DATA STRUCTURES LAB RECORD

USN: 1BM19CS074

NAME: Kizhakel Sharat Prasad

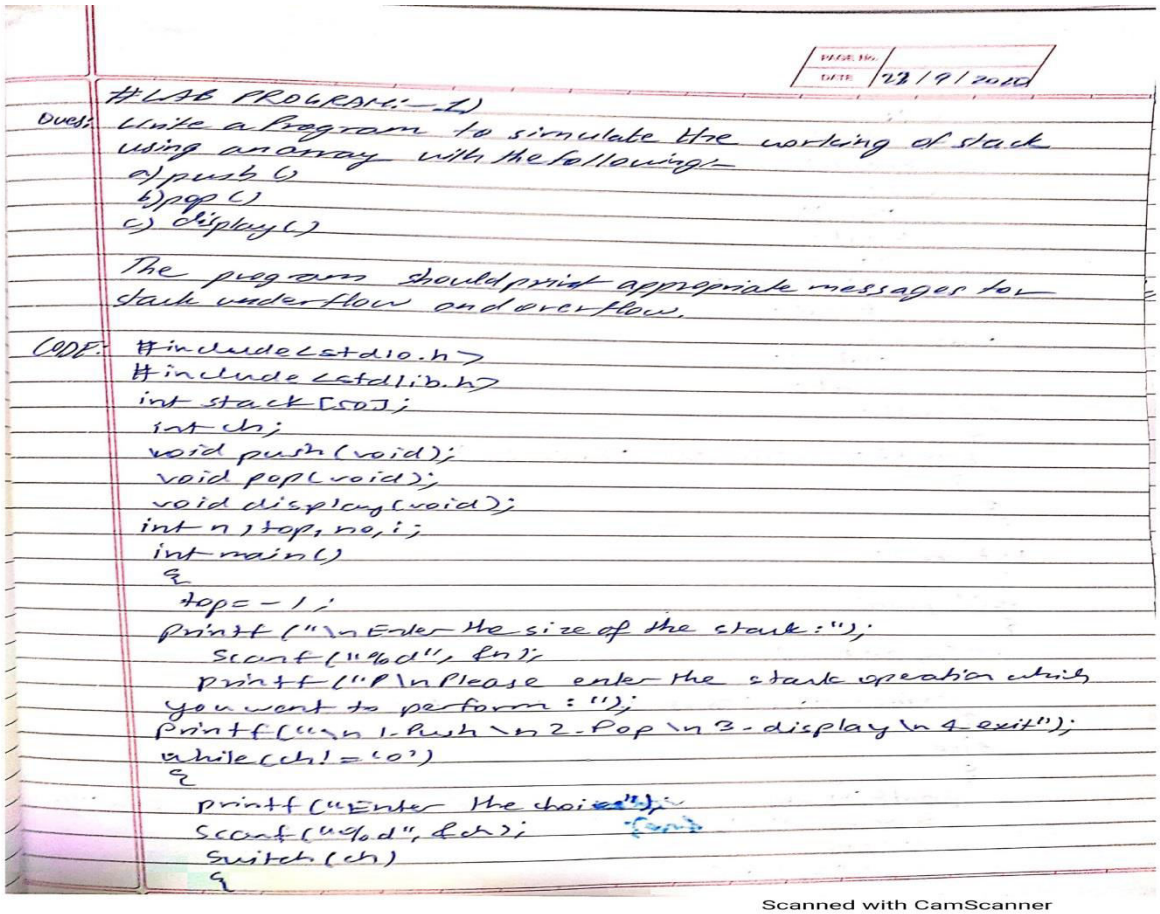
LAB PROGRAMS 1-10:

1.

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

a) Push
b) Pop
c) Display

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



```
case 1:
    push();
    break;
case 2:
    pop();
    break;
case 3:
    display();
    break;
case 4:
    exit(0);
    break;
default:
{
printf("\n Invalid choice!");
}
}
}
return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n STACK OVERFLOW");
}
else
{
printf("Enter a value to be pushed or inserted:");
scanf("%d", &no);
top++;
stack[top]=no;
```

```
}
}
void pop()
{
 if (top <= -1)
   {
   printf("\n UNDERFLOW");
  }
  else
   {
   printf("\n The popped element is %d", stack[top]);
   top--;
  }
}
void display()
 {
 if(top >= 0)
   {
   printf("\n The elements in stack are as follows:\n");
  for(i=top; i>=0; i--)
    {
    printf("\n%d\t", stack[i]);
   }
   printf("\n Press next choice");
    else
    {
   printf("\n The stack is empty");
 }
}
```

OUTPUT:

1. **PUSH**

```
Please enter the stack operation which you want to perform:
1.Push
2.Pop
3.display
4.exit
Enter the Choice:1
Enter a value to be inserted/pushed:20

Enter the Choice:1
Enter a value to be inserted/pushed:30

Enter the Choice:1
Enter a value to be inserted/pushed:40

Enter the Choice:1
Enter a value to be inserted/pushed:50
```

2. **POP**

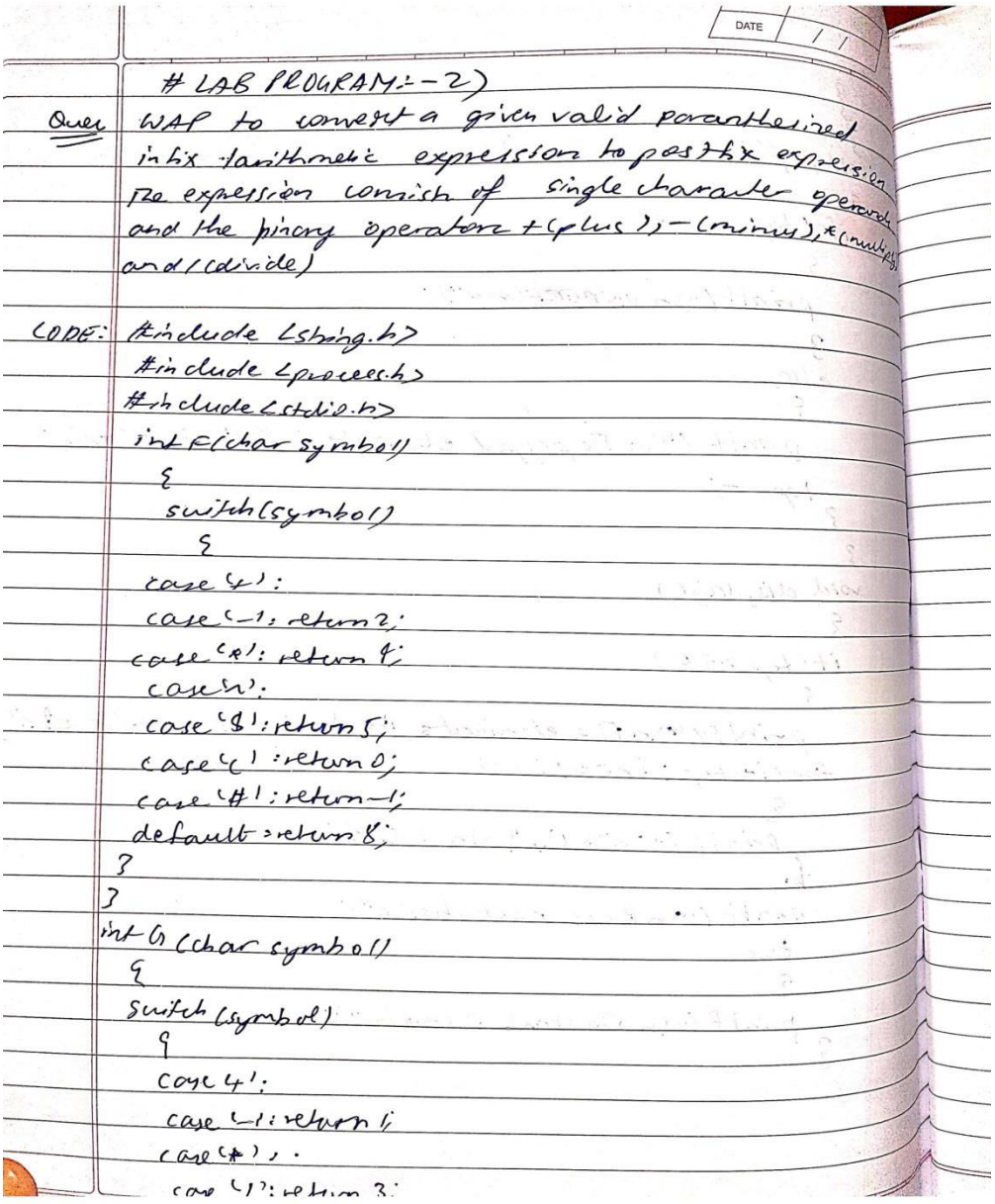
```
Enter the Choice:2

The popped element is 50
```

3. **DISPLAY**

```
 The elements in stack are as follows:

40,
30,
20,
```

2.

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



DATE / /

LAB PROGRAM:-2)

Ques 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)

CODE:

```
#include <string.h>
#include <process.h>
#include <stdio.h>
int F(char symbol)
  {
   switch(symbol)
     {
   case '+':
   case '-': return 2;
   case '*': return 4;
   case '/':
   case '%': return 5;
   case '(' : return 0;
   case '#' : return -1;
   default : return 8;
  }
}
int G(char symbol)
  {
  switch(symbol)
    {
    case '+':
    case '-': return 1;
    case '*':
    case '/': return 3;
```

```
case 1:
case '$': return 6;
case '(': return 9;
case ')': return 0;
default: return 7;
}
}
void infix_postfix (char infix[], char postfix[])
{
int top, i, j;
char s[30];
top = -1;
s[++top] = '#';
j = 0;
for (i = 0; i < strlen(infix); i++)
{
symbol = infix[i];
while (F(s[top]) > G(symbol))
{
postfix[j] = s[top--];
j++;
}
if (F(s[top]) != G(symbol))
s[++top] = symbol;
else
top--;
}
while (s[top] != '#')
{
postfix[j++] = s[top--];
}
postfix[j] = '\0';
```

```
void main()
{
char infix[20];
char postfix[20];
clrscr();
printf("Enter the valid infix expression \n");
scanf("%s", infix);
infix_postfix(infix, postfix);
printf("postfix exp. is ");
printf("%s\n", postfix);
getch();
}
```

OUTPUT:

```
Please enter the valid infix expression
a+b*(c^d-e)^(f+g*h)-i
the postfix is
abcd^e-fgh*+^*+i-
Process returned 0 (0x0)   execution time : 2.890 s
Press any key to continue.
```

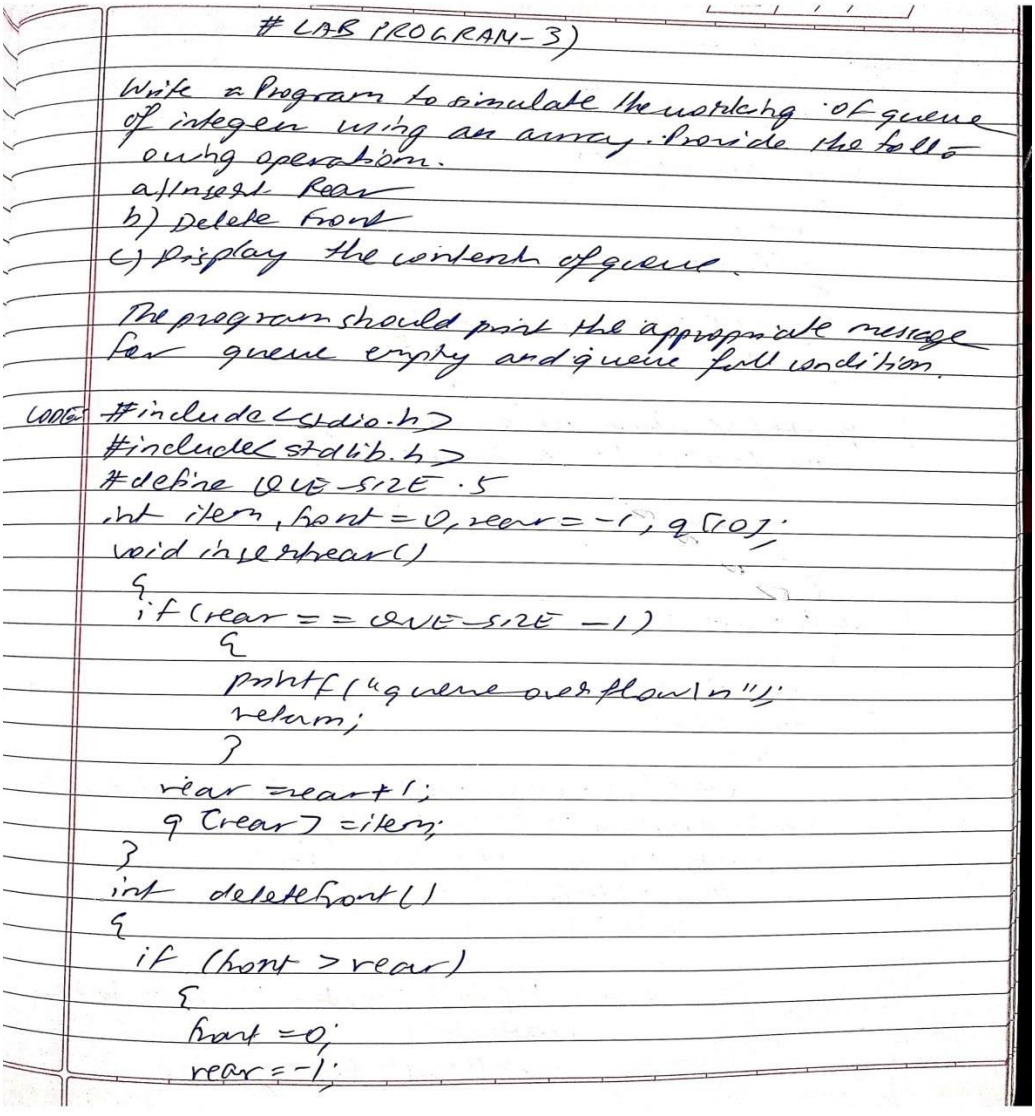
```
Please enter the valid infix expression
X^Y^Z-M+N+P/Q
the postfix is
XYZ^^M-N+PQ/+
Process returned 0 (0x0)   execution time : 1.739 s
Press any key to continue.
```

3.

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

a) Insert
b) Delete
c) Display

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



LAB PROGRAM-3)

Write a Program to simulate the working of queue of integer using an array. Provide the following operations.
a) Insert Rear
b) Delete Front
c) Display the contents of queue.

The program should print the appropriate message for queue empty and queue full condition.

```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 5
int item, front=0, rear=-1, q[10];
void insertrear()
{
 if(rear==QUE_SIZE-1)
    {
     printf("queue overflow\n");
     return;
    }
  rear=rear+1;
  q[rear]=item;
}
int deletefront()
{
 if (front>rear)
   {
    front=0;
    rear=-1;
```

```
return -1;
}
return q[front++];
}
void display()
{
int i;
if (front>=rear)
{
printf("queue is empty\n");
return;
}
printf("Content of queue\n");
for(i=front; i<=rear; i++)
{
printf("%d\n",q[i]);
}
}
void main()
{
int choice;
for(;;)
{
printf("1:insert rear\n2:delete front\n3:display\n4:exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item to be inserted\n");
scanf("%d",&item);
```

```
insertrear();
break;
case 2: item=deletefront();
        if(item==-1)
        {
        printf("queue is empty\n");
        }
        else
        {
        printf("item deleted=%d\n",item);
        }
    break;
case 3: displayQ();
        break;
default: exit(0);
}
}
}
```

OUTPUT:

1. INSERTION

```
Enter
1.for insertion
2.for deletion
3.for display
4.exit
1
Enter the item to be inserted
2
Enter
1.for insertion
2.for deletion
3.for display
4.exit
1
Enter the item to be inserted
3
Enter
1.for insertion
2.for deletion
3.for display
4.exit
1
Enter the item to be inserted
4
--------------------
Queue OVERFLOW!!
--------------------
```

2. DELETION

```
Enter
1.for insertion
2.for deletion
3.for display
4.exit
2
Item deleted = 1
Enter
1.for insertion
2.for deletion
3.for display
4.exit
3
Contents of Queue
2
3
```

3. DISPLAY

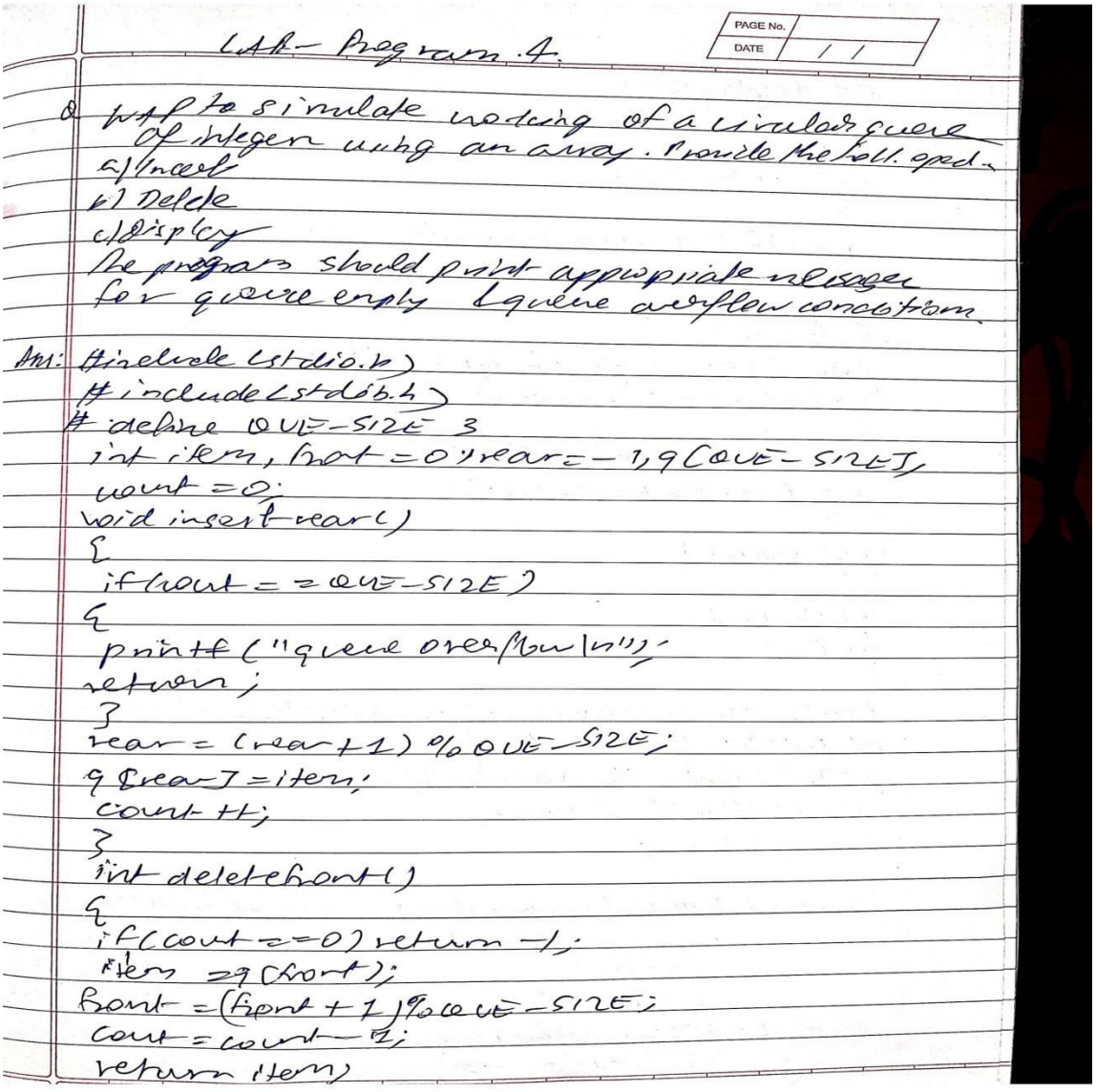
```
1.for insertion
2.for deletion
3.for display
4.exit
3
Contents of Queue
1
2
3
```

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 – Program 4.

PAGE No.
DATE / /

Q wap to simulate working of a circular queue of integer using an array. Provide the foll. oper.
a) Insert
b) Delete
c) Display
The program should print appropriate messages for queue empty & queue overflow conditions.

Ans:
```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 3
int item, front=0, rear=-1, q[QUE_SIZE];
count=0;
void insert_rear()
{
if(count==QUE_SIZE)
{
printf("queue overflow\n");
return;
}
rear=(rear+1)%QUE_SIZE;
q[rear]=item;
count++;
}
int delete_front()
{
if(count==0) return -1;
item=q[front];
front=(front+1)%QUE_SIZE;
count=count-1;
return item;
```

Scanned with CamScanner

```
void display()
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("content of queue\n");
for(int i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}}
void main()
{
int choice;
for(;;)
{
printf("\n1:insertrear\n2:deletefront\n3:display
\n4 exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item to be inserted\n");
        scanf("%d",&item);
        insertrear();
        break;
case 2: item=deletefront();
        if(item==-1
```

```
printf("queue is empty\n");
else
printf("\nitem deleted = %d\n", item);
    break;
case 3: display();
        break;
case 4: exit(0);
break;
default: printf("\nInvalid choice\n");
}
}
```

1.INSERT REAR

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
1

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
2

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
3
```

2.DELETE FRONT

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
4
queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
5
queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =1
```

3.DISPLAY

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
3

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
2
3
3
```

5.

WAP to Implement Singly Linked List with following operations

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

Labprogram-5

WAP to implement singly linked list with operations.
a) create a linked list
b) insertion at first position, any position and at end of list.
c) Display content of linked list.

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

PAGE No. / DATE / /

```
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->link=temp
       cur=cur->link;
    cur->link=temp;
  return first;
}
```

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

```
prev->link = temp;
temp->link = cur;
return first;
}
printf("\nIP\n");
return first;
}
void display(NODE first)
{
NODE temp;
if(first == NULL)
printf("cannot display item");
else
printf("\nContent of list:\n");
for(temp = first; temp != NULL; temp = temp->link)
{
printf("%d\n", temp->info);
}
}
void main()
{ int item, choice, pos;
NODE first = NULL;
for(;;)
{
printf("\n1.insert_front\n2.insert_rear
\n3.insert_pos\n4.Display_list\n5.Exit\n");
printf("\nEnter choice");
switch(choice
scanf("%d", &choice);
switch(choice)
```

DATE / /

```
{
case 1:
    printf("Enter the item at 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 position and item:\n");
    scanf("%d", &pos);
    scanf("%d", &item);
    first = insert_pos(item, pos, first);
    break;
case 4:
    display(first);
    break;
case 5:
    exit(0);
    break;
    default: printf("Invalid choice\n");
  }
 }
}
```

OUTPUT:

1.INSERT FRONT

```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
1

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
2

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
1
Enter the item at front-end
3
```

2.INSERT REAR

```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
2
Enter the item at rear-end
5

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
2
Enter the item at rear-end
6
```

3.INSERT POS

```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
3
Enter the position and item:
2
9

1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
4
Contents of the list:
4
9
3
2
1
5
6
```

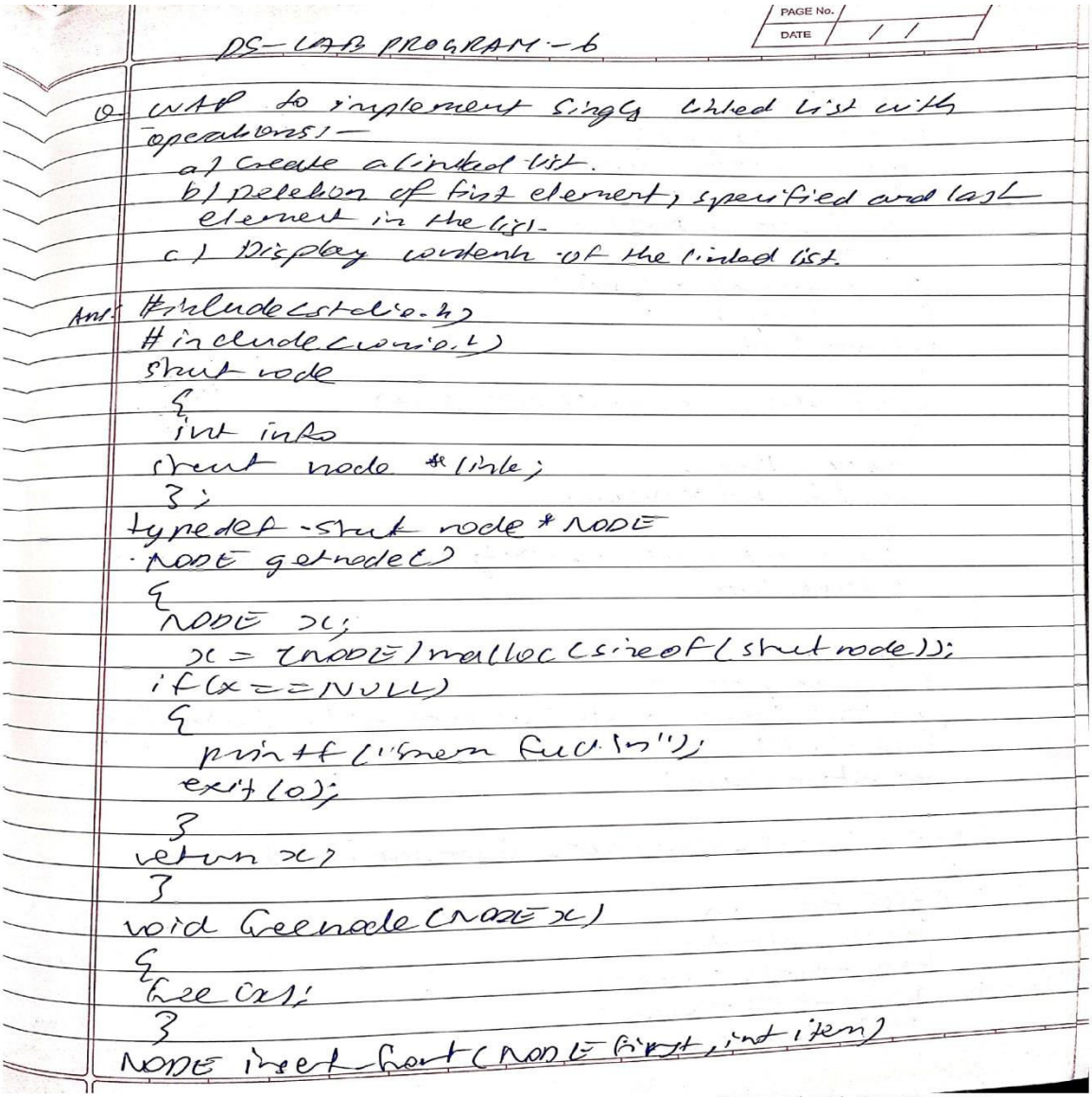
4.DISPLAY

```
1:Insert_front
2:Insert_rear
3:Insert_pos
4:Display_list
5:Exit
Enter the choice
4
Contents of the list:
4
3
2
1
5
6
```

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.



DS-LAB PROGRAM-6

Q WAP to implement Singly Linked List with operations:-
a) Create a linked list.
b) Deletion of first element, specified and last element in the list.
c) Display contents of the linked list.

Ans:
```
#include<stdio.h>
#include<conio.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("mem 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 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 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 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=first;cur->link;
   } delete
 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 prev, cur;
int count;
if (first==NULL||pos<=0)
{
printf("invalid position\n");
return NULL;
}
if(pos==1)
{
cur=first;
first=first->link;
printf("item deleted is %d", cur->info);
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
if(count==pos)
{
break;
}
prev=cur;
cur=cur->link;
count++;
}
```

PAGE No.
DATE / /

```
if(count!=pos)
{
 printf("invalid position\n");
 return first;
}
prev->link = cur->link;
printf("item deleted was %d", cur->info);
freenode(cur);
return first;
}
void display(NODE first)
 {
  NODE temp;
  if(first==NULL)
    {
     printf("list empty cannot display items");
    }
  else
    {
    printf("contents of the list:\n");
    }
for(temp=first; temp!=NULL; temp=temp->link)

    {
    printf("%d\n", temp->info);
    }
  }
void main()
{ int item, choice, pos;
  NODE first=NULL;
  for(;;)
  {
```

```
printf("\n1: Insert_front \2. Delete
Front-\n 3: Insert_rear 4: Delete_rear \n
Delete_pos \n 6. Display_list 7: Exit\n");
printf("Enter choice");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter item at front end");
        scanf("%d", &item);
        first = insert_front(item, first);
        break;
case 2:
        first = delete_front(first);
        break;
case 3: printf("Enter item at rear end");
        scanf("%d", &item);
        first = insert_rear(first, item);
        break;
case 4: first = delete_rear(first);
        break;
case 5:
        printf("Enter position:\n");
        scanf("%d", &pos);
        first = delete_pos(pos, first);
        break;
case 6: display(first);
        break;
case 7: exit(0);
        break;
default: printf("Invalid choice\n");
}
}
```

OUTPUT:

1. DELETE FRONT AND DELETE REAR

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete_pos
 6:Display_list
 7:Exit
Enter the choice
6
Contents of the list:
4
3
2
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete_pos
 6:Display_list
 7:Exit
Enter the choice
2
Item deleted at front-end is=4
```

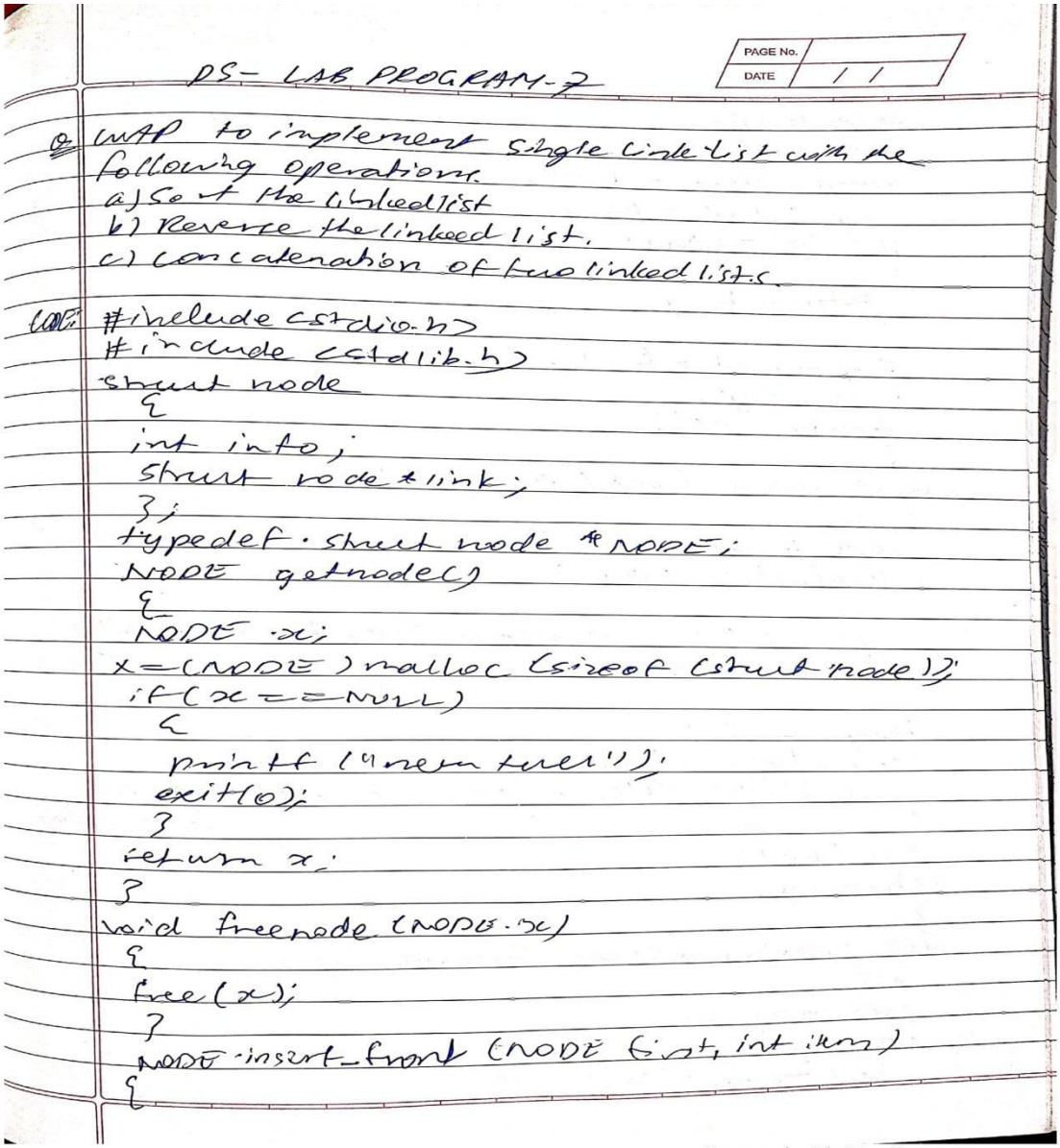
2. DELETE AT POS

```
Contents of the list:
3
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete_pos
 6:Display_list
 7:Exit
Enter the choice
5
Enter the position:
2
Item deleted is 2
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Delete_pos
 6:Display_list
 7:Exit
Enter the choice
6
Contents of the list:
3
```

7.

WAP Implement Single Link List with following operations

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



DS- LAB PROGRAM-7

PAGE No.
DATE / /

Q) WAP to implement single linked list with the following operations.
a) Sort the LinkedList
b) Reverse the linked list.
c) Concatenation of two 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("mem full");
   exit(0);
  }
 return x;
}
void freenode(NODE x)
 {
 free(x);
 }
NODE insert_front(NODE first, int item)
{
```

DATE / /

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

DATE / /

```
temp->link = NULL;
if(first == NULL)
  {
   return temp;
  }
cur = first;
while (cur->link != NULL)
  cur = cur->link;
cur->link = temp;
return first;
}
NODE deleterear(NODE first)
{
NODE cur, prev;
if (first == NULL)
  {
  printf("List is empty cannot delete");
return first;
  }
if (first->link == NULL)
{
printf("Item deleted %d\n", first->info);
free (first);
return NULL;
 }
prev = NULL;
cur = first;
while (cur->link != NULL)
  {
  prev = cur
cur = cur->link;
 }
```

```
printf("Item deleted is at rear end is %d", cur->info);
free(cur);
prev->link = NULL;
return first;
}
NODE order_list(int item, NODE first)
{
NODE temp, prev, cur;
temp = getnode();
temp->info = item;
temp->link = NULL;
if(first == NULL)
    return temp;
if(item < first->info)
  {
  temp->link = first;
  return temp;
  }
prev = NULL;
cur = first;
while(cur != NULL && item > cur->info)
{
prev = cur;
cur = cur->link;
}
prev->link = temp;
temp->link = cur;
return first;
}
```

```
void display (NODE first)
{
 NODE temp;
  if (first == NULL)
    printf("\n List empty cannot display items\n");
  else
  printf("\n Content of the list: \n");
  for (temp = first ; temp != NULL ; temp = temp->link)
   {
    printf("\n %d\n", temp->info);
  }
}
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 main()
{
int item, choice, key, n, i;
NODE first = NULL, a, b;
for(;;)
{
printf("\n1: Insert_front \n2: Delete_front \n3
Insert_rear \n4: Delete_rear\n");
printf("5: Order_list \n6: Display_list \n7)
Concat \n8: Reverse \n9: Exit \n");
printf("Enter your choice");
scanf("%d", &choice);
switch(choice)
{
case 1: printf("Enter item at front end");
        scanf("%d", &item);
        first = insert_front(first, item);
        break;
case 2: first = delete_front(first);
        break;
case 3: printf("Enter the item at rear end\n");
        scanf("%d", &item);
        first = insert_rear(first item);
        break;
```

PAGE No.
DATE / /

```
case 4: first = delete_rear(first);
        break;
case 5: printf("Enter item to be inserted in ordered
                list");
        scanf("%od", &item);
        first = order_list(item, first);
        break;
case 6: display(first);
        break;
case 7: printf("Enter the no of nodes in 1st");
        scanf("%od", &item);
        a = insert_rear(a, item);
        a = NULL;
        for(int i=0; i<n; i++)
        {
        printf("Enter the item");
        scanf("%od", &item);
        a = insert_rear(a, item);
        }
        printf("Enter the no. of nodes in 2nd");
        scanf("%od", &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);
display(a);
break;
```

PAGE No.
DATE / /

```
case 8: first = reverse (first);
        display (first);
        break;

case 9: exit(0);
        break;
        default: printf("Invalid choice \n");
    }
}
```

OUPUT:

1. SORT THE LINKED LIST

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
5
Enter the item to be inserted in ordered_list
1

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
5
Enter the item to be inserted in ordered_list
77
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
5
Enter the item to be inserted in ordered_list
30

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
5
Enter the item to be inserted in ordered_list
45
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
6
Contents of the list:
1
30
45
77
```

2.CONCAT THE LINKED LIST

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
7
Enter the no of nodes in 1
3
Enter the item
1
Enter the item
2
Enter the item
3
Enter the no of nodes in 2
4
Enter the item
5
Enter the item
6
Enter the item
7
Enter the item
8
```

```
Contents of the list:
1
2
3
5
6
7
8
```

3.REVERSE THE LINKED LIST

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice
8
Contents of the list:
77
45
30
1
```

8.

WAP to implement Stack & Queues using Linked Representation

DS LAB PROGRAM 8

PAGE No.
DATE / /

Q) WAP to implement Stack and Queues using Linked representation.

```
Ans: #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 delete_front(NODE first)
{
NODE temp;
if (first == NULL)
{
printf("List is empty cannot delete\n");
return first;
}
```

```
temp=first;
 temp=temp->link;
printf("\nItem 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 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->link~~)
 }
printf("
 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;
}
void display (NODE first)
{
NODE temp;
if (first == NULL)
    {
    printf("List empty");
    return;
    }
printf("Items");
for (temp=first; temp!=NULL; temp=temp->link)
    {
    printf("%d\n", temp->info);
    }
  }
void main()
{ int item, choice;
  int pos, i, n, count, key;
NODE first = NULL, a, b;
for(;;)
 {
 printf("\n1: Stack \n2. Queue \n3: Exit\n");
 printf("Enter your choice");
 scanf("%d", &choice);
 switch(choice)
 {
case 1: printf("stack\n");
      for(;;)
       {
      printf("\n1: Insert rear \n2: Delete rear \n3: Display
        list \n4: Exit");
```

```
printf("Enter the choice \n");
scanf("%d", &choice);
switch (choice)
  {
  case 1: printf("Enter item");
    scanf("%d", &item);
    first = insert_rear (first, item);
    break;
case 2: first = delete_rear (first);
        break;
case 3: display (first);
        break;
default: exit(0);
        break;
  }
}
case 2: printf("Queue \n");
  for(;;)
  {
  printf("\n 1: Insert Rear \n 2: Delete_front
  \n 3: Display_list \n 4: Exit \n");
  printf("Enter the choice");
  scanf("%d", &choice);
  switch(choice)
   {
  case 1: printf("Enter item at rear end");
        scanf("%d", &item);
        first = insert_rear (first, item);
      break;
case 2: first = delete_front (first);
        break;
```

```
case 3: display(first);
        break;
default: exit(0);
        break;
 }
}
case 3: exit(0);
default: printf("\nInvalid choice\n");
}}}
```

OUTPUT:

1.STACK IMPLEMENTATION

```
1:Stack
2:Queue
3:Exit
Enter the choice
1
Stack

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
1

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
2
```

```
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
3

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
1
2
3

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 3
```

```
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
1
2
```

2.QUEUE IMPLEMENTATION

```
1:Stack
2:Queue
3:Exit
Enter the choice
2
QUEUE

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
1

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
4
```

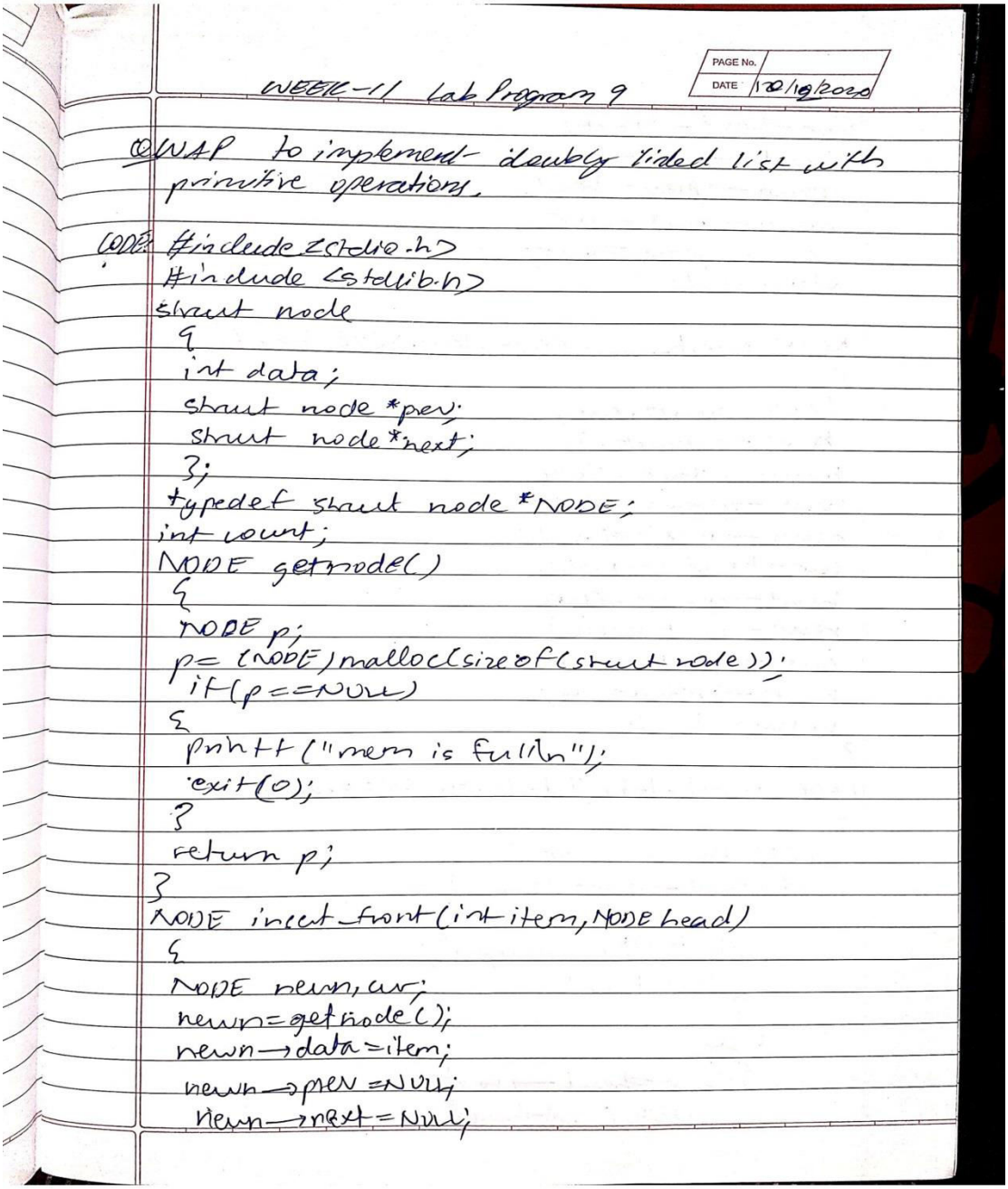
```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
6

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
7

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
1
4
6
7
```

```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=1

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=4

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
6
7
```

9.

WAP Implement doubly link list with primitive operations

a) Create a doubly linked list.
b) Insert a new node to the left of the node.
c) Delete the node based on a specific value
d) Display the contents of the list



WEEK-11 Lab Program 9

PAGE No.
DATE 10/10/2022

Q) WAP to implement doubly linked list with primitive operations.

```
Code: #include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *prev;
struct node *next;
};
typedef struct node *NODE;
int count;
NODE getnode()
{
NODE p;
p=(NODE)malloc(sizeof(struct node));
if(p==NULL)
{
printf("mem is full\n");
exit(0);
}
return p;
}
NODE insert_front(int item, NODE head)
{
NODE new, cur;
new=getnode();
new->data=item;
new->prev=NULL;
new->next=NULL;
```

```
cur = head -> next;
 head -> next = newn;
 newn -> prev = head;
 newn -> next = cur;
  cur -> prev = newn;
return head;
}
NODE insert_rear(int item, NODE head)
{
 NODE newn, cur;
 newn = getnode();
 newn -> data = item;
 newn -> prev = NULL;
 newn -> next = NULL;
 cur = head -> prev;
 head -> prev = newn;
 newn -> next = head;
 cur -> next = newn;
 newn -> prev = cur;
 return head;
}
NODE insert_left(int item, NODE head)
{
   NODE newn, cur, left;
   if(head -> next == head)
      {
       printf("DLL is empty");
       return head;
       }
      else
       { cur = head -> next;
          while(cur != head)
```

PAGE No.
DATE / /

```
{
if(cur->data == item)
  {
    break;
    }
    cur = cur -> next;
}
if (cur == head)
  {
   printf("key node for insertion not found");
   return head;
}
left = cur -> prev;
printf("Enter data towards left of %d", item);
newn = getnode();
scanf("%d", &newn->data);
newn -> prev = left;
left -> next = newn;
newn -> next = cur;
cur -> prev = newn;
return head;
}
}
NODE insert_right(int item, NODE head)
{
 NODE newn, cur, right;
 if(head -> next == head)
  {
   printf("DLL is empty");
   return head;
  }
 cur = head -> next;
```

```
while (cur!=head)
{
 if(cur->data==item)
    {
     break;
    }
    cur=cur->next;
}
if(cur==head)
{
 printf("key node for insertion not found");
  return head;
}
right=cur->next;
 rear = getnode();
 printf("Enter data towards right of %d",item);
 scanf("%d",&rear->data);
 right->prev=rear;
 rear->next=right;
 rear->prev=cur;
 cur->next=rear;
 return head;
}
NODE del_front(NODE head)
{
 NODE cur, next;
 if(head->next==head)
  {
   printf("DLL is empty");
   return head;
  }
```

```
cur=head->next;
next=cur->next;
head->next=next;
next->prev=head;
printf("The node deleted is :%d",cur->data);
free(cur);
return head;
}
NODE del_rear(NODE head)
{
NODE cur,left;
if(head->next==head)
{
printf("DLL is empty");
}
cur=head->prev;
left=cur->prev;
left->next=head;
head->prev=left;
printf("The node is deleted %d",cur->data);
free(cur);
return head;
}
NODE dll_search(int item, NODE head)
{
count=0;
NODE cur;
cur=head->next;
while(cur!=head)
{
count++;
if(cur->data==item)
```

```
{
printf("Item found at %d", count);
return head;
}
cur=cur->next;
}
if (count==0)
{
printf("Element not found");
}
return head;
}
NODE del_all(int item, NODE head)
{
NODE left, cur, right;
if(head->next==head)
{
printf("DLL empty");
return head;
}
count=0;
cur=head->next;
while(cur!=head)
{
if(item!=cur->data)
{
cur=cur->next;
}
else
{
count++;
left=cur->prev;
```

```
right = cur -> next;
left -> next = right;
right -> prev = left;
free(cur);
cur = right;
}
}
if (count == 0)
{
printf("key not found");
}
else
{
printf("key found at %d position and are
deleted", count);
}
return head;
}
NODE remove_duplicates(int item, NODE head)
{
NODE cur, left, right;
count = 0;
if (head -> next == head)
{
printf("DLL is empty");
}
cur = head -> next;
while (cur != head)
{
if (cur -> data == item)
{
count++;
```

```
if(count>1)
  {
  left=cur->prev;
  right=cur->next;
  left->next=right;
  right->prev=left;
  free(cur);
  cur=right;
  }
  else
  {
  cur=cur->next;
  }
}
else if(cur->data!=item)
  {
  cur=cur->next;
  }
}
if(count==1)
{
printf("given key has no duplicates");
}
else
 {
 printf("Duplicates have been removed
 only unique present");
 }
return head;
}
void display(NODE head)
{
```

```
NODE cur;
  if(head->next==head)
  {
  printf("Doubly linked list is empty");
  return;
  }
 cur=head->next;
 while(cur!=head)
  {
  printf("%d", cur->data);
  cur=cur->next;
  }
 printf("\n");
 }
int main()
 {
 NODE head;
 head=getnode();
 head->next=head;
 head->prev=head;
 NODE p;
 int ch, item;
 int i, cval;
 for(;;)
  {
printf("\n1. Insert front \n2. Insert rear \n3.
 ..Insert left of the given node \n4. insert
 right of given node \n5. delete front \n6.
 delete rear \n7. delete all occurrences of key
 8: display \n9. searching for a given element
 \n10. removing duplicates");
printf("\nPlease enter your choice");
```

```
scanf("%d", &ch);
switch(ch)
{
case 1:
    printf("Enter the item:");
    scanf("%d", &item);
    head = insert_front(item, head);
    break;

case 2: printf("Enter the key item");
    scanf("%d", &item);
    head = insert_rear(item, head);
    break;
case 3: printf("Enter the key item\n");
    scanf("%d", &item);
    head = insert_left(item, head);
    break;
case 4: c=0;
    printf("Enter the key item\n");
    scanf("%d", &item);
    head = insert_right(item, head);
    break;
case 5: del_front(head);
    break;
case 6: del_rear(head);
    break;
case 7: printf("Enter the key to be deleted");
    scanf("%d", &item);
    head = del_all(item, head);
    break;
case 8: display(head);
    break;
```

```
case 9: printf("Enter the value to be searched");
        scanf("%d", &item);
        head = dll_search(item, head);
        break;
case 10: printf("Enter the key whose duplicate need
        to be deleted!");
        scanf("%d", &item);
        head = remove_duplicates(item, head);
        break;
default: exit(0);
}
}
return 0;
}
```

OUTPUT:

Function wise:

1.INSERT FRONT

```
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:3

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:5
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice
1
Enter the item:1

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice
1
Enter the item:22
```

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:5
```

2.INSERT REAR

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice2
Enter the key item1

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22 1 1
```

3.INSERT LEFT

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice 3
Enter the key item
5
Enter data towards left of 5=4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 22 1
```

4.INSERT RIGHT

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice4
Enter the key item
3
Enter data towards right of 37

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22 1
```

5.DELETE FRONT

```
Please enter your choice8
1 22 7 3 5 5 4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice5
The node deleted is 1
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5 4
```

6.DELETE REAR

```
Please enter your choice8
22 7 3 5 5 4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice6
The node deleted is 4
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5
```

7.DELETE ALL OCCURENCES

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice7
Enter the key to be deleted1
key found at 2 positions and are deleted
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22
```

8.DISPLAY

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5
```

9.SEARCH FOR KEY

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice9
Enter the value to be searched7
Item found at 5
```

10.REMOVE DUPLICATES

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice10
Enter the key whose duplicates need to be deleted5
Duplicates have been removed..only unique present
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 3 7 22
```

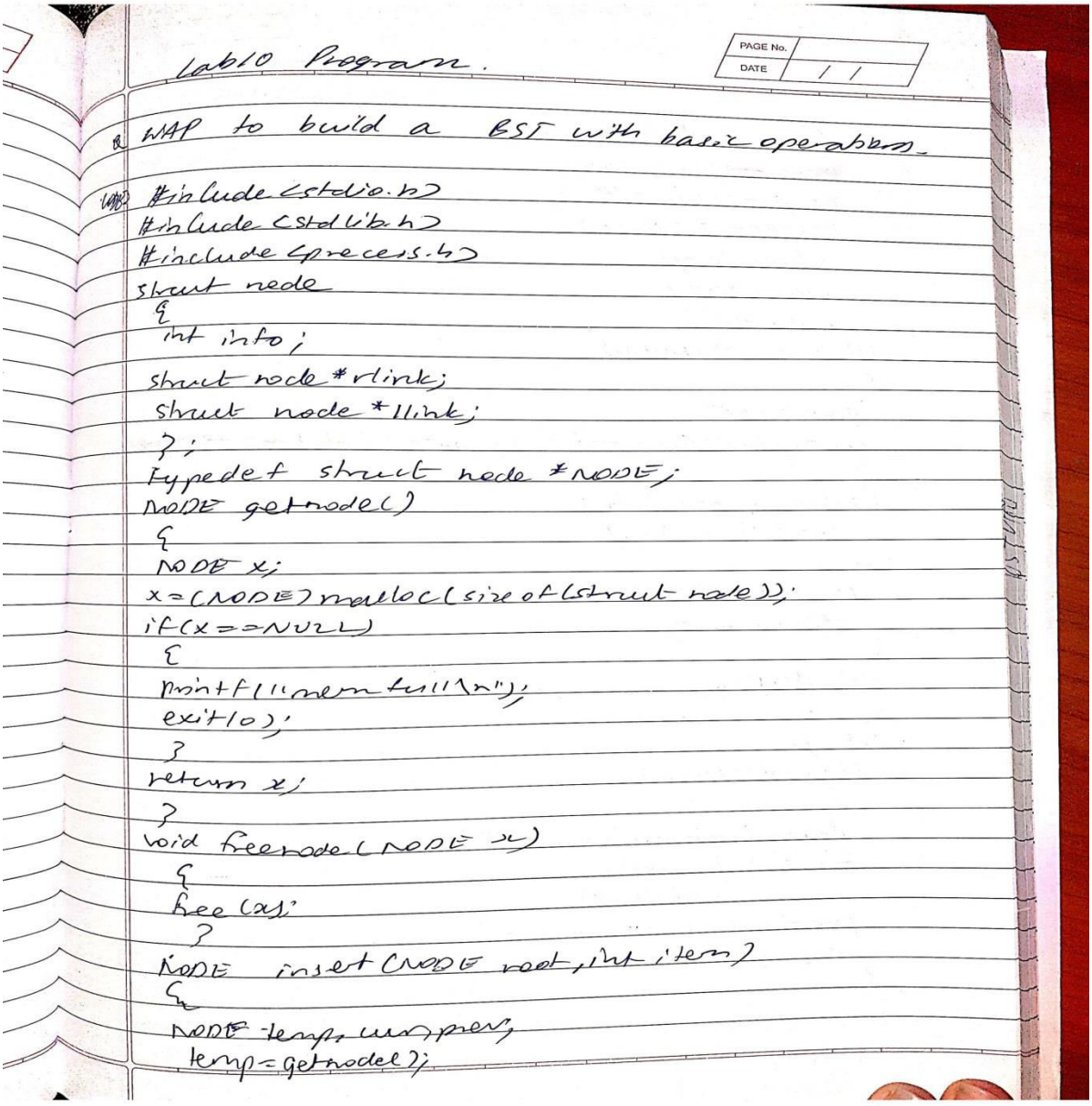
10.

Write a program

a) To construct a binary Search tree.

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

c) To display the elements in the tree.



Lab10 Program.

PAGE No.
DATE / /

WAP to build a BST with basic operations.

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

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

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

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

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

OUTPUT:

1. BUILD A BST:

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

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

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
8
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
67

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
45

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    67
      45
  23
10
  8
```

2. PREORDER TRAVERSAL :

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
10
8
23
67
45
```

3. POSTORDER TRAVERSAL :

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
8
45
67
23
10
```

4. INORDER TRAVERSAL :

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
8
10
23
45
67
```

5.DELETE:

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
  67
    45
23
  8
```