REPORT DATA STRUCTURE LAB PROGRAMS:--

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

* Program:-

```
#include<stdio.h>

#include<stdlib.h>

#define STACK_SIZE 3

int top=-1;

int s[10];

int item;

void push()

{

if(top==STACK_SIZE-1)

{

printf("stack overflow\n");

return;

}

top=top+1;

s[top]=item;

}

int pop()
```

```
{
if(top==-1) return -1;
return s[top--];
}
void display()
{
int i;
if(top==-1)
{
printf("stack is empty\n");
return;
}
printf("contents of the stack\n");
for(i=top; i>=0; i--)
{
printf("%d\n",s[i]);
}
}
int main()
{
int item_deleted;
int choice;

for(;;)
{
```

```
printf("\n 1:push\n 2:pop\n 3:display\n 4: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);

      push();

      break;

case 2:item_deleted=pop();

        if(item_deleted==-1)

      printf("stack is empty\n");

      else

      printf("item deleted is %d\n",item_deleted);

      break;

case 3:display();

      break;

default:exit(0);

}

}

}
```

screenshots:--

Data Structure 3C | DS_pgms-1 - Google Docs | Repl.it - DeafeningPointedKernelm

repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c

PremaPrema / DeafeningPointedKer... Run ▶ Upgrade Share

Files

main.c

```c
#include<stdio.h>
#include<stdlib.h>
#define STACK_SIZE 3
int top=-1;
int s[10];
int item;
void push()
{
if(top==STACK_SIZE-1)
{
printf("stack overflow\n");
return;
}
top=top+1;
s[top]=item;
}
int pop()
{
if(top==-1) return -1;
return s[top--];
}
void display()
```

```
 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
10

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
20

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
```

Type here to search ENG 11:53 08-12-2020

Data Structure 3C | DS_pgms-1 - Google Docs | Repl.it - DeafeningPointedKernelm

repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c

PremaPrema / DeafeningPointedKer... Run ▶ Upgrade Share

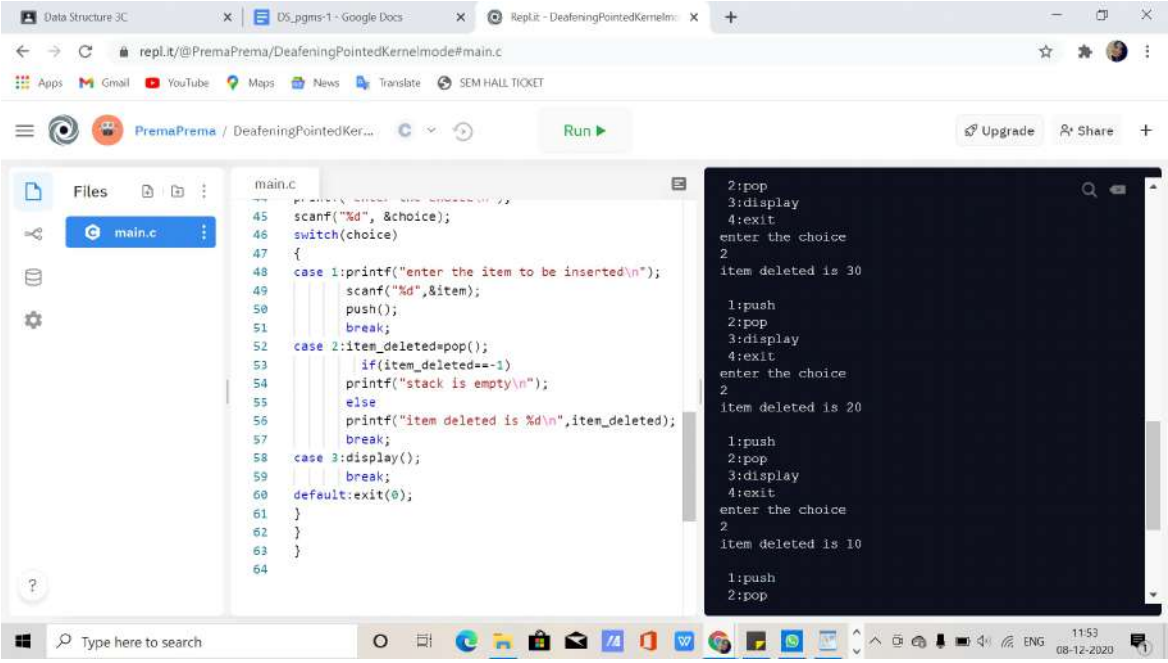
Files

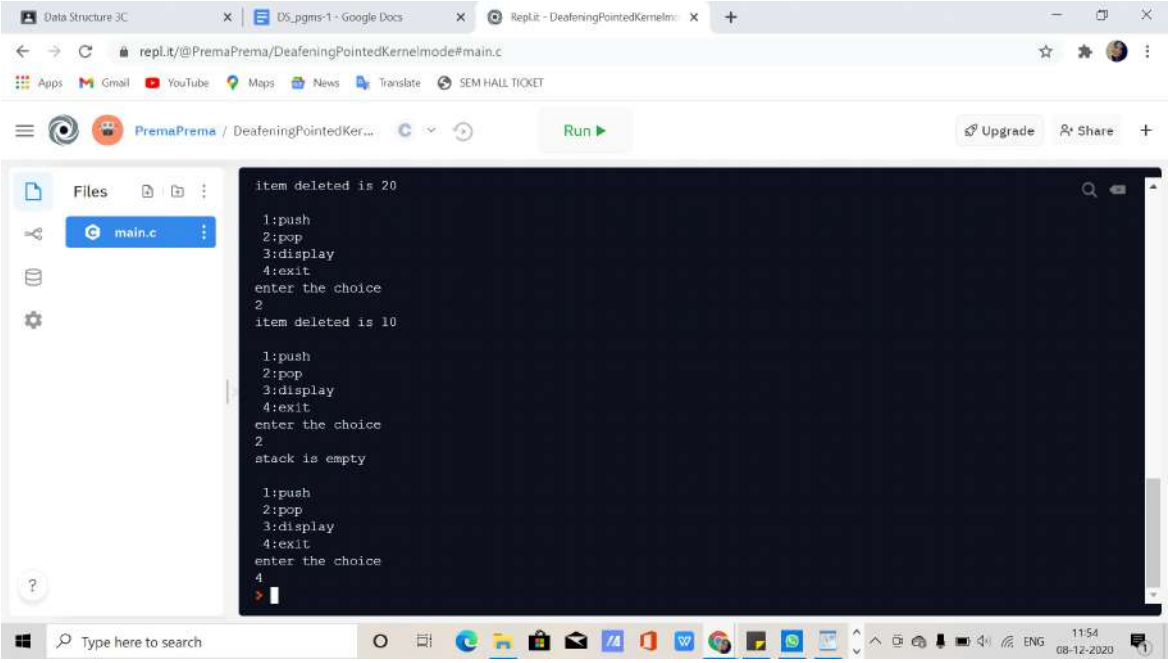
main.c

```c
{
int i;
if(top==-1)
{
printf("stack is empty\n");
return;
}
printf("contents of the stack\n");
for(i=top; i>=0; i--)
{
printf("%d\n",s[i]);
}
}
int main()
{
int item_deleted;
int choice;

for(;;)
{
printf("\n 1:push\n 2:pop\n 3:display\n 4:exit\n");
printf("enter the choice\n");
```

```
enter the item to be inserted
30

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
40
stack overflow

 1:push
 2:pop
 3:display
 4:exit
enter the choice
3
contents of the stack
30
20
10

 1:push
```

Type here to search ENG 11:53 08-12-2020





Written program:-

PAGE NO.
DATE: / /

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int top= -1;
int stack [SIZE];
void push(int ele)
{
if (top == SIZE -1)
{
printf ("The stack is \n")
}
else
{
  top++
  stack [top] = ele;
  }
}
int pop()
{
if (top==-1)
{
return 0;
}
else
{ printf (" Element removed is : %d \n", stack[top]
```

↳ overflow

```
return 1;
}
}
void display()
{
if (top==-1)
printf("The stack is empty\n");
else
{
printf("The elements are\n");
for (int i=0; i<=top; i++)
{
printf("%d \n", stack[i]);
}
}
}
int main()
{
int c,d,p;
while (c!=4)
{
printf("Enter command \t1- push \t2- pop
\t3- Display \t4- Exit \n");
scanf("%d", &c);
switch(c)
{
case 1: printf("Enter an element \n");
scanf("%d", &d);
push(d);
break;
case 2: p=pop();
if (p==0)
printf("Stack is ~~empty~~ underflow \n");
else
printf("\n Element removed \n");
break;
case 3: display();
break;
case 4: break;
default: printf("Invalid input \n");
}
}
return 0;
}
```

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)

Program:

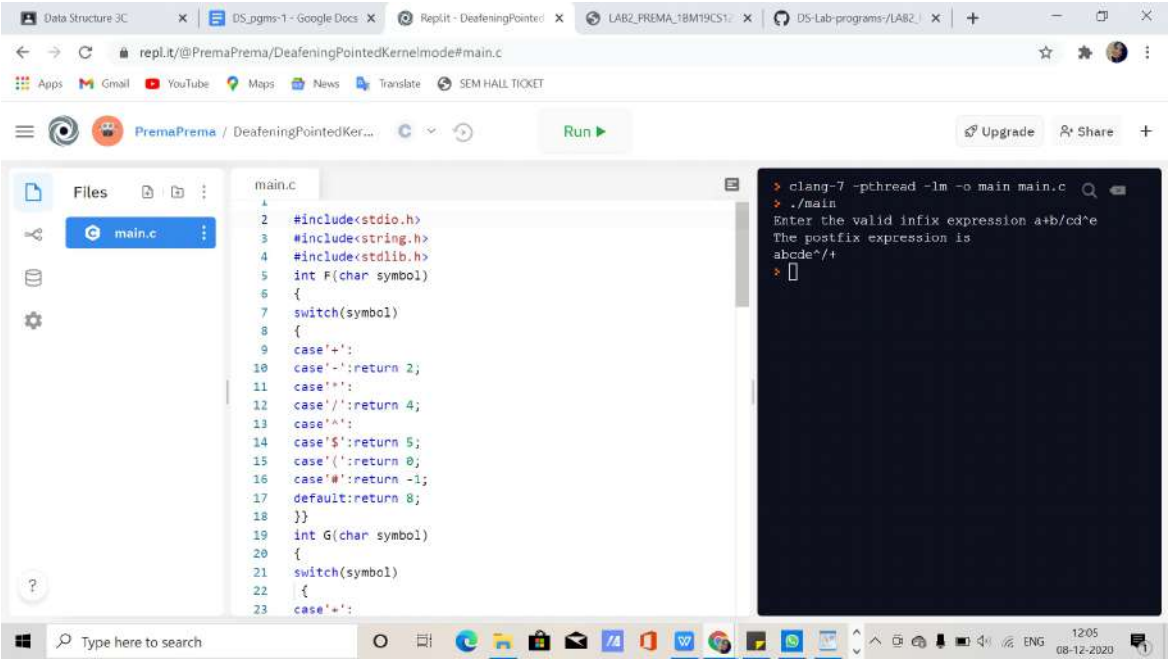
```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int F(char symbol)

{

switch(symbol)

{

case'+':
```

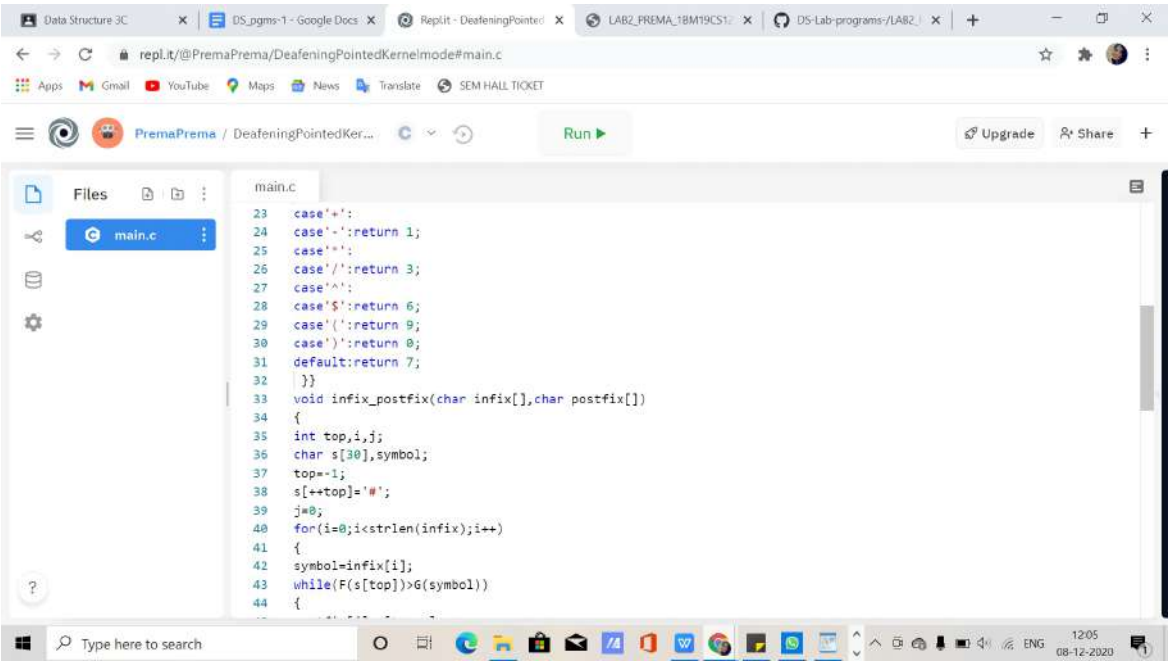
```
case'-':return 2;
case'*':
case'/':return 4;
case'^':
case'$':return 5;
case'(':return 0;
case'#':return -1;
default:return 8;
}}
int G(char symbol)
{
switch(symbol)
 {
case'+':
case'-':return 1;
case'*':
case'/':return 3;
case'^':
case'$':return 6;
case'(':return 9;
case')':return 0;
default:return 7;
 }}
void infix_postfix(char infix[],char postfix[])
{
int top,i,j;
char s[30],symbol;
```

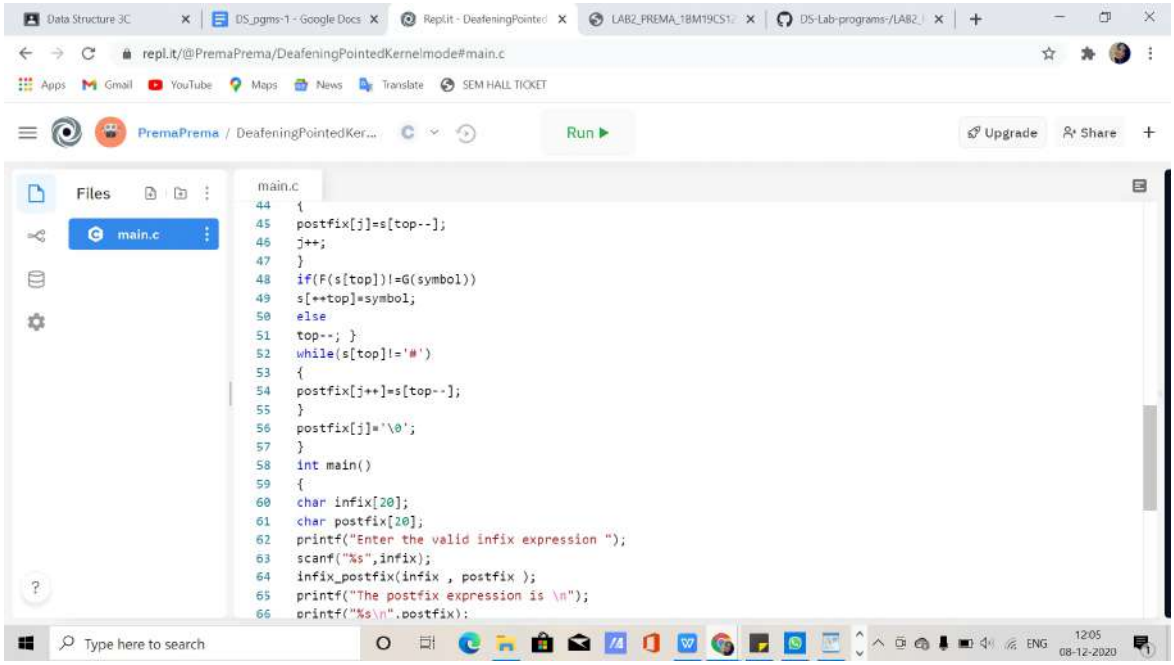
```
top=-1;
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
{
postfix[j]=s[top--];
j++;
}
if(F(s[top])!=G(symbol))
s[++top]=symbol;
else
top--; }
while(s[top]!='#')
{
postfix[j++]=s[top--];
}
postfix[j]='\0';
}
int main()
{
char infix[20];
char postfix[20];
printf("Enter the valid infix expression ");
scanf("%s",infix);
```

```
infix_postfix(infix , postfix );

printf("The postfix expression is \n");

printf("%s\n",postfix);

}
```

screenshots:

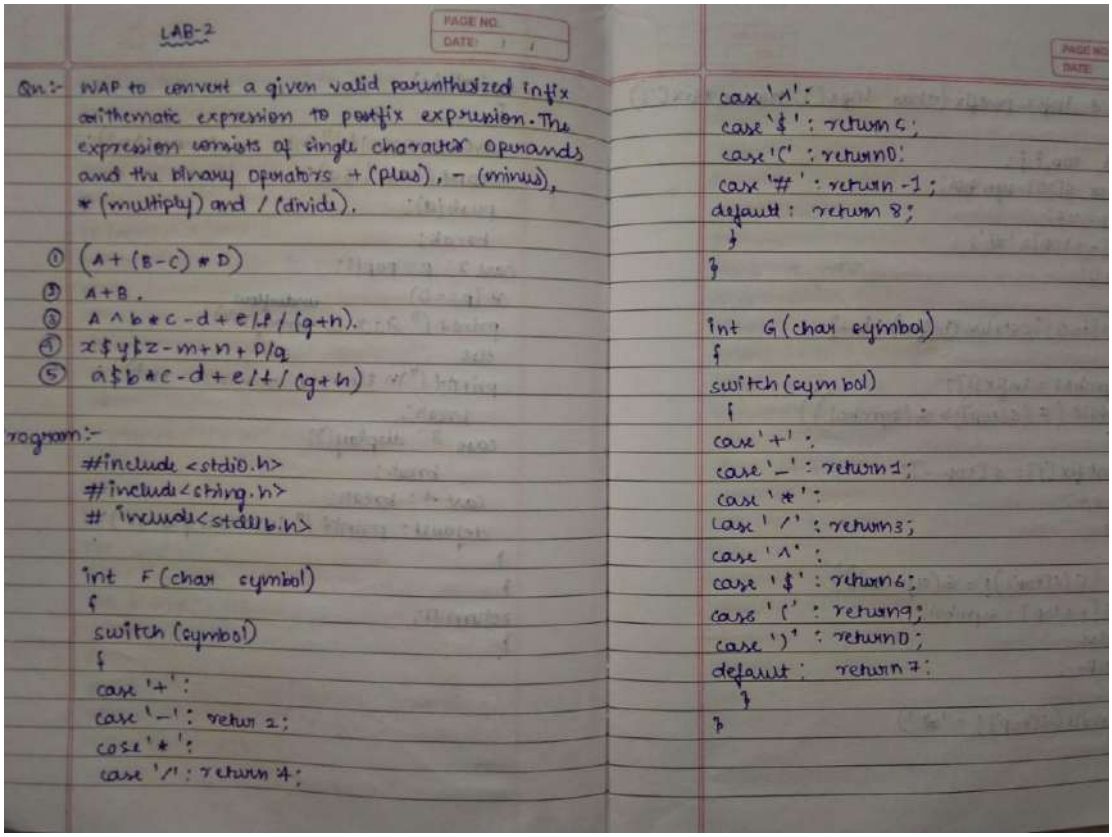






```c
{
postfix[j]=s[top--];
j++;
}
if(F(s[top])!=G(symbol))
s[++top]=symbol;
else
top--; }
while(s[top]!='#')
{
postfix[j++]=s[top--];
}
postfix[j]='\0';
}
int main()
{
char infix[20];
char postfix[20];
printf("Enter the valid infix expression ");
scanf("%s",infix);
infix_postfix(infix , postfix );
printf("The postfix expression is \n");
printf("%s\n",postfix);
```

written program:--



LAB-2

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

① (A+(B-C)*D)
② A+B.
③ A^b*c-d+e/f/(g+h).
④ xyz-m+n+p/q
⑤ a$b*c-d+e/f/(g+h)

rogram:-

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

int F(char symbol)
{
switch (symbol)
{
case '+':
case '-': retur 2;
cose '*':
case '/': return 4;
case '^':
case '$': return 6;
case '(': return 0;
case '#': return -1;
default: return 8;
}
}

int G(char symbol)
{
switch (symbol)
{
case '+':
case '-': return 1;
case '*':
case '/': return 3;
case '^':
case '$': return 6;
case '(': return 9;
case ')': return 0;
default: return 7;
}
}
```

```
void infix_postfix (char infix[], char postfix[])
{
int top, i, j;
char s[30], symbol;
top = -1;
s[++top] = '#';
j=0;

for (i=0; i<strlen (infix); i++)
{
symbol = infix[i];
while (F(s[top]) > G(symbol))
{
postfix[j] = s[top--];
j++;
}

if (F(s[top]) != G(symbol))
s[++top] = symbol;
else
top--;
}
while (s[top] != '#')
{
postfix[j++] = s[top--];
}
postfix[j] = '\0';
}
void main()
{
char infix[20];
char postfix[20];
printf ("Enter the valid infix expression");
scanf ("%s", infix);
infix_postfix (infix, postfix);
printf ("The postfix expression is \n");
printf ("%s \n", postfix);
}
```

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

Program:-

```
#include<stdio.h>

#include<stdlib.h>

#define QUE_SIZE 3
```

```
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) return -1;
return q[front++];
}
void displayQ()
{
int i;
if(front>rear)
{
printf("queue is empty\n");
return;
}
printf("Contents of queue \n");
for(i=front;i<=rear;i++)
{
```

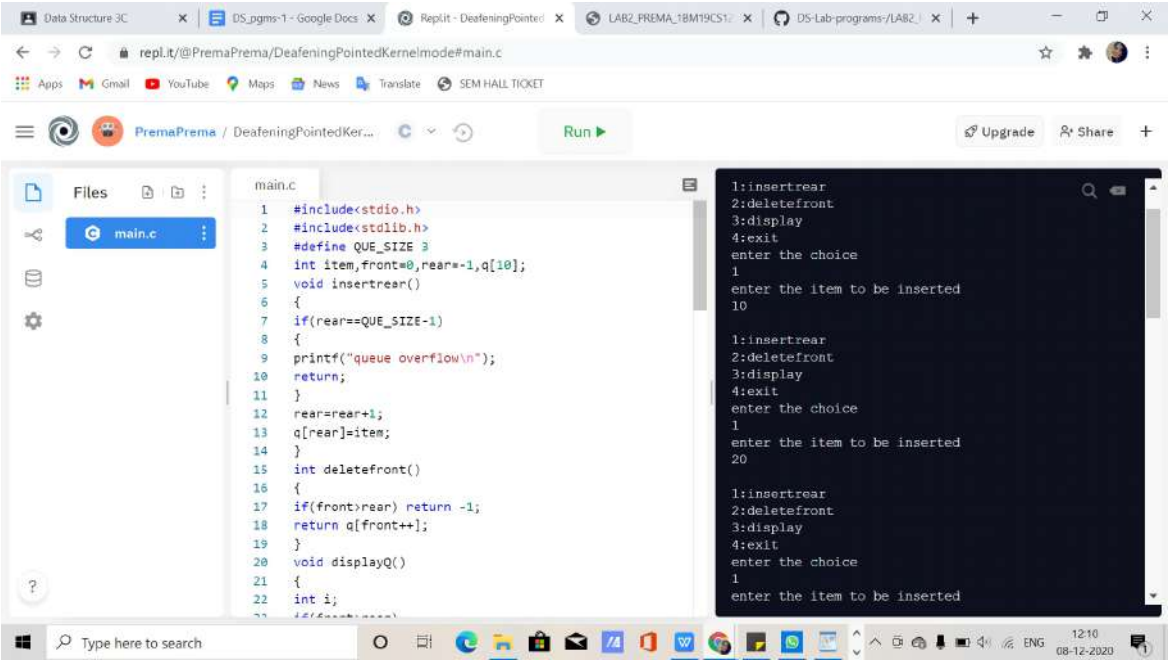
```
printf("%d\n",q[i]);
}
}
int main()
{
 int choice;

 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("item deleted =%d\n",item);
        break;
 case 3:displayQ();
        break;
 default:exit(0);
```

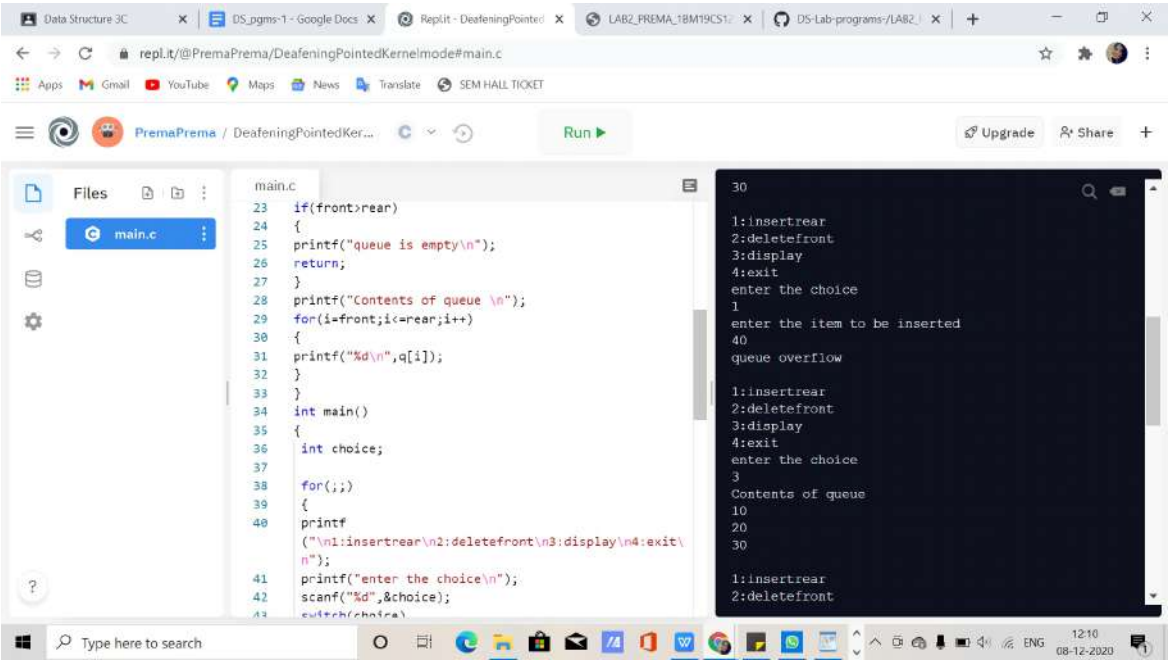
}

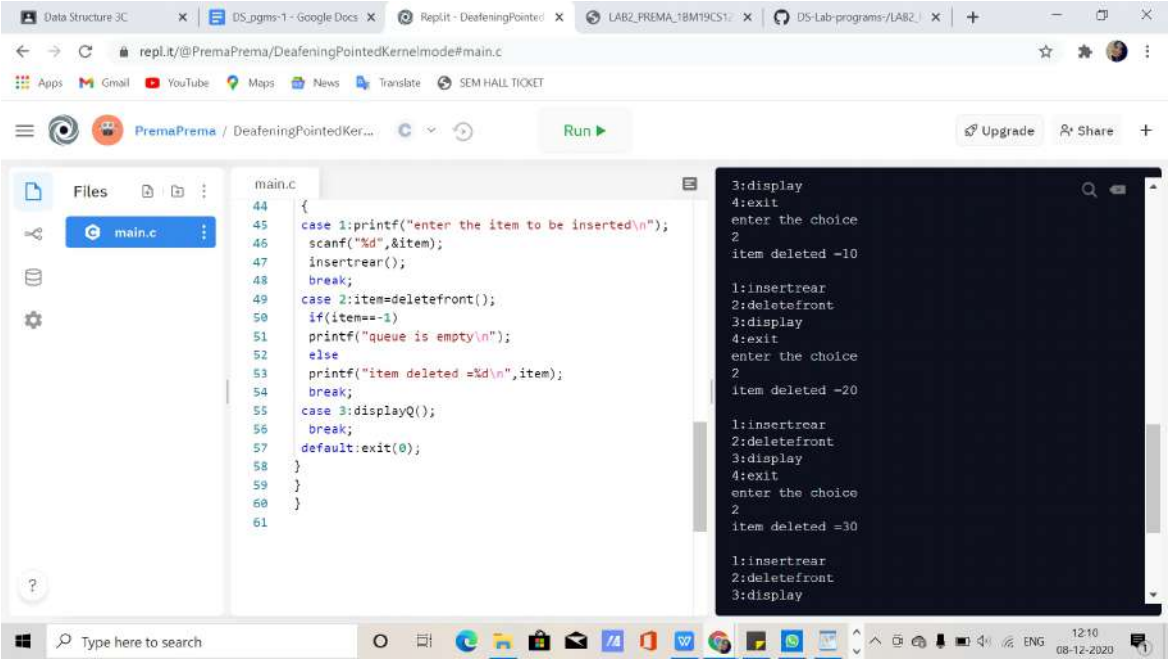
}

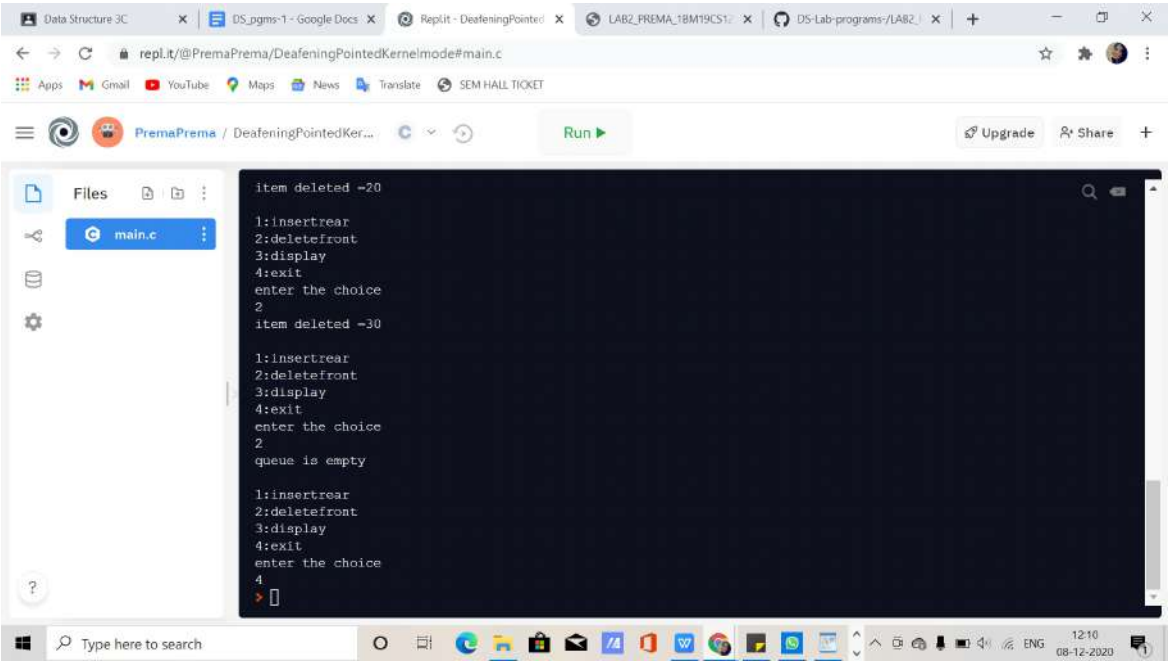
}

screenshots:






repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c
PremaPrema / DeafeningPointedKer...
Run
Upgrade
Share
Files
main.c
{
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);
}
}
}
3:display
4:exit
enter the choice
2
item deleted =10
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =20
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =30
1:insertrear
2:deletefront
3:display
Type here to search


repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c
PremaPrema / DeafeningPointedKer...
Run
Upgrade
Share
Files
main.c
item deleted =20
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =30
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
queue is empty
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
4
Type here to search

written :

Lab Program 3 :-

```
#include <stdio.h>
# include <stdlib.h>
# define QUE_SIZE 3
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 Q()
{
int i;
if (front > rear)
{
printf(" queue is empty \n");
 return;
}
printf (" contents of queue \n");
for (i = front ; i <= rear; i++)
{
printf (" %d \n", q[i]);
}
}
int main()
{
int choice;
 for(;;)
{
printf (" 1 :Insertrear 2 : deletefront 3 : display 4 :
printf (" enter the choice \n");
scanf (" %d", &choice);
```

PAGE NO.
DATE: / /

```
switch(choice)
{
case1: 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;
case3: displayQ();
 break;
default: exit(0);
 }
}
```

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

program:-

```
#include<stdio.h>

#include<stdlib.h>

#define QUE_SIZE 3

int item,front=0,rear=-1,q[QUE_SIZE],count=0;

void insertrear()

{

if(count==QUE_SIZE)

{

printf("queue overflow\n");

return;

}

rear=(rear+1)%QUE_SIZE;

q[rear]=item;

count++;

}

int deletefront()

{

if(count==0) return -1;

item=q[front];

front=(front+1)%QUE_SIZE;

count=count-1;

return item;

}

void displayQ()
```

```
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
int 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("item deleted =%d\n",item);
        break;
 case 3:displayQ();
        break;
 default:exit(0);
}
}
}
```

screenshot:-

repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c

PremaPrema / DeafeningPointedKer...

main.c

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

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
10

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
20

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

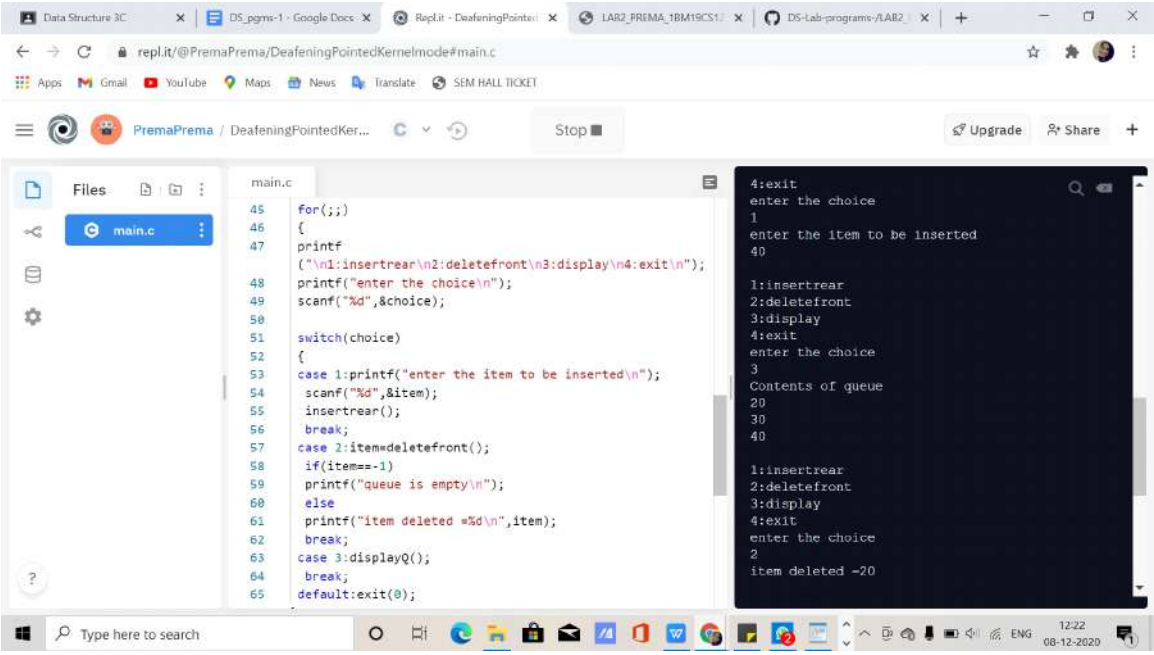
repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c

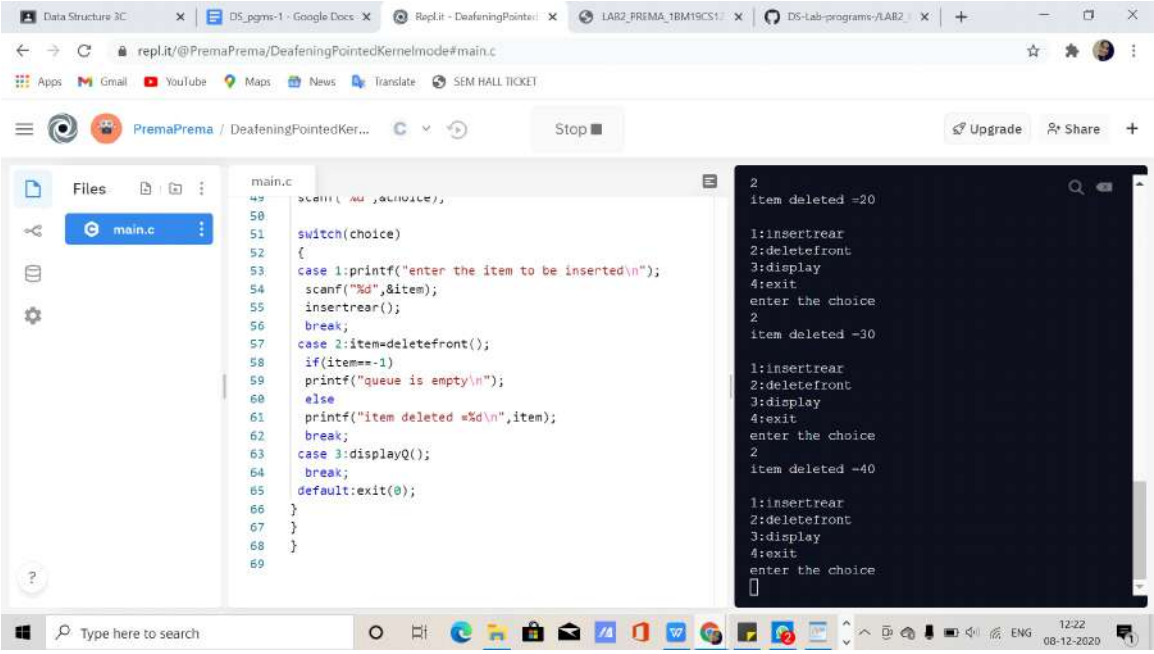
PremaPrema / DeafeningPointedKer...

main.c

```c
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
int main()
{
 int choice;
```

```
30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
10
20
30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =10

1:insertrear
2:deletefront
3:display
4:exit
```


repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c
PremaPrema / DeafeningPointedKer...
Stop
Upgrade
Share
Files
main.c
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("item deleted =%d\n",item);
break;
case 3:displayQ();
break;
default:exit(0);
4:exit
enter the choice
1
enter the item to be inserted
40
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
20
30
40
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =20
Type here to search


repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c
PremaPrema / DeafeningPointedKer...
Stop
Upgrade
Share
Files
main.c
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);
}
}
}
2
item deleted =20
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =30
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =40
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
Type here to search

written:--

Lab program 4:

PAGE NO.
DATE: / /

```
#include <stdio.h>
#include <stdlib.h>
# define quesize 3
int item, front=0, rear=-1, q[que_size], count=0;
void insertrear()
{
 if (count==que_size)
 {
printf("queue overflow");
 return;
}
rear=(rear+1)%que_size;
q[rear]=item;
count++;
}
int deletefront()
{
 if (count==0) return -1;
 item=q[front];
front=(front+1)%que_size;
 count=count-1;
 return item;
 }
```

```
void displayq()
{
int i,f;
if (count==0)
{
printf("queue is empty");
return;
}
f=front;
printf(" contents of queue \n");
for (i=0;i<=count;i++)
{
printf("%d \n",q[f]);
f=(f+1)%que_size;
}
}

void main()
{
int choice;
for (;;)
{
printf("\n1.Insert rear \n2. Delete front \n3 Display
\n4.exit \n");
printf("Enter the choice:");
scanf("%d",&choice);
switch (choice)
{
case 1: printf(" Enter the item to be inserted:
scanf("%d",&item);
insertrear();
break;
case 2: item=deletefront();
if (item==-4)
printf("queue is empty \n");
else
printf(" item deleted is %d \n",item);
break;
case 3: displayq();
break;
default: exit(0);
}
}
```

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.

program:-

```
#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\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 IF(NODE second,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
return temp;
temp->link=second;
second=temp;
return second;
}

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 IR(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 insert_pos(int item,int pos,NODE first)

{

NODE temp;

NODE prev,cur;

int count;

temp=getnode();
```

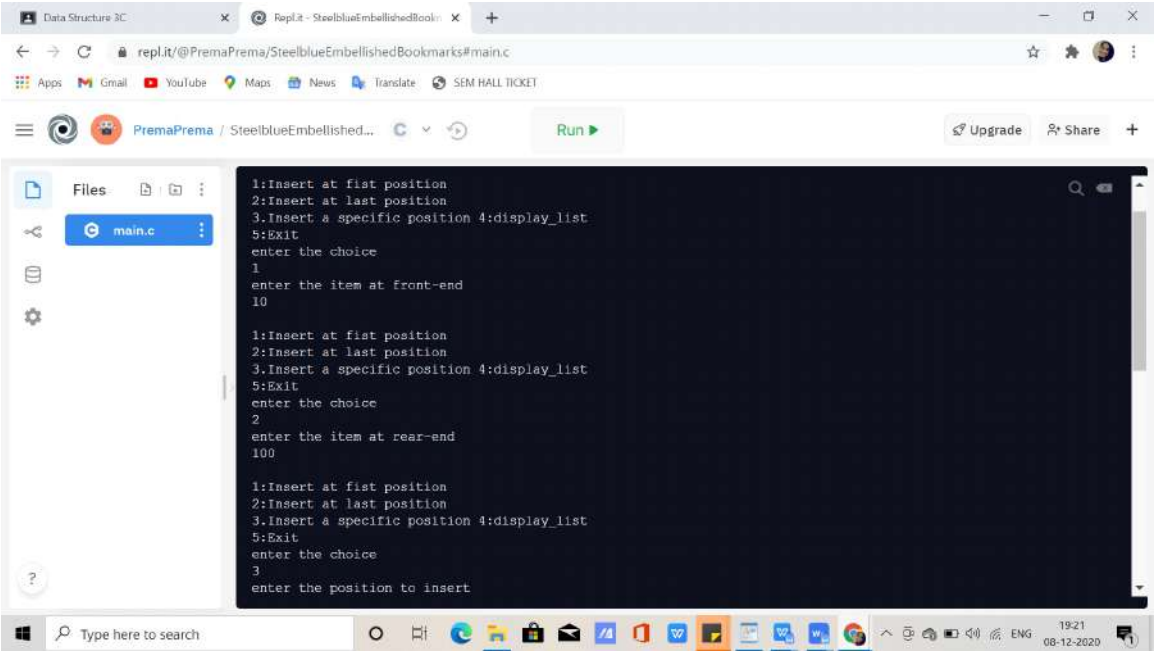
```
temp->info=item;

temp->link=NULL;

if(first==NULL && pos==1)

return temp;

if(first==NULL)

{

 printf("invalid pos\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("Invalid Position \n");

return first;

}





void display(NODE first)

{

 NODE temp;

 if(first==NULL)

 printf("list empty cannot display items\n");

 for(temp=first;temp!=NULL;temp=temp->link)

  {

  printf("%d\n",temp->info);

  }

}

int main()

{

int item,choice,pos,element;

NODE first=NULL;

NODE second=NULL;
```

```
for(;;)
{
printf("\n1:Insert at fist position\n2:Insert at last position\n3.Insert a specific position
4:display_list\n5:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  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 to insert \n");
                scanf("%d",&pos);
                printf("enter the item to insert \n");
                scanf("%d",&item);
                first=insert_pos(item,pos,first);
                break;
  case 4:display(first);
```

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

screenshot:--



```
2
enter the item to insert
20

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
3
enter the position to insert
3
enter the item to insert
30

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
4
10
20
30
100
```

```
enter the choice
3
enter the position to insert
3
enter the item to insert
30

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
4
10
20
30
100

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
5
```

written:--

Lab-6

```
#include <stdio.h>
#include <stdlib.h>
struct node{
 int info;
 struct node *link;
 };
typedef structnode *NODE;
NODE getnode()
 {
 NODE x;
x = (NODE) malloc (sizeof(structnode));
 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 IF (NODE second, int item)
 {
NODE temp;
temp = getnode();
temp -> info = item;
 temp -> link = NULL
if (second == NULL)
return temp;
temp -> link = second;
 second = temp;
 return second;
 }
NODE insert_pos (int item, int pos, NODE fi
  {
  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)
 {
 pf ("invalid pos\n");
 return first;
 }
if (pos == 1)
 {
temp -> link = first;
return temp;
 }
if (count == pos)
 {
prev -> link = temp;
temp -> link = cur;
return first;
}
void display (NODE first)
 {
NODE temp;
if (first == NULL)
```

```
print ("list empty cannot display items\n")
for (temp = first, temp! = NULL; temp = temp -> li
 { pf ("%d \n", temp -> info);
 }
}
int main() {
int item, choice, pos, element;
NODE first = NULL;
 NODE second = NULL;
for(;;)
 { pf ("\n1 insert @ first pos \n2 insert @ last
 \n3 - insert @ specific pos \n4 display \n5 e
 pf ("enter choice");
 sf ("%d", &choice);
 switch (choice)
 {
case 1: pf ("enter the item @ front end \n");
 sf ("%d", &item);
first = insert_front (first);
 break;
case 2: print ("enter the item @ rear end")
  sf ("%d", &item);
 first = insert_rear (first, item);
  break;
```

```
case 3:
pf ("enter the position to insert");
sf ("%d", &pos);
pf ("enter the item to insert");
sf ("%d", &item);
first = insert_pos(item, pos, first);
break;
case 4: display(first);
 break;
default : exit(0);
    break;
}
}
```

6.WAP to Implement Singly Linked List with following operations

a) 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.

program:--

```
#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\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 IF(NODE second,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
return temp;
temp->link=second;
second=temp;
return second;
}

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 IR(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 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;
```

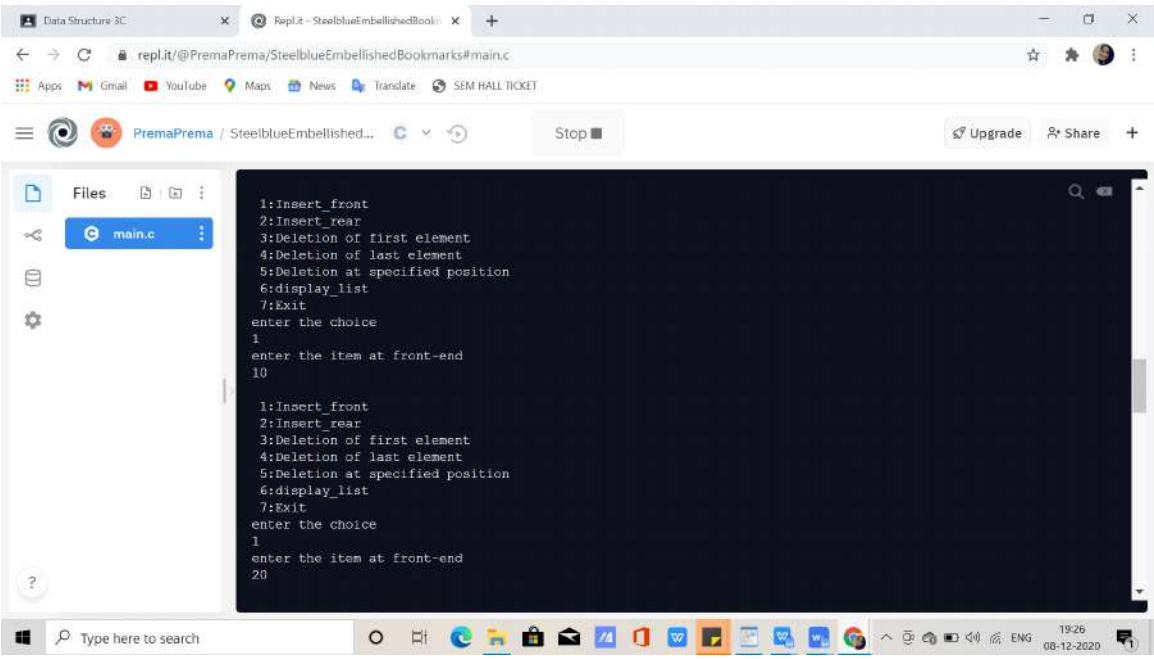
```
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}


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);

return first;

}

prev=NULL;

cur=first;

count=1;

while(cur!=NULL)

{

if(count==pos)

break;  //if found

prev=cur;

cur=cur->link;

count++;

}

if(count!=pos)
```

```
{
  printf("invalid position\n");
   return first;
}
if(count!=pos)
{
  printf("invalid position specified\n");
  return first;
}

prev->link=cur->link;
 freenode(cur);
return first;
}


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

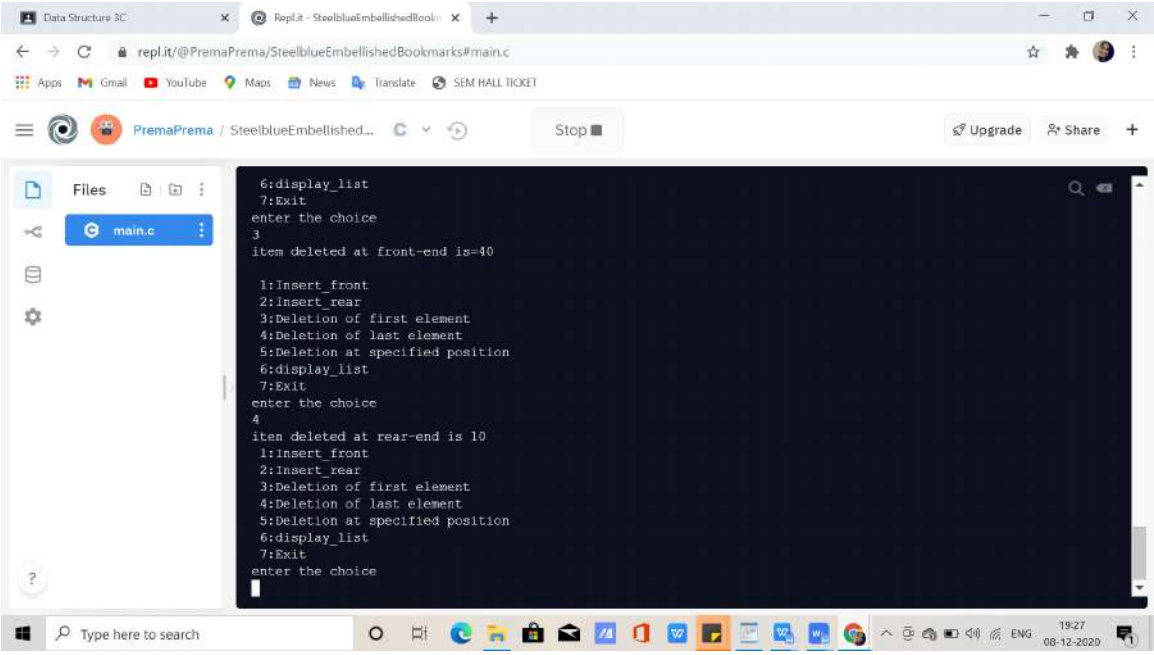
```
}

int main()

{

int item,choice,pos;

NODE first=NULL;

NODE second=NULL;


for(;;)

{

printf("\n 1:Insert_front\n 2:Insert_rear\n 3:Deletion of first element\n 4:Deletion of last element\n 5:Deletion at specified position\n 6:display_list\n 7:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

 {

  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:first=delete_front(first);

          break;
```

```
 case 4:first=delete_rear(first);
        break;
case 5:
  printf("enter the position to delete \n");
            scanf("%d",&pos);
            first=delete_pos(pos,first);
  break;


 case 6:display(first);
        break;
default:exit(0);
        break;
 }
}
}
```



```
 1:Insert_front
 2:Insert_rear
 3:Deletion of first element
 4:Deletion of last element
 5:Deletion at specified position
 6:display_list
 7:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Insert_rear
 3:Deletion of first element
 4:Deletion of last element
 5:Deletion at specified position
 6:display_list
 7:Exit
enter the choice
1
enter the item at front-end
40
```

```
 1:Insert_front
 2:Insert_rear
 3:Deletion of first element
 4:Deletion of last element
 5:Deletion at specified position
 6:display_list
 7:Exit
enter the choice
6
40
30
20
10

 1:Insert_front
 2:Insert_rear
 3:Deletion of first element
 4:Deletion of last element
 5:Deletion at specified position
 6:display_list
 7:Exit
enter the choice
5
enter the position to delete
2
```


Data Structure 3C
Repl.it - SteelblueEmbellishedBook
repl.it/@PremaPrema/SteelblueEmbellishedBookmarks#main.c
Apps Gmail YouTube Maps News Translate SEM HALL TICKET
PremaPrema / SteelblueEmbellished...
Stop
Upgrade
Share
Files
main.c
6:display_list
7:Exit
enter the choice
3
item deleted at front-end is=40
1:Insert_front
2:Insert_rear
3:Deletion of first element
4:Deletion of last element
5:Deletion at specified position
6:display_list
7:Exit
enter the choice
4
iten deleted at rear-end is 10
1:Insert_front
2:Insert_rear
3:Deletion of first element
4:Deletion of last element
5:Deletion at specified position
6:display_list
7:Exit
enter the choice
Type here to search
ENG
19:27
08-12-2020

written:--

PAGE NO.
DATE: / /

```
#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)
{
pf ("mem full \n")
}
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;
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)
{
pf ("list is empty \n");
return first;
}
temp = first
temp = temp->link;
pf ("item deleted = %d \n", first->info);
free (first);
return temp;
}
NODE delete_rear (NODE first)
{
NODE cur, prev;
if (first == NULL)
{
pf ("list is empty\n");
return first;
}
if (first->link == NULL)
{
pf ("item deleted is %d \n", first->info);
return NULL;
}
```

```
void display(NODE first)
{
NODE temp;
if (first == NULL)
pf ("list is empty \n");
for (temp = first; temp != NULL;
temp = temp->link)
{
pf ("%d \n", temp->info);
}
}
int main()
{
int item, choice, pos;
NODE first = NULL;
NODE second = NULL;
for (;;)
{
pf ("\n1. insert front \n2 insert rear \n3 deletion
of first element \n4: deletion of last element
\n5 deleion @ specified position \n6 display
\n7 exit \n");
pf ("enter the choice \n");
sf ("%d ", &choice);
switch (choice)
```

```
{
case 1: pf ("enter the item @ front_end \n
sf ("%d", &item);
first = item_front (first, item);
break;
case 2: pf ("enter the item @ rear_end \n
sf ("%d", &item);
first = insert_rear (first, item);
break.
case 3: first = delete_front (first)
break.
case 4: first = delete_rear (first);
break;
case 5;
pf ("enter the position to delete \n");
scanf ("%d", &pos);
first = delete_pos (pos, first);
break;
case 6; display (first);
break;
default: exit (0);
break;
}
}
}
```

7.WAP Implement Single Link List with following operations

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

program:--

Lab 7:--
Program:-

```
#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\n");
exit(0);
}
return x;
}
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;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty");
for(temp=first;temp!=NULL;temp=temp->link)
{
```

```
printf("%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;
}NODE asc(NODE first)
{
NODE prev=first;
NODE cur=NULL;
int temp;
if(first== NULL) {
return 0;
}
else {
while(prev!= NULL) {
cur = prev->link;
while(cur!= NULL) {
if(prev->info > cur->info) {
temp = prev->info;
prev->info = cur->info;
cur->info = temp;
}
```

```
cur = cur->link;
}
prev= prev->link;
}}
return first;
}
NODE des(NODE first)
{
NODE prev=first;
NODE cur=NULL;
int temp;
if(first==NULL) {
return 0;
}
else {
while(prev!= NULL) {
cur = prev->link;
while(cur!= NULL) {
if(prev->info < cur->info) {
temp = prev->info;
prev->info = cur->info;
cur->info = temp;}
cur = cur->link;
}
prev= prev->link;
}
}
return first;
}
void main()
{
int item,choice,pos,i,n,option;
NODE first=NULL,a,b;
for(;;)
{
printf("1.insert_front\t 2.concatenation\t 3.reverse the list\t 4:Sort the list\t 5.dislay\t
6.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");
scanf("%d",&item);
first=insert_rear(first,item);
```

```
break;
case 2: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);
display(a);
break;
case 3:first=reverse(first);
display(first);
break;
case 4:printf("Press 1 for ascending sort and 2 for descending sort:\n");
scanf("%d",&option);if(option==1)
first=asc(first);
if(option==2)
first=des(first);
break;
case 5:display(first);
break;
default:exit(0);
}
}
}
```

screenshots:--

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
1
Enter the item at front-end
10

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
```

```
1
Enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
3
Enter the item at rear-end
40

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
```

```
Enter the choice:-
3
Enter the item at rear-end
50

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
8
20
10
40
50

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
```

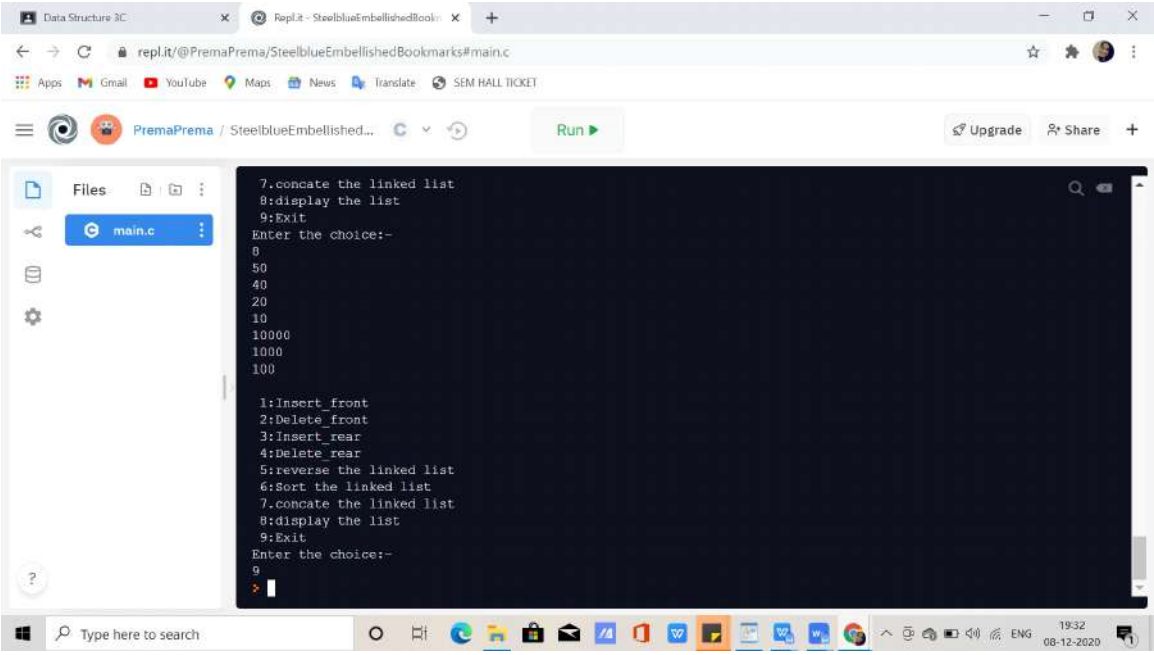
```
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
5

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
8
50
40
10
20
```

```
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
6
Press 1 for ascending sort and 2 for descending sort:
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
8
10
20
40
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
6
Press 1 for ascending sort and 2 for descending sort:
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
```

```
Enter the choice:-
8
50
40
20
10

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
7
Create a second list
Enter the number of elements in second list
3

 Press 1 to insert front and 2 to insert rear
1
Enter the item at front-end
```

```
 Press 1 to insert front and 2 to insert rear
1
Enter the item at front-end
1000

 Press 1 to insert front and 2 to insert rear
1
Enter the item at front-end
10000

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:reverse the linked list
 6:Sort the linked list
 7.concate the linked list
 8:display the list
 9:Exit
Enter the choice:-
8
50
40
20
10
```


Data Structure 3C
Repl.it - SteelblueEmbellishedBook
repl.it/@PremaPrema/SteelblueEmbellishedBookmarks#main.c
Apps
Gmail
YouTube
Maps
News
Translate
SEM HALL TICKET
PremaPrema / SteelblueEmbellished...
Run
Upgrade
Share
Files
main.c
7.concate the linked list
8:display the list
9:Exit
Enter the choice:-
8
50
40
20
10
10000
1000
100
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:reverse the linked list
6:Sort the linked list
7.concate the linked list
8:display the list
9:Exit
Enter the choice:-
9
Type here to search
ENG
19:32
08-12-2020

written:--

Lab 7.

```
#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;
}
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;
}
void display (NODE first)
{
```

```
NODE asc (NODE first)
{ NODE prev = first;
  NODE curr = NULL;
  int temp;
if (first == NULL) {
  return 0;
  }
else {
while (prev != NULL) {
 curr = prev -> link;
 while (curr != NULL) {
if (prev -> info > curr -> info) {
temp = prev -> info;
prev -> info = curr -> info;
curr -> info = temp;
 }
curr = curr -> link;
 }
prev = prev-link; }
return first; }

NODE des (NODE first) {
NODE prev = first;
 NODE curr = NULL;
int temp;
if (first == NULL) {
return 0;
 }
else { while (prev != NULL) {
curr = prev -> link;
while (curr != NULL) {
if (prev -> info < curr -> info) {
```

```
temp = prev -> info;
prev -> info = cur -> info;
cur -> info = temp; }
cur = cur -> link; }
prev = prev -> link; }
}
return first;
}
void main () {
int item, choice, pos, i, n, option;
NODE first = NULL, a, b;
for (;;)
{
printf (" 1. insert_front \t 2. concatenation \t 3.
reverse the list \t 4: sort the list \t 5. display
\t 6. exit \n ");
printf ("enter the choice \n");
scanf ("%d", &choice);
switch (choice)
{
case 1 : printf ("enter the item \n");
    scanf ("%d", &item);
    first = insert_rear (first, item);
    break;
case 2: printf (" enter the no of nodes in a \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);
display(a);
break;
case 3: first = reverse (first);
    display(first); break;
case 4: prt ("Press 1 for asc and 2 for desc\n")
  sf ("%d", &option);
  if (option == 1)
   first = asc(first);
  if (option == 2)
  first = dis(first);
   break;
case 5: display (first); break;
default: exit(0);
}
}
}
```

LAB-8:--

WAP to implement Stack & Queues using Linked Representation

PROGRAM

```
#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\n");exit(0);

}

return x;

}

void freenode(NODE x)

{

free(x);

}
```

```
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("item deleted at front-end is=%d\n",first->info);free(first);
return temp;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
int main()
{
int item,choice,pos;
NODE first=NULL;for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
```

```
switch(choice)
{
case 1:printf("enter the item at rear-end\n");
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;
}
}
}
```

IMPLEMENTING STACKS:--

program:--

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

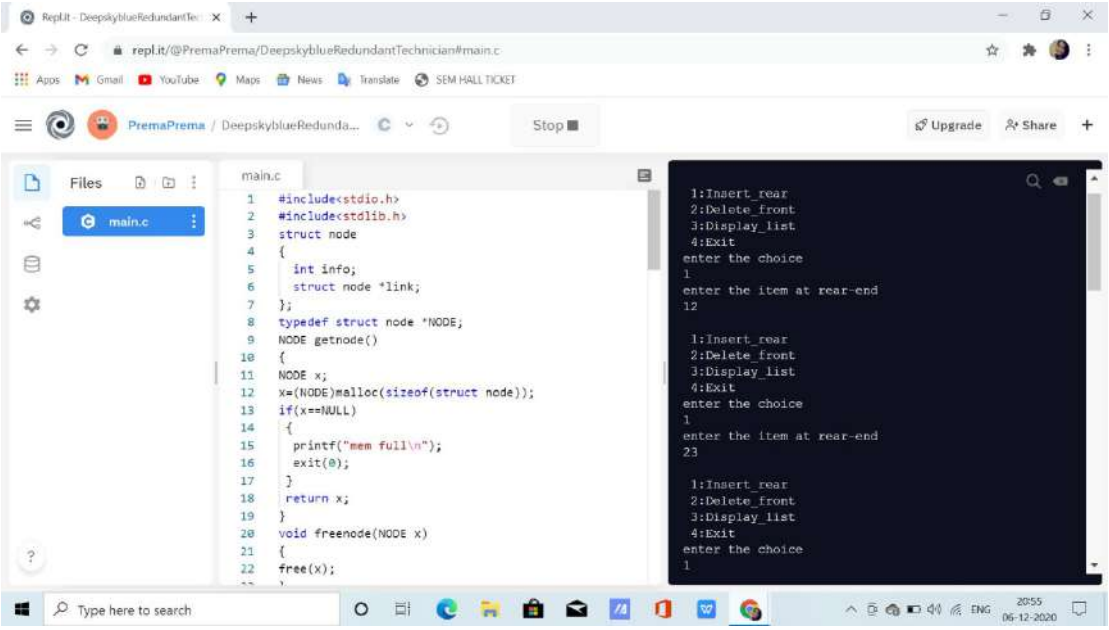
```
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("mem full\n");
exit(0);
}
return x;
}
void freenode(NODE x){
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
```

```
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{NODE temp;
if(first==NULL)
{
printf("stack 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;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
```

```
printf("stack empty cannot display

items\n");for(temp=first;temp!=NULL;temp=temp->link)

{

printf("%d\n",temp->info);

}

}

int main()

{

int item,choice,pos;

NODE first=NULL;

for(;;)

{

printf("\n 1:Insert_front\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{case 1:printf("enter the item at front-end\n");

scanf("%d",&item);

first=insert_front(first,item);

break;

case 2:first=delete_front(first);

break;
```

```
case 3:display(first);

break;

default:exit(0);

break;

}

}

}
```

screenshots:--



repl.it/@PremaPrema/DeepskyblueRedundantTechnician#main.c

PremaPrema / DeepskyblueRedunda...

Stop ■ Upgrade Share

Files

main.c

```c
}
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");
```

```
enter the item at rear-end
24

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at rear-end
25

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
12
23
24
25

 1:Insert_rear
 2:Delete_front
```

Type here to search

ENG 20:55 06-12-2020

repl.it/@PremaPrema/DeepskyblueRedundantTechnician#main.c

PremaPrema / DeepskyblueRedunda...

Stop ■ Upgrade Share

Files

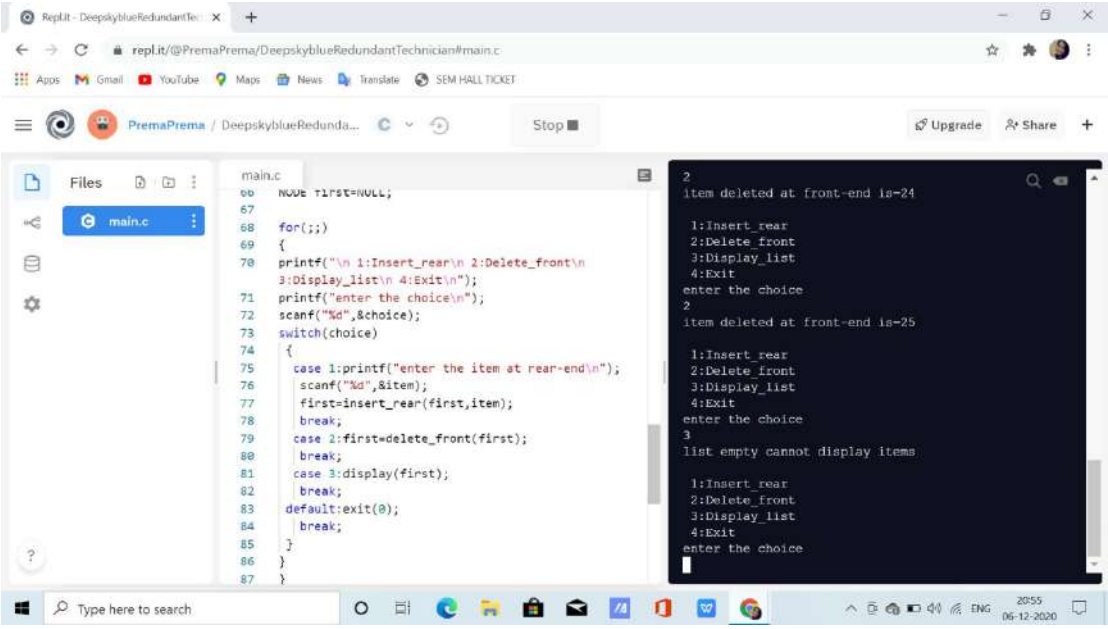
main.c

```c
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",
first->info);
free(first);
return temp;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("list empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
int main()
{
int item,choice,pos;
```

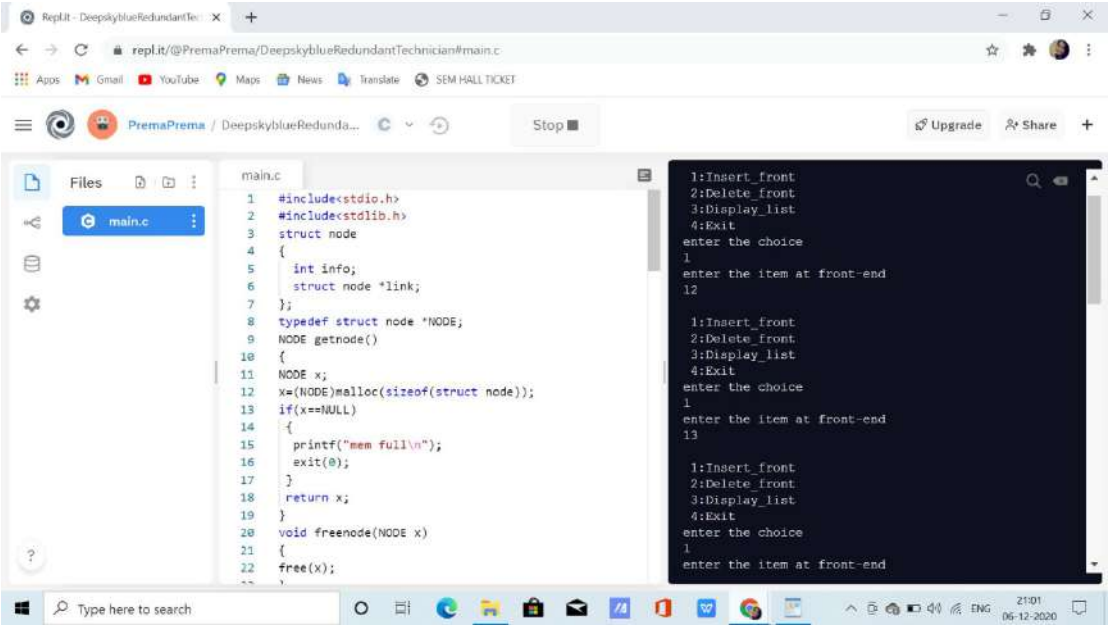
```
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=12

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=23

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=24

 1:Insert_rear
 2:Delete_front
 3:Display_list
```

Type here to search

ENG 20:55 06-12-2020



Screenshot of stacks:



```c
temp=temp->link;
printf("item deleted at front-end is=%d\n",
first->info);
free(first);
return temp;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("stack empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
int main()
{
int item,choice,pos;
NODE first=NULL;

for(;;)
```

```
enter the choice
2
item deleted at front-end is=13

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=12

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
stack is empty cannot delete

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
```

```c
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n
3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item at front-end\n");
   scanf("%d",&item);
   first=insert_front(first,item);
   break;
  case 2:first=delete_front(first);
   break;
  case 3:display(first);
   break;
 default:exit(0);
   break;
 }
}
}
```

```
2
item deleted at front-end is=12

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
stack is empty cannot delete

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
stack empty cannot display items

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
```

Written:

Lab 8

① Stack implementation.

```
#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\n");
exit(0); }
return x; }
void freenode(NODE x)
{
free(x); }
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)
{ pf("LE ");
return first;
}
temp = first
temp = temp -> link;
pf("item deleted @ front end is - %d\n", first->
free(first);
return temp;
}
void display (NODE first)
{ NODE temp;
if (first == NULL)
pf ("LE ");
for (temp= first; temp != NULL; temp=temp->link){
pf ("%d " temp->info);
}
}

int main() {
int item, choice, pos;
NODE first = NULL;
for (;;)
{
pf ("\n 1. insert_rear \n2: Delete_front \n3: display
\n4 Exit);
pf ("enter the choice");
switch (choice)
{
```

```
case 1: pf ("enter the item at rear end)
        sf ("%d", &item);
first = insert_rear (first, item);
  break;
case 2: first = delete_front (first);
  break;
case 3: display (first);
    break;
default : exit (0);
  break;
  }
  }
}
```

LAB 9:-

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

PROGRAMS:-

LAB 9:-

Program:-

```
#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("mem 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;
}
```

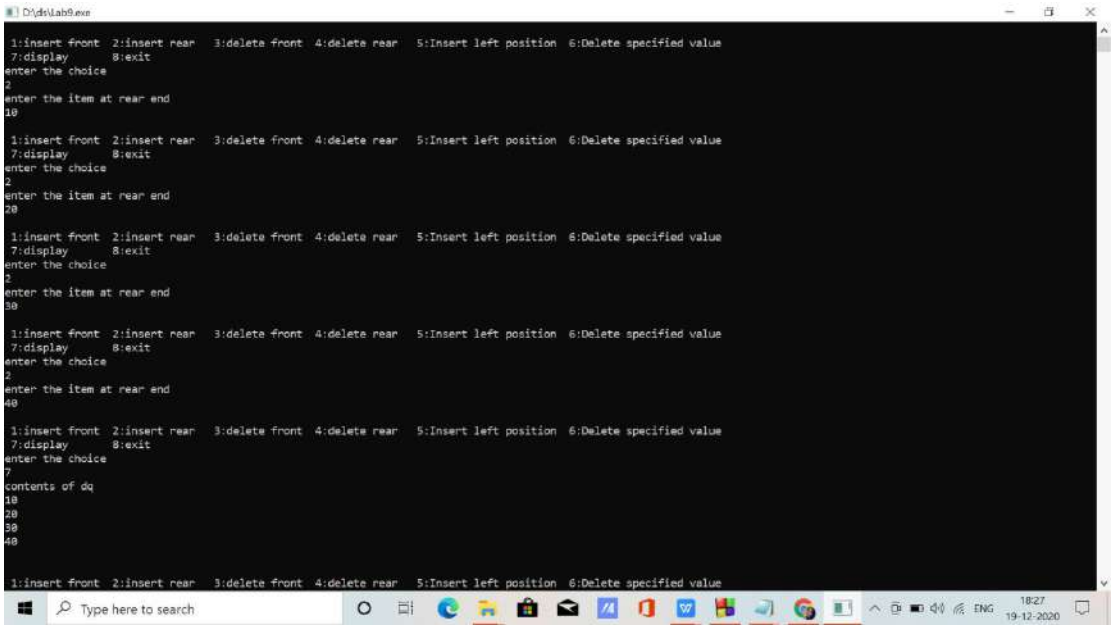
```
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;}
NODE ddelete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("dq empty\n");
return head;
}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

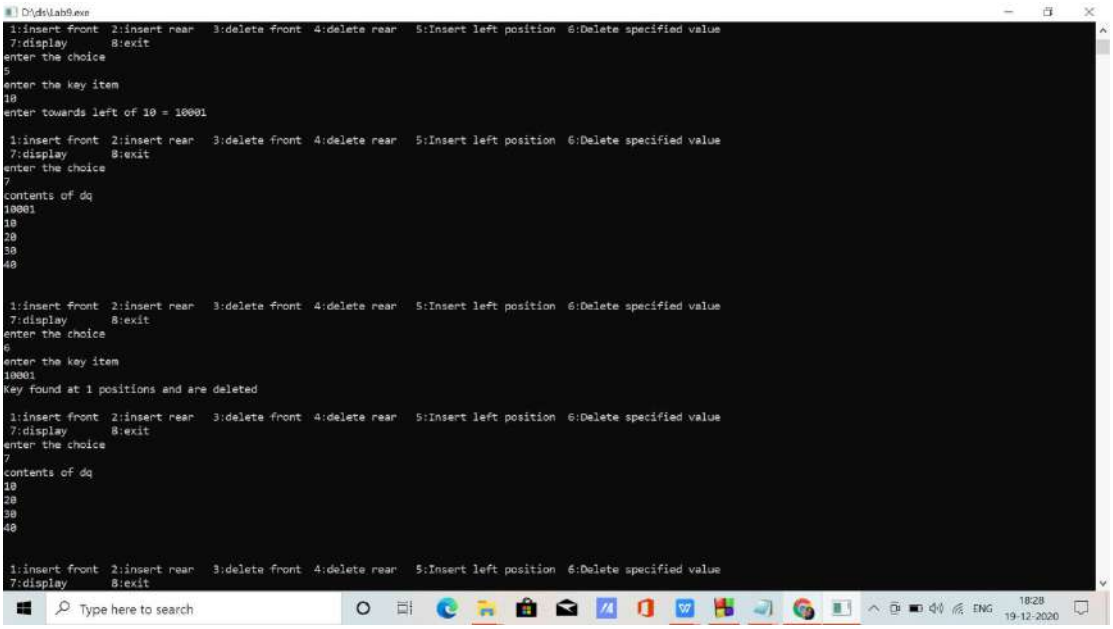
```
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
printf("key not found\n");return head;
}
prev=cur->llink;
printf("enter towards left of %d = ",item);
temp=getnode();
scanf("%d",&temp->info);
prev->rlink=temp;
temp->llink=prev;
cur->llink=temp;
temp->rlink=cur;
return head;
}
NODE delete_specified_value(int item,NODE head)
{
NODE prev,cur,next;
int count;
if(head->rlink==head)
{
printf("List is empty");
return head;
}
count=0;
cur=head->rlink;
while(cur!=head){
if(item!=cur->info)
cur=cur->rlink;
else
{
count++;
prev=cur->llink;
```

```
next=cur->rlink;
prev->rlink=next;
next->llink=prev;
freenode(cur);
cur=next;
}
}
if(count==0)
printf("key not found");
else
printf("Key found at %d positions and are deleted\n", count);
return head;
}
void display(NODE head)
{NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of dq\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\n",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("\n 1:insert front\t 2:insert rear\t 3:delete front\t 4:delete rear\t 5:Insert left position\t
6:Delete specified value\n 7:display\t 8:exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item at 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);
head=insert_leftpos(item,head);
break;
case 6:printf("enter the key item\n");
scanf("%d",&item);
head=delete_specified_value(item,head);break;
case 7: display(head);
break;
default:exit(0);
}
}
}
```

OUTPUT SCREENSHOT:-





```
D:\ds\Lab9.exe
 1:insert front  2:insert rear   3:delete front  4:delete rear   5:Insert left position  6:Delete specified value
 7:display       8:exit
enter the choice
5
enter the key item
10
enter towards left of 10 = 10001

 1:insert front  2:insert rear   3:delete front  4:delete rear   5:Insert left position  6:Delete specified value
 7:display       8:exit
enter the choice
7
contents of dq
10001
10
20
30
40


 1:insert front  2:insert rear   3:delete front  4:delete rear   5:Insert left position  6:Delete specified value
 7:display       8:exit
enter the choice
6
enter the key item
10001
Key found at 1 positions and are deleted

 1:insert front  2:insert rear   3:delete front  4:delete rear   5:Insert left position  6:Delete specified value
 7:display       8:exit
enter the choice
7
contents of dq
10
20
30
40


 1:insert front  2:insert rear   3:delete front  4:delete rear   5:Insert left position  6:Delete specified value
 7:display       8:exit
```

WRITTEN:-

Doubly linked list:-

```
#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("mem full \n");
exit(0); }
return x; }
void frenode(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) {
pf ("dq empty \n");
return head; }
cur = head -> rlink;
next = cur -> rlink;
head -> rlink = next;
next -> llink = head;
pf ("the node deleted is %d", cur -> info);
freenode(cur);
return head;
}
NODE ddelete_rear (NODE head)
{
NODE cur, prev;
if (head -> rlink == head)
{
pf (" dq empty \n");
return head;
}
```

```
curr = head -> llink;
prev = curr -> llink;
head -> llink = prev;
prev -> rlink = head;
pf ("the node deleted is %d", curr -> info);
freenode(curr);
return head;
}
NODE insert_restpos(int item, NODE head)
{
NODE temp, curr, prev;
if (head -> rlink == head) {
printf ("list empty\n");
return head;
}
curr = head -> rlink;
while (curr != head)
{
printf ("key not found\n");
return head; }
temp -> llink = prev;
curr -> llink = temp;
temp -> rlink = curr;
return head;
}
NODE delete_specified_value (int item, NODE head)
{
NODE prev, curr, next;
int count;
if (head -> rlink == head)
{
pf ("\n
```

```
NODE temp;
if (first == NULL)
printf("list empty");
for (temp = first; temp != NULL; temp = temp->link)
{
printf("%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() {
  NODE head, last;
int item, choice;
 head = getnode();
 head -> rlink = head;
 head -> llink = head;
for(;;)
{
pf("1F \t2 1R \t3 DF \t4 DR \t5 insert left
 pos \t6 . delete specified value \t7 : display
 \t8 exit \n");
 switch(choice);
 { case 1 : pf(" enter the item @ front end \n");
scanf("%d", &item);
last = dinsert_front (item, head);
 break;
 case 2: printf (" enter the item @ rear
   end \n");
 sf("%d", &item);
last = dinsert_rear(item, head);
 break;
case 3: last = ddelete from(head);
   break;
case 4: last = ddelete rear(head); break;
case 5: pf(" key \n"); sf("%d", &item);
case 6: head = delete_specified_value (item, head)
 break;
case 7 : display (head);
break;
default : exit(0);
}
```

LAB 10:-

Program:-

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.

Program:-

```
#include<stdio.h>

#include<stdio.h>

#include<stdlib.h>

struct node

 {

  int info;

  struct node *rlink;

  struct node *llink;

 };

typedef struct node *NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

 {

  printf("mem full\n");

  exit(0);

 }

 return x;

}

void freenode(NODE x)
```

```
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
```

```
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
          printf("  ");
   printf("%d\n",root->info);
         display(root->llink,i+1);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
```

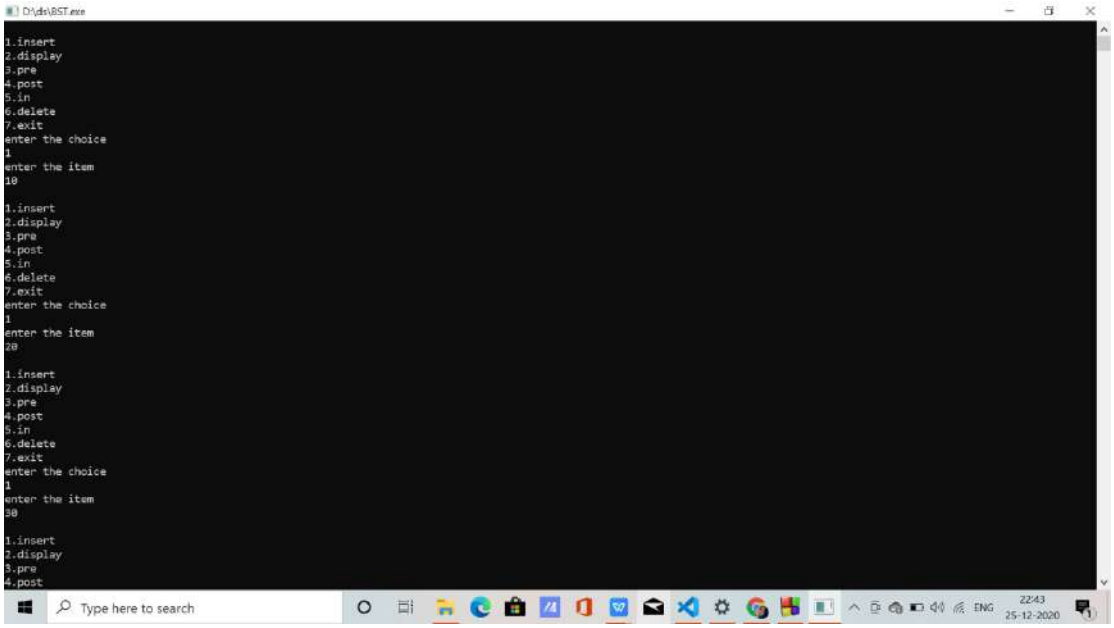
```
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

void preorder(NODE root)
{
```

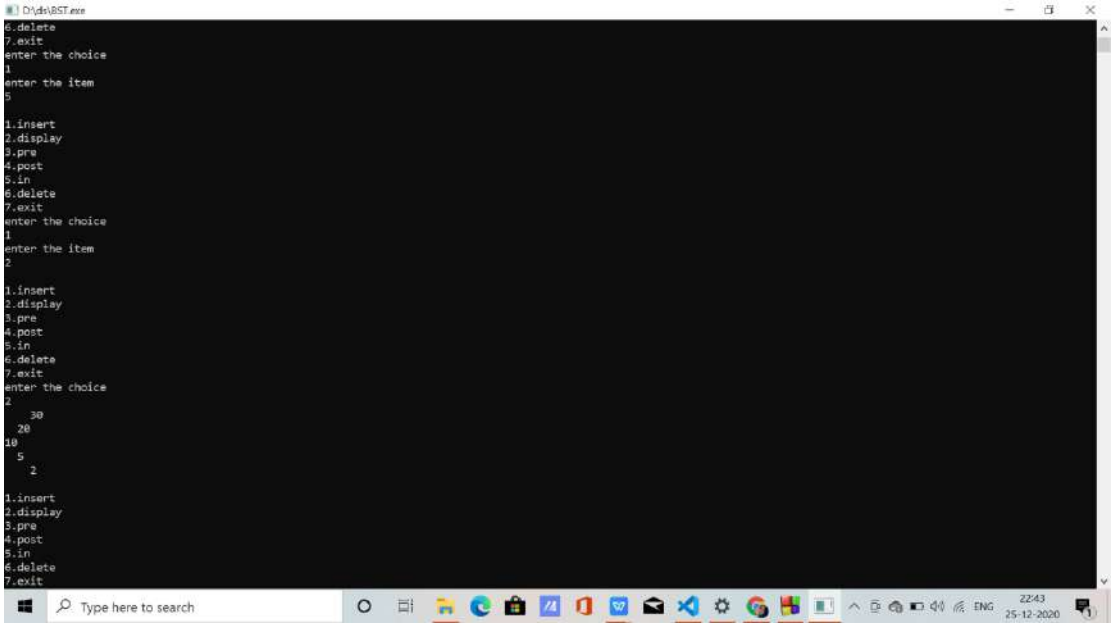
```
if(root!=NULL)
 {
  printf("%d\n",root->info);
  preorder(root->llink);
  preorder(root->rlink);
  }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {

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

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

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

```
 default:exit(0);

                break;

        }

      }

 }
```

Output:-





```
D:\ds\BST.exe
enter the choice
3
10
5
2
20
30

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
2
5
30
20
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
2
5
10
20
30

1.insert
2.display
3.pre
```

```
D:\ds\BST.exe
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
2

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    30
  20
10
  5

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

Written:-

Lab 10

```
#include<stdio.h>
#include <stdlib.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)
{
pf("memfull \n");
}
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 = cur -> link~~;
  cur = (item < cur -> info)? cur -> llink : cur -> rlink;
 }
 if (~~temp~~ item < prev -> info)
  prev -> llink = temp;
 else
 prev -> rlink = temp;
  return root;
 }
 void display (NODE root, int i)
 {
  int j;
 if (root != NULL)
  {
 display (root -> rlink, i+1);
  for (j=0; j<i; j++)
   printf ("  ");
  printf ("%d\n", root -> info);
   display (root -> llink, i+1);
   }
  }

NODE delete (NODE root, int item)
 {

NODE cur, parent, q, suc;
if (root == NULL)
 {
 printf ("empty\n");
```

```
return root;
}
parent = NULL;
cur = root;
while (cur != NULL && item != cur -> info)
{
parent = cur
cur = (item < cur -> info) ? cur -> llink : cur -> rlink;
}
if (cur == NULL)
pf ("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;
}
freenode(cur);
return root;
}
```

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

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

int main()
{
int item, choice;
 NODE root = NULL;
 for(;;)
```

```
{
Pf("\n1. insert \n2 display \n3 pre order \n4 post
\n5 in \n6 delete \n7 exit \n");
Pf ("enter the choice \n");
sf("%d", &choice);
switch (choice)
{
case 1: prinf ("enter the item \n");
      sf ("%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");
      sf ("%d", &item);
      root = delete (root, item)
      break.
  default. exit (0);
      break;
    }
  }
}
```

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

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

int main()
{
int item, choice;
 NODE root = NULL;
 for(;;)
```