was deleted from Queue
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2

30 was deleted from Queue
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:3
```

input

```
4.Exit
Choice:3

The Stack Contains:
TOP->  20      10
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2

20 was deleted from Queue
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2

10 was deleted from Queue
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2
```

input

```
10 was deleted from Queue
Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:2

Stack UnderFlow, The Stack is empty!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:3

Stack is empty!

Enter the choice:
1.Push.
2.Pop.
3.Display.
4.Exit
Choice:4


...Program finished with exit code 0
```

input

```
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:1

Enter the element to Insert: 10

Element 10 was inserted!

Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:1

Enter the element to Insert: 20

Element 20 was inserted!

Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
```

input

```
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:1

Enter the element to Insert: 30

Element 30 was inserted!

Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:3

The Queue Contains:10 20 30
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:2

10 was deleted from Queue
Enter the choice:
```

Search the web and Windows

23:31
03-01-2021

input

```
10 was deleted from Queue
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:2

20 was deleted from Queue
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:3

The Queue Contains:30
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:2

30 was deleted from Queue
Enter the choice:
1.Insert.
```

input

```
The Queue Contains:30
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:2

30 was deleted from Queue
Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:3

Queue is empty!

Enter the choice:
1.Insert.
2.Delete.
3.Display.
4.Exit
Choice:4


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

Search the web and Windows

23:32
03-01-2021

NAME: ANKUL H SUKANA
USN: 1BM19CS020

LAB- 9.

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

```
#include <stdio.h>
#include <stdlib.h>
struct node
{   int data;
    struct node * next;
    struct node * prev;
};
struct node * head = NULL;
void insert_left()
{   struct node * new_node;
    new_node = (struct node*) malloc (sizeof (struct node));
    printf("Enter the item\n");
    scanf("%d", &newnode->data);
    new_node -> next = NULL;
    new_node -> prev = NULL;
    if (head == NULL)
    {   head = new_node; }
    else
    {   new_node -> next = head;
        head -> prev = new_node;
        head = new_node;
    }
```


Ankul H. Sukana

NAME: ANJALI H. SURANA
USN: 1BM19CS020

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


Anjali H. Surana

```
    if (temp==head)
    {   head=head→next; }
    else if (temp→next=NULL)
    {   temp=temp→prev;
        temp→next=NULL;
    }
    else
    {   temp→prev→next=temp→next;
        temp→next→prev=temp→prev;
    }
}
void display()
{   struct node * ptr;
    ptr=head;
    while (ptr!=NULL)
    {   printf("%d\t", ptr→data);
        ptr=ptr→next;
    }
    printf("\n");
}
int main()
{   int choice;
    while(1)
{   printf("1. Insert at left\n");
    printf("2. Insert at right\n");
    printf("3. Delete\n");
    printf("4. Display\n");
    printf("5. Exit\n");
    scanf("%d", &choice);
    switch(choice)
{   case 1: insert_left();
        break;
```

Anshul H. Surana

NAME: ANSHUL H. SURANA
USN: 1BM19CS010

```
        case 2: insert_right();
                break;
        case 3: del();
                break;
        case 4: display();
                break;
        case 5: exit(0);
    }
}
```


Anshul H. Surana

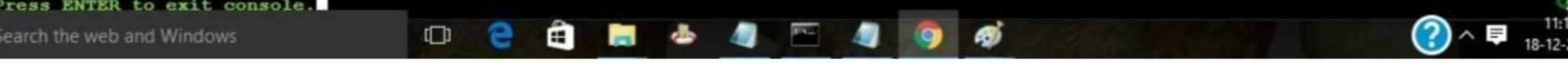
```
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
1
Enter the item
10
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
1
Enter the item
20
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
1
Enter the item
30
 1. Insert at the left
```

```
Enter the item
30
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
2
Enter the item
40
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
3
Enter the element to be deleted
20
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
4
30      10      40
```

```
Enter your choice
4
30      10      40
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
1
Enter the item
50
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
4
50      30      10      40
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
2
Enter the item
```

```
Enter your choice
2
Enter the item
60
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
4
50      30      10      40      60
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
3
Enter the element to be deleted
10
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
3
```

```
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
3
Enter the element to be deleted
60
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
4
30      40      60
 1. Insert at the left
 2. Insert at the right
 3. Delete
 4. Display
 5. Exit
Enter your choice
5


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



Page No.
Date / /

NAME: ANUSHA H SHARMA
USN: 1BM19CS020

LAB-10.

8-10) Write a program
a) To construct a binary search Tree
b) To traverse the tree using all the methods i.e. in order, preorder & post order.
c) To display the elements in the tree.

→
```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
    int data;
    struct node * left;
    struct node * right;
} Node;
void tree();
Node * create();
Node * insert (Node *, Node *);
void traverse();
void preorder (Node *);
void inOrder (Node *);
void postOrder (Node *);
void display (Node *, int);

Node * root;
int main()
{
    tree();
```

Anushka H. Sharma

```
    return 0;
}
void tree()
{   int choice;
    printf("Binary Search Tree \n 1. Insert Elem
    \n 2. Traverse All methods \n 3. Displa
    BST\n 4. Exit\n Choice");
    scanf("%d", &choice);
    switch (choice)
{ case 1: insert(root, create());
        break;
case 2:  traverse();
        break;
case 3:  if (root==NULL)
    printf("\n Tree is empty!");
    else
        display (root, 0);
        break;
case 4: exit(0);
        break;
default: printf("\n Error choice \n");
        tree();
  } tree();
}
Node * create()
{ Node * newnode = (Node*) malloc (sizeof(Node));
  printf("\n Enter the element:");
  scanf("%d", &newnode→data);
  newnode→left = NULL;
  newnode→right = NULL;
  return newnode;
}
```


Anshul H. Supana

NAME: ANSHUL H. SUPANA
USN °. 1BM19CS020

Page No.
Date

```
Node * insert (Node *Root, Node *newNode)
{   if (root == NULL)
    {   root = newnode;
        printf("\n Root Node Created");
    } else
    {   if (Root → right = NULL)
        {
            Root → right = newNode;
        }
        else
        insert (Root → right, newNode);
    }
    else
    if (newNode → data < Root → data)
{   if (Root → left == NULL)
    {   Root → left = newNode;
    }
        else
        insert (Root → left, newNode);
    }
}
}
void traverse()
{   if (root == NULL)
    {   printf("\n The tree is Empty");
        return;
    }
    printf("\n Pre Order Traverse:");
    preOrder(root);
    printf("\n Inorder Traverse:");
    inOrder (root);
```

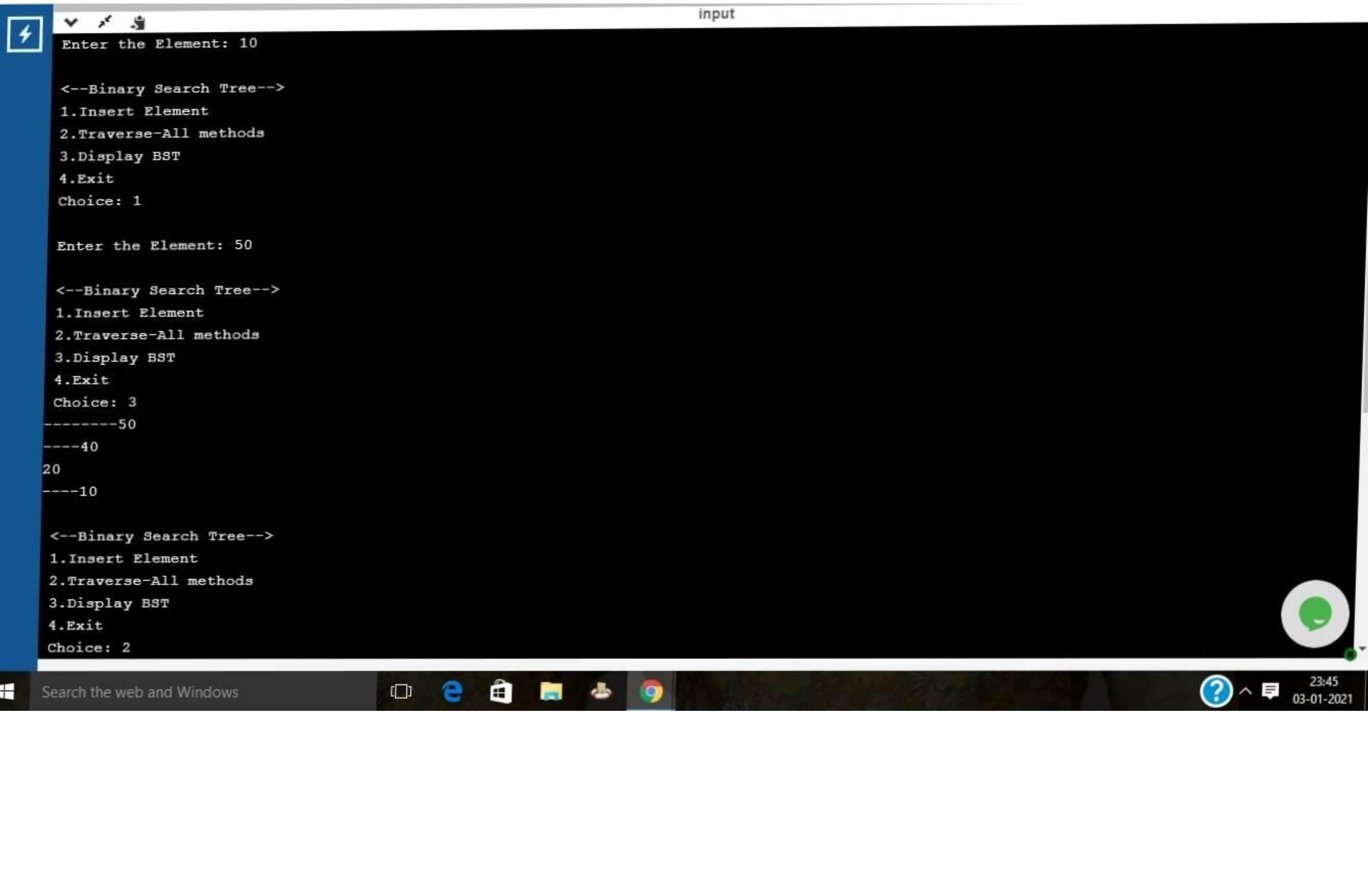
Anshul H. Supana

NAME: ANSHUL H. SURANA
USN: 1BM19CS020

```
printf("\n Post Order Traverse: ");
 PostOrder (root);
}
void PreOrder (Node * Root)
{   if (Root != NULL)
    { printf (" %d ", Root → data);
      preOrder (Root → left);
      preOrder ( Root → right);
    }
}
void inOrder (Node * Root)
{ if (Root != NULL)
    {   postorder (Root → left );
        postorder ( Root → right);
        printf (" %d ", Root → data);
        inOrder (Root → right);
    }
}
void postOrder (Node * Root)
{   if (Root != NULL)
    { postOrder (Root → left);
      postOrder (Root → right);
      printf (" %d ", Root → data);
    }
}
void display (Node * root, int i)
{   int j;
    if (root != NULL)
    { display (root → right , i+1);
      for (j=0; j<i ; j++)
      printf (" %d \n", root → data);
      display (root → left, i+1);
} }
```


Anshul H. Surana

```
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 20

Root Node Created
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 40

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 10
```



input

```
4.Exit
Choice: 2

Pre-Order Traverse:  20  10  40  50
In-Order Traverse:  10  20  40  50
Post-Order Traverse:  10  50  40  20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 80

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 2

Pre-Order Traverse:  20  10  40  50  80
In-Order Traverse:  10  20  40  50  80
Post-Order Traverse:  10  80  50  40  20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
```

input

```
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 80

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 2

Pre-Order Traverse:  20  10  40  50  80
In-Order Traverse:  10  20  40  50  80
Post-Order Traverse:  10  80  50  40  20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 4


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