

## 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:--

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

```
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
```

```
23 {
24 int i;
25 if(top==-1)
26 {
27 printf("stack is empty\n");
28 return;
29 }
30 printf("contents of the stack\n");
31 for(i=top; i>=0; i--)
32 {
33 printf("%d\n",s[i]);
34 }
35 }
36 int main()
37 {
38 int item_deleted;
39 int choice;
40
41 for(;;)
42 {
43 printf("\n 1:push\n 2:pop\n 3:display\n 4:exit\n");
44 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
```

```
main.c
45 scanf("%d", &choice);
46 switch(choice)
47 {
48 case 1:printf("enter the item to be inserted\n");
49         scanf("%d",&item);
50         push();
51         break;
52 case 2:item_deleted=pop();
53         if(item_deleted==--1)
54             printf("stack is empty\n");
55         else
56             printf("item deleted is %d\n",item_deleted);
57         break;
58 case 3:display();
59         break;
60 default:exit(0);
61 }
62 }
63 }
64 }
```

```
2:pop
3:display
4:exit
enter the choice
2
item deleted is 30

1:push
2:pop
3:display
4:exit
enter the choice
2
item deleted is 20

1:push
2:pop
3:display
4:exit
enter the choice
2
item deleted is 10

1:push
2:pop
```

```
item deleted is 20

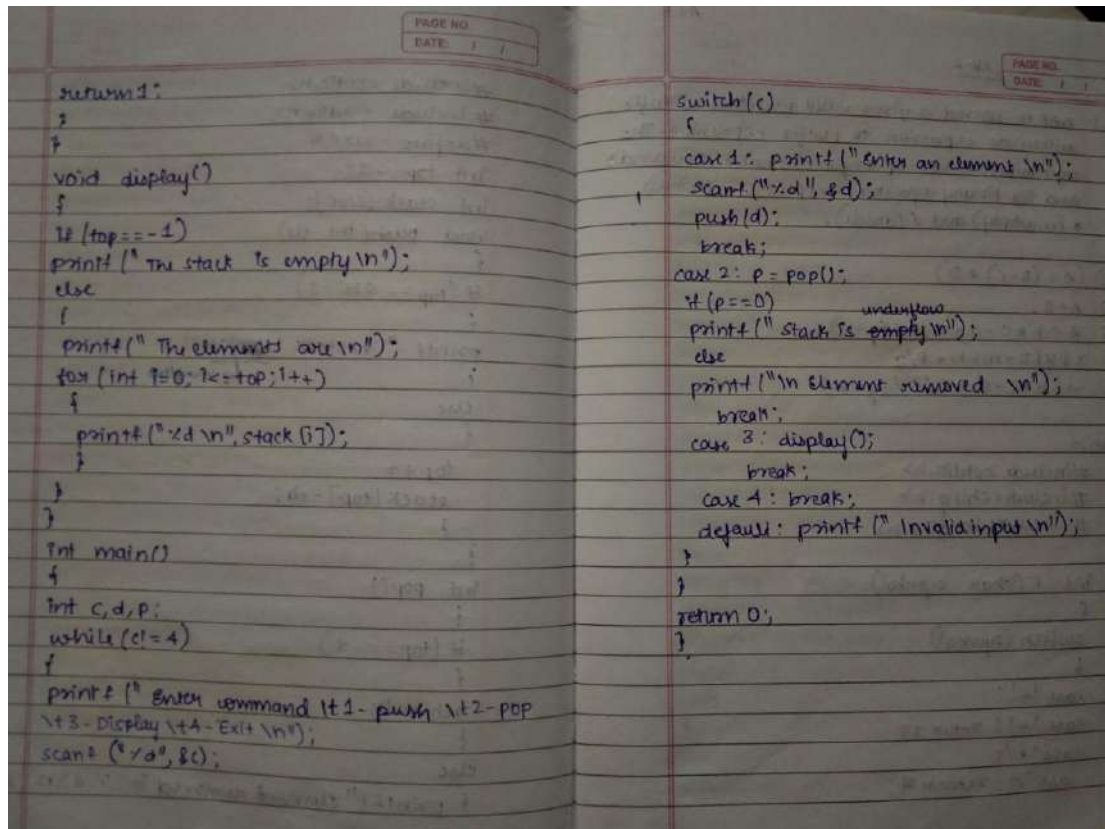
1:push
2:pop
3:display
4:exit
enter the choice
2
item deleted is 10

1:push
2:pop
3:display
4:exit
enter the choice
2
stack is empty

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

Written program:-

```
#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 full \n");
        // overflow
    }
    else
    {
        top++;
        stack[top] = ele;
    }
}
int pop()
{
    if (top == -1)
    {
        return 0;
    }
    else
    {
        printf("element removed is: %d \n", stack[top]);
    }
}
```



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:

```

main.c
1
2 #include<stdio.h>
3 #include<string.h>
4 #include<stdlib.h>
5 int F(char symbol)
6 {
7     switch(symbol)
8     {
9         case '+':
10        case '-':return 2;
11        case '*':
12        case '/':return 4;
13        case '^':
14        case '$':return 5;
15        case '(':return 0;
16        case '#':return -1;
17        default:return 8;
18    }}
19 int G(char symbol)
20 {
21     switch(symbol)
22     {
23         case '+':

```

```

> clang-7 -pthread -lm -o main main.c
> ./main
Enter the valid infix expression a+b/cd^e
The postfix expression is
abcde^/+
>

```

```

main.c
23 case '+':
24 case '-':return 1;
25 case '*':
26 case '/':return 3;
27 case '^':
28 case '$':return 6;
29 case '(':return 9;
30 case ')':return 0;
31 default:return 7;
32 }}
33 void infix_postfix(char infix[],char postfix[])
34 {
35     int top,i,j;
36     char s[30],symbol;
37     top=-1;
38     s[++top]='#';
39     j=0;
40     for(i=0;i<strlen(infix);i++)
41     {
42         symbol=infix[i];
43         while(F(s[top])>G(symbol))
44         {

```

The screenshot shows a web-based IDE with a browser window at the top displaying the URL `repl.it/@PremaPrema/DeafeningPointedKernelmode#main.c`. The IDE interface includes a file explorer on the left with a file named `main.c`. The main editor displays the following C code:

```
main.c
44 {
45     postfix[j]=s[top--];
46     j++;
47 }
48 if(F(s[top])!=G(symbol))
49     s[++top]=symbol;
50 else
51     top--; }
52 while(s[top]!='#')
53 {
54     postfix[j++]=s[top--];
55 }
56 postfix[j]='\0';
57 }
58 int main()
59 {
60     char infix[20];
61     char postfix[20];
62     printf("Enter the valid infix expression ");
63     scanf("%s",infix);
64     infix_postfix(infix , postfix );
65     printf("The postfix expression is \n");
66     printf("%s\n",postfix);
67 }
```

written program:--

LAB-2

Qn:- 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).

①  $(A + (B - C) * D)$   
②  $A + B$   
③  $A \wedge b * c - d + e / f / (g + h)$   
④  $x \wedge y \wedge z - m + n + p / q$   
⑤  $a \wedge b * c - d + e / f / (g + h)$

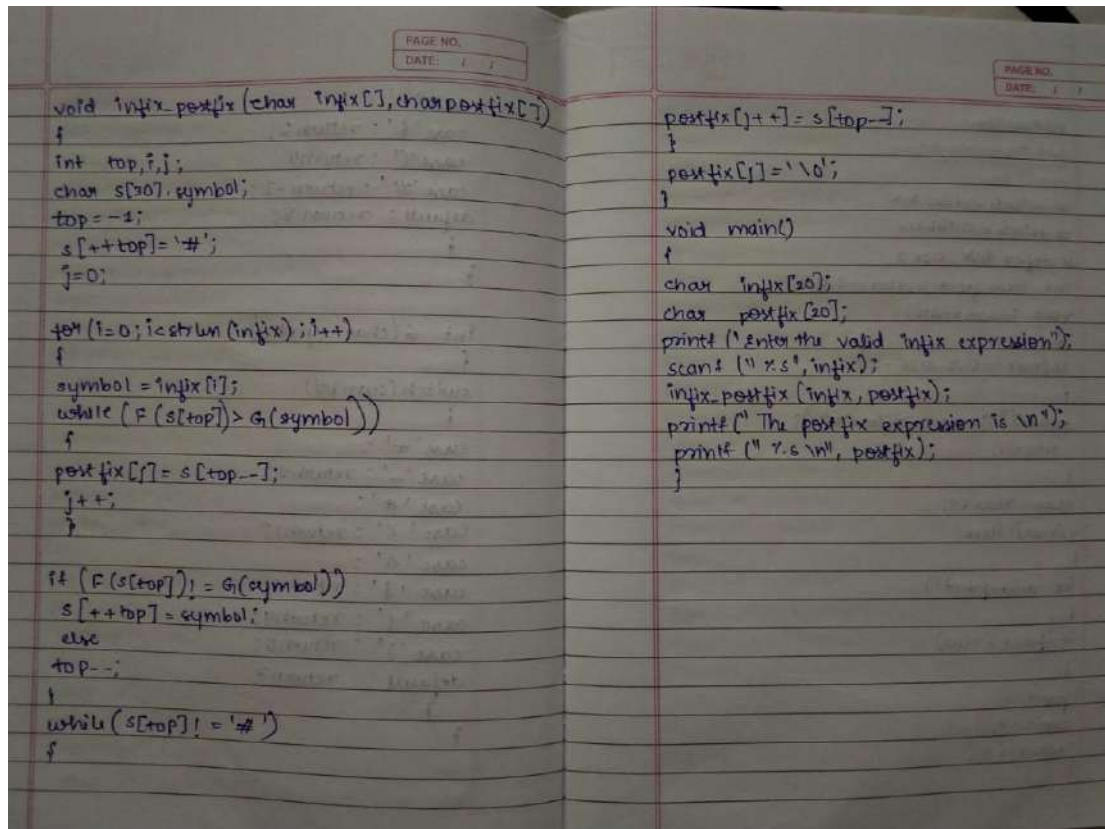
program:-

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

int F(char symbol)
{
    switch (symbol)
    {
        case '+':
        case '-': return 2;
        case '*':
        case '/': return 4;
    }
}
```

```
case 'd':
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;
    }
}
```



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:

```

main.c
1  #include<stdio.h>
2  #include<stdlib.h>
3  #define QUEUE_SIZE 3
4  int item,front=0,rear=-1,q[10];
5  void insertrear()
6  {
7  if(rear==QUEUE_SIZE-1)
8  {
9  printf("queue overflow\n");
10 return;
11 }
12 rear=rear+1;
13 q[rear]=item;
14 }
15 int deletefront()
16 {
17 if(front>rear) return -1;
18 return q[front++];
19 }
20 void displayQ()
21 {
22 int i;
23 if(front>rear)

```

```

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

```

```

main.c
23 if(front>rear)
24 {
25 printf("queue is empty\n");
26 return;
27 }
28 printf("Contents of queue \n");
29 for(i=front;i<=rear;i++)
30 {
31 printf("%d\n",q[i]);
32 }
33 }
34 int main()
35 {
36 int choice;
37 for(;;)
38 {
39 printf
40 ("1:insertrear\n2:deletefront\n3:display\n4:exit\n");
41 printf("enter the choice\n");
42 scanf("%d",&choice);
43 switch(choice)

```

```

30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
40
queue overflow

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

1:insertrear
2:deletefront

```

The screenshot shows a web-based IDE with the following components:

- Files Panel:** Contains a file named `main.c`.
- Editor:** Displays the following C code in `main.c`:

```
44 {  
45     case 1:printf("enter the item to be inserted\n");  
46     scanf("%d",&item);  
47     insertrear();  
48     break;  
49     case 2:item=deletefront();  
50     if(item== -1)  
51         printf("queue is empty\n");  
52     else  
53         printf("item deleted =%d\n",item);  
54     break;  
55     case 3:displayQ();  
56     break;  
57     default:exit(0);  
58 }  
59 }  
60 }  
61 }
```
- Output Panel:** Shows the program's execution output:

```
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
```

This screenshot shows the continuation of the program's execution from the previous state:

- Output Panel:** The output continues with:

```
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  
> []
```

written :

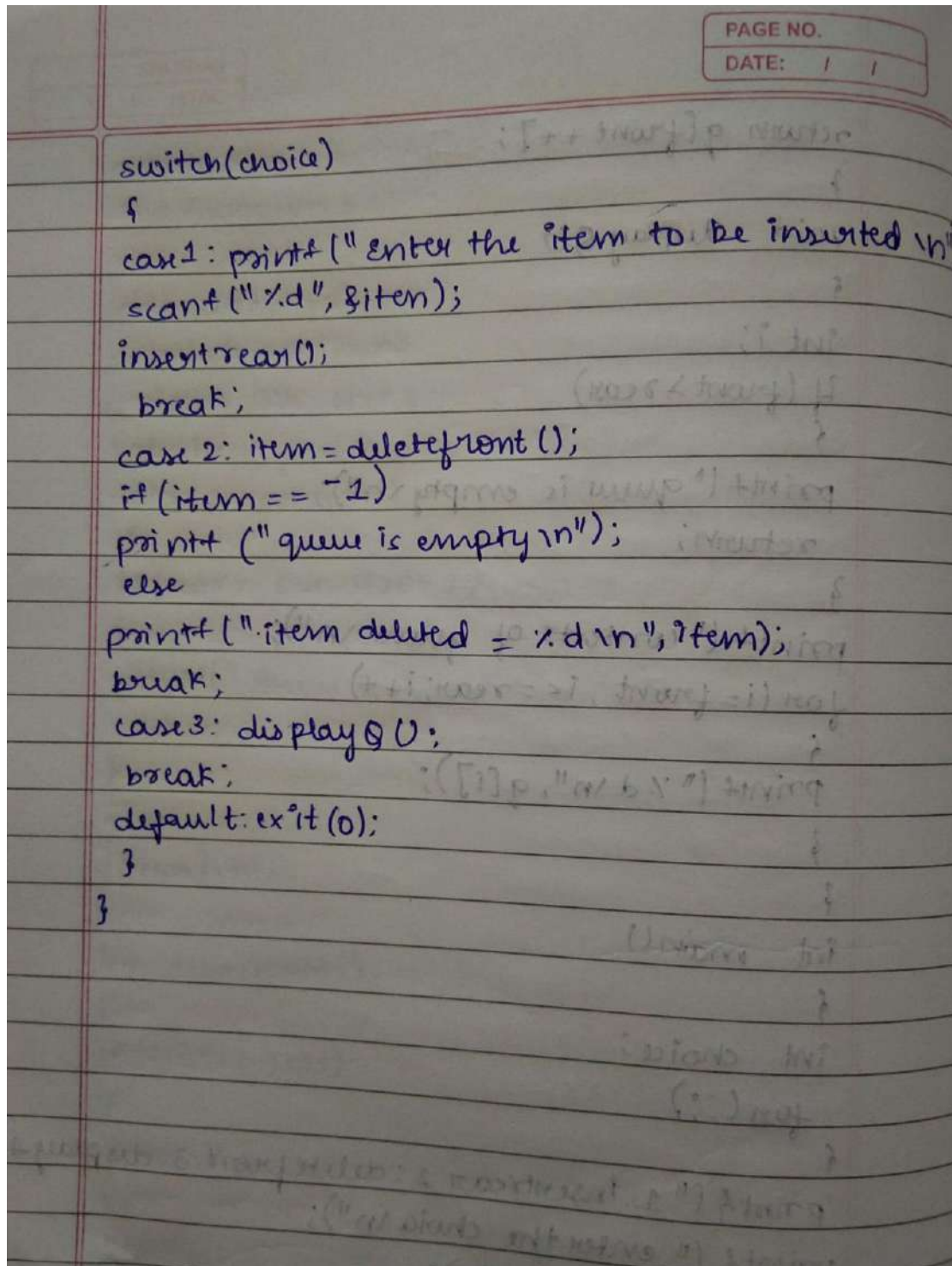


~~Lab Program 3~~

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()
{
    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. Insert rear 2. delete front 3. display\n");
        printf("Enter the choice\n");
        scanf("%d", &choice);
```



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

1 #include<stdio.h>
2 #include<stdlib.h>
3 #define QUE_SIZE 3
4 int item,front=0,rear=-1,q[QUE_SIZE],count=0;
5 void insertrear()
6 {
7     if(count==QUE_SIZE)
8     {
9         printf("queue overflow\n");
10        return;
11    }
12    rear=(rear+1)%QUE_SIZE;
13    q[rear]=item;
14    count++;
15 }
16 int deletefront()
17 {
18     if(count==0) return -1;
19     item=q[front];
20     front=(front+1)%QUE_SIZE;
21     count=count-1;
22     return item;
23 }
```

```
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

23 }
24 void displayQ()
25 {
26     int i,f;
27     if(count==0)
28     {
29         printf("queue is empty\n");
30         return;
31     }
32     f=front;
33     printf("contents of queue \n");
34     for(i=1;i<=count;i++)
35     {
36         printf("%d\n",q[f]);
37         f=(f+1)%QUE_SIZE;
38     }
39 }
40 int main()
41 {
42     int choice;
43
44 }
```

```
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
```

```
main.c
45 for(;;)
46 {
47     printf
48     ("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
49     printf("enter the choice\n");
50     scanf("%d",&choice);
51
52     switch(choice)
53     {
54     case 1:printf("enter the item to be inserted\n");
55             scanf("%d",&item);
56             insertrear();
57             break;
58     case 2:item=deletefront();
59             if(item==1)
60                 printf("queue is empty\n");
61             else
62                 printf("item deleted %d\n",item);
63             break;
64     case 3:displayQ();
65             break;
66     default:exit(0);
67     }
```

```
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
```

```
main.c
49 scanf("%d",&choice);
50
51 switch(choice)
52 {
53 case 1:printf("enter the item to be inserted\n");
54         scanf("%d",&item);
55         insertrear();
56         break;
57 case 2:item=deletefront();
58         if(item==1)
59             printf("queue is empty\n");
60         else
61             printf("item deleted %d\n",item);
62         break;
63 case 3:displayQ();
64         break;
65 default:exit(0);
66 }
67 }
68 }
69 }
```

```
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

```

written:--



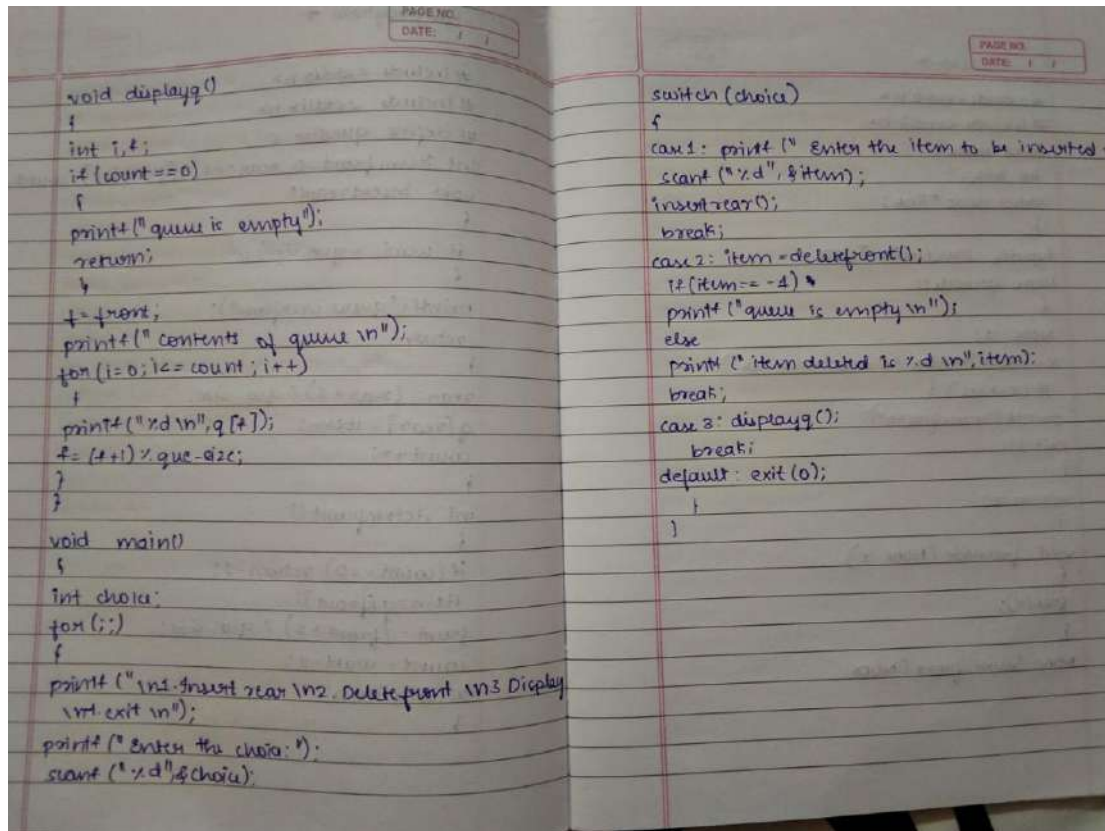
#### Lab program 4:

PAGE NO.

DATE: / /

```
#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");
        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;
}
```





5.WAP to Implement Singly Linked List with following operations

- Create a linked list.
- Insertion of a node at first position, at any position and at end of list.
- 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:--

The screenshot shows a Repl.it environment with a C++ program. The program is a menu-driven application for a linked list. The menu options are: 1: Insert at first position, 2: Insert at last position, 3: Insert at a specific position, 4: display\_list, 5: Exit. The program prompts the user to enter the choice, then the item to insert, and finally the position to insert (if applicable). The screenshot shows three iterations of the program's execution with different inputs.

```

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
1
enter the item at front-end
10

1:Insert at fist position
2:Insert at last position
3.Insert a specific position 4:display_list
5:Exit
enter the choice
2
enter the item at rear-end
100

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

```



```
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:--

```
#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("main fun\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 = first;
    if (first == NULL) {
        first = temp;
    }
    return first;
}
```

```
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 first)
{
    NODE temp;
    NODE p, q, 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;
}
if (count == pos)
{
    p->link = temp;
    temp->link = cur;
    return first;
}
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
```

```
printf("list empty cannot display item\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("\n 1. Insert @ front pos 2. Insert @ last pos 3. Insert @ specific pos 4. Display list\n");
        printf("enter choice:");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: printf("enter the item @ front end\n");
                    scanf("%d", &item);
                    first = insert_front(first, item);
                    break;
            case 2: printf("enter the item @ rear end\n");
                    scanf("%d", &item);
                    first = insert_rear(first, item);
                    break;
```

DATE: / /

```

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("item deleted at rear-end is %d",cur->info);

free(cur);

prev->link=NULL;

return first;

}

```

```

NODE delete_pos(int pos, NODE first)

```

```

{

NODE cur;

NODE prev;

```



```
int count;

if(first==NULL || pos<=0)

{

printf("invalid position \n");

return NULL;

}

if (pos==1)

{

cur=first;

first=first->link;

freenode(cur);

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;
```

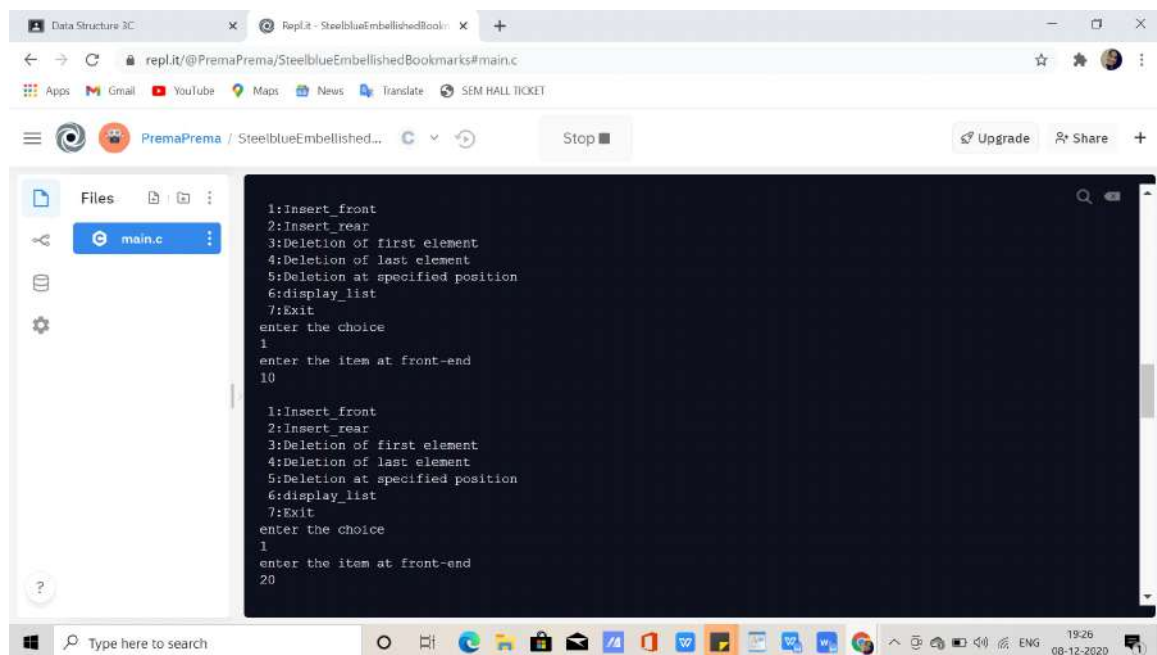
```
default:exit(0);
```

```
break;
```

```
}
```

```
}
```

```
}
```



The screenshot shows a web-based IDE (Repl.it) with a C program for a linked list. The program includes functions for insert, delete, and display. The terminal shows the program running with inputs 1 and 20.

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

Data Structure 2C x Repl.it - SteelblueEmbellishedBookmarks#main.c

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / SteelblueEmbellished... Stop Upgrade Share

Files main.c

```
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
```

Type here to search

19:26 08-12-2020

Data Structure 2C x Repl.it - SteelblueEmbellishedBookmarks#main.c

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / SteelblueEmbellished... Stop Upgrade Share

Files main.c

```
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
```

Type here to search

19:26 08-12-2020

The screenshot shows a web-based IDE (Repl.it) with a C program for a linked list. The program includes functions for inserting, deleting, and displaying list elements. The output shows two successful insertions and two deletions.

```
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
item 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
```

written:--

```
#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");
    }
    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)
{
    if (first == NULL)
        return first;
    temp = first;
    temp = temp->link;
    printf("Item deleted = %d\n", first->info);
    free(first);
    return temp;
}

NODE delete-rear(NODE first)
{
    NODE cur, prev;
    if (first == NULL)
        return first;
    printf("List is empty\n");
    return first;
    if (first->link == NULL)
    {
        printf("Item deleted is %d\n", first->info);
        return NULL;
    }
}

```

```

void display(NODE first)
{
    NODE temp;
    if (first == NULL)
        printf("List is empty\n");
    for (temp = first; temp != NULL; temp = temp->link)
    {
        printf("%d\n", temp->info);
    }
}

int main()
{
    int item, choice, pos;
    NODE first = NULL;
    NODE rear = NULL;
    for (;;)
    {
        printf("\n1. Insert front 2. Insert rear 3. Delete front 4. Delete rear 5. Delete at specified position 6. Display 7. Exit\n");
        printf("Enter the choice\n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: printf("Enter the item @ front-end\n");
                    item = getitem();
                    first = insert-front(first, item);
                    break;
            case 2: printf("Enter the item @ rear-end\n");
                    item = getitem();
                    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");
                    pos = getpos();
                    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.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 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:--

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

Files

- main.c

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

Type here to search

Microsoft Store

19:31 08-12-2020

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

Files

- main.c

```
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:concat 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:concat the linked list
```

Type here to search

19:31 08-12-2020

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

Files

- main.c

```
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.concat 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
```

Type here to search

19:31 08-12-2020

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

Files

- main.c

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

Type here to search

19:31 08-12-2020

Data Structure 2C x Repl.it - SteelblueEmbellishedBookmarks x +

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / SteelblueEmbellished... Run Upgrade Share +

Files main.c

```
4:Delete_rear
5:reverse the linked list
6:Sort the linked list
7.concat 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.concat the linked list
8:display the list
9:Exit
Enter the choice:-
8
10
20
40
```

Type here to search

19:31 08-12-2020

Data Structure 2C x Repl.it - SteelblueEmbellishedBookmarks x +

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / SteelblueEmbellished... Run Upgrade Share +

Files main.c

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

Type here to search

19:32 08-12-2020

The screenshot shows a Repl.it environment with a C program for linked list operations. The terminal output is as follows:

```
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:concat 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
```

The screenshot shows the same Repl.it environment after several operations. The terminal output is as follows:

```
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:concat the linked list
8:display the list
9:Exit
Enter the choice:-
8
50
40
20
10
```



The screenshot shows a web-based IDE (Repl.it) with a C program for linked list operations. The program includes a menu with options like Insert, Delete, Reverse, Sort, Concatenate, and Display. The output shows a list of numbers: 0, 50, 40, 20, 10, 10000, 1000, 100.

```
7:concat the linked list
8:display the list
9:Exit
Enter the choice:-
0
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:concat the linked list
8:display the list
9:Exit
Enter the choice:-
9
```

written:--

Lab 7.

```
#include <stdio.h>
```

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

```
struct node {
```

```
int into;
```

```
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_at_start (NODE first, int item)
```

```
{
```

```
NODE temp, cur;
```

```
temp = getnode();
```

```
temp->into = 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 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;
    }
}

```



```

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-front(first, item);
break;

```

```

case 2: printf("enter the no of nodes in list \n");
scanf("%d", &n);

```

```

a = NULL;

```

```

for(i=0; i<n; i++) {

```

```

printf("enter the item \n");

```

```

scanf("%d", &item);

```

```

a = insert-front(a, item);

```

```

}

```

```

printf("enter the no of nodes in 2nd");
scanf("%d", &n);
b = NULL;
for (i=0; i<n; i++) {
    printf("enter the item\n");
    scanf("%d", &item);
    b = insert_start(b, item);
}
a = concat(a, b);
display(a);
break;
case 3: first = reverse(first);
        display(first); break;
case 4: printf("press 1 for asc and 2 for desc\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);
}
}
}
}

```

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:--

The screenshot displays a Replit IDE interface. The left sidebar shows a file named 'main.c'. The main editor area contains the following C code:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node
4 {
5     int info;
6     struct node *link;
7 };
8 typedef struct node *NODE;
9 NODE getnode()
10 {
11     NODE x;
12     x=(NODE)malloc(sizeof(struct node));
13     if(x==NULL)
14     {
15         printf("mem full\n");
16         exit(0);
17     }
18     return x;
19 }
20 void freenode(NODE x)
21 {
22     free(x);
23 }
```

The right sidebar shows the execution output, which is a terminal window. It displays the menu options and user input for three separate runs of the program:

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
12

1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
23

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

Replit - DeepskyblueRedundantTec... x +

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / DeepskyblueRedunda... Stop Upgrade Share +

Files main.c

```
23 }
24 NODE insert_rear(NODE first,int item)
25 {
26     NODE temp,cur;
27     temp=getnode();
28     temp->info=item;
29     temp->link=NULL;
30     if(first==NULL)
31         return temp;
32     cur=first;
33     while(cur->link!=NULL)
34         cur=cur->link;
35     cur->link=temp;
36     return first;
37 }
38
39 NODE delete_front(NODE first)
40 {
41     NODE temp;
42     if(first==NULL)
43     {
44         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

Replit - DeepskyblueRedundantTec... x +

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

Apps Gmail YouTube Maps News Translate SEM HALL TICKET

PremaPrema / DeepskyblueRedunda... Stop Upgrade Share +

Files main.c

```
45     return first;
46 }
47 temp=first;
48 temp=temp->link;
49 printf("item deleted at front-end is=%d\n",
first->info);
50 free(first);
51 return temp;
52 }
53 void display(NODE first)
54 {
55     NODE temp;
56     if(first==NULL)
57         printf("list empty cannot display items\n");
58     for(temp=first;temp!=NULL;temp=temp->link)
59     {
60         printf("%d\n",temp->info);
61     }
62 }
63 int main()
64 {
65     int item,choice,pos;
66     NODE first=NULL;
```

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

```
main.c
66 NODE first=NULL;
67
68 for(;;)
69 {
70     printf("\n 1:Insert_rear\n 2:Delete_front\n
71     3:Display_list\n 4:Exit\n");
72     printf("enter the choice\n");
73     scanf("%d",&choice);
74     switch(choice)
75     {
76     case 1:printf("enter the item at rear-end\n");
77             scanf("%d",&item);
78             first=insert_rear(first,item);
79             break;
80     case 2:first=delete_front(first);
81             break;
82     case 3:display(first);
83             break;
84     default:exit(0);
85     }
86 }
87 }
```

```
2
item deleted at front-end is-24
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is-25
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
list empty cannot display items
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
```

Screenshot of stacks:

```
main.c
1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node
4 {
5     int info;
6     struct node *link;
7 };
8 typedef struct node *NODE;
9 NODE getnode()
10 {
11     NODE x;
12     x=(NODE)malloc(sizeof(struct node));
13     if(x==NULL)
14     {
15         printf("mem full\n");
16         exit(0);
17     }
18     return x;
19 }
20 void freenode(NODE x)
21 {
22     free(x);
23 }
```

```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
12
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
13
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
```

The screenshot shows a Replit IDE window with the URL `repl.it/@PremaPrema/DeepskyblueRedundantTechnician#main.c`. The file `main.c` contains the following C code:

```

45 temp=temp->link;
46 printf("item deleted at front-end is=%d\n",
47 first->info);
48 free(first);
49 return temp;
50 }
51 void display(NODE first)
52 {
53     NODE temp;
54     if(first==NULL)
55     printf("stack empty cannot display items\n");
56     for(temp=first;temp!=NULL;temp=temp->link)
57     {
58         printf("%d\n",temp->info);
59     }
60 }
61 int main()
62 {
63     int item,choice,pos;
64     NODE first=NULL;
65     for(;;)

```

The terminal output on the right shows the program's execution:

```

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

```

The screenshot shows the same Replit IDE window, but with the completed C code in `main.c`:

```

64 for(;;)
65 {
66     printf("\n 1:Insert_front\n 2:Delete_front\n
67 3:Display_list\n 4:Exit\n");
68     printf("enter the choice\n");
69     scanf("%d",&choice);
70     switch(choice)
71     {
72     case 1:printf("enter the item at front-end\n");
73             scanf("%d",&item);
74             first=insert_front(first,item);
75             break;
76     case 2:first=delete_front(first);
77             break;
78     case 3:display(first);
79             break;
80     default:exit(0);
81     }
82 }
83 }
84 }
85 }

```

The terminal output on the right shows the program's execution after the code is completed:

```

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-at-end(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;
}
```

DATE \_\_\_\_\_  
PAGE \_\_\_\_\_

```
NODE delete-front(NODE first)
```

```
{
```

```
    NODE temp;
```

```
    if (first == NULL)
```

```
        printf("LE ");
```

```
        return first;
```

```
    }
```

```
    temp = first
```

```
    temp = temp->link;
```

```
    printf("Item deleted @ front end is- %d\n", temp->data);
```

```
    free(first);
```

```
    return temp;
```

```
}
```

```
void display(NODE first)
```

```
{ NODE temp;
```

```
  if (first == NULL)
```

```
      printf("LE ");
```

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

```
          printf("%d ", temp->data);
```

```
}
```

```
int main() {
```

```
    int item, choice, pos;
```

```
    NODE first = NULL;
```

```
    for (;;) {
```

```
        if
```

```
        printf("1: Insert rear\n2: Delete-front\n3: display\n4: exit\n5: exit\n");
```

```
        printf("enter the choice: ");
```

```
        switch (choice)
```

```
        {
```

```
            case
```



Date \_\_\_\_\_  
Page \_\_\_\_\_

```
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\t 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\lab0.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
2
enter the item at rear end
10

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
2
enter the item at rear end
20

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
2
enter the item at rear end
30

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
2
enter the item at rear end
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
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

```



```
D:\ds\lab0\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 freeNode(NODE x)
{
    free(x);
}

NODE insertFront(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);
    free(cur);
    return head;
}

```

```

NODE ddelete-rear(NODE head) {
    NODE cur, prev;
    if (head->rlink == head) {
        pf("dq empty\n");
        return head;
    }
}

```

```

    cur = head → llink;
    prev = cur → rlink;
    head → llink = prev;
    prev → rlink = head;
    pf ("the node deleted is %d", cur → info);
    free node (cur);
    return head;
}

```

```

NODE insert_testpos (int item, NODE head)
{

```

```

    NODE temp, cur, prev;
    if (head → rlink == head) {
        printf ("list empty\n");
        return head;
    }

```

```

    cur = head → rlink;
    while (cur != head)
    {

```

```

        printf ("key not found\n");
        return head;
    }

```

```

    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;

```

```

    pf (head → rlink == head)
    {

```

```

        pf ("%d", item);
    }

```



```

    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->link = head;
    head->link = head;
    for(;;)

```

```

    printf("1F 1T 2 IR 1T 3 DF 14 DR 15 insert left  

    pos 16 . delete specified value 17 : display  

    18 exit\n");

```

```

    switch(choice);

```

```

    { case 1: printf("enter the item @ front end\n");
      scanf("%d", &item);

```

```

      last = dinsert-front(item, head);

```

```

      break;

```

```

    case 2: printf("enter the item @ rear  

    end\n");

```

```

    scanf("%d", &item);

```

```

    last = dinsert-rear(item, head);

```

```

    break;

```

```

    case 3: last = ddelete from(head);

```

```

    break;

```

```

    case 4: last = ddelete rear(head); break;

```

```

    case 5: printf("key\n"); scanf("%d", &item);

```

```

    case 6: head = ddelete-specified-value(item, head);
    break;

```

```

    case 7: display(head);

```

```

    break;

```

```

    default: exit(0);

```

2

## 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);
```

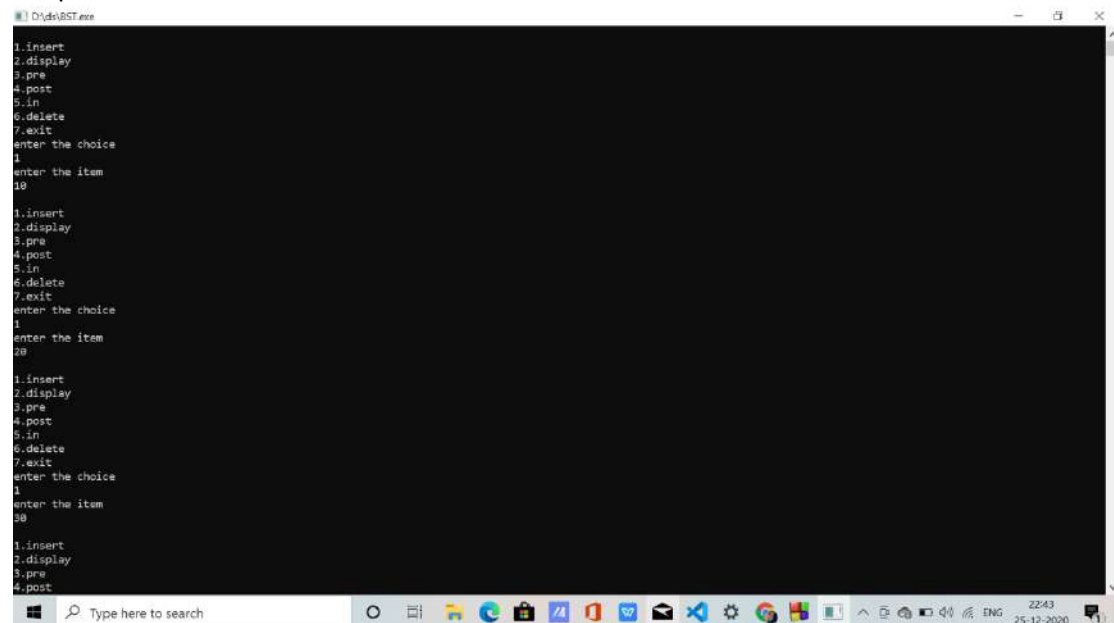
```
break;
```

```
}
```

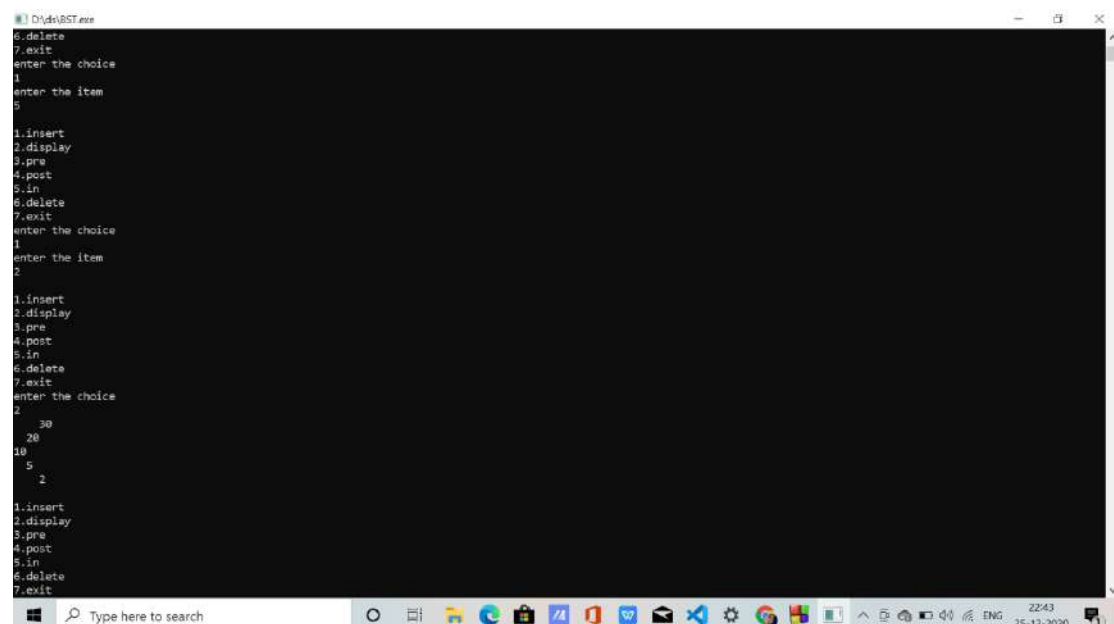
```
}
```

```
}
```

Output:-



```
D:\ds\BST.exe
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
10
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
```



```
D:\ds\BST.exe
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
enter the item
20
6.delete
7.exit
enter the choice
2
enter the item
5
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
enter the item
2
```

```
D:\ds\BST.exe
enter the choice
3
10
5
28
30

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
2
5
30
28
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
2
5
28
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
4

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
30
28
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)
```

```
  {
```

```
    printf("mem full\n");
```

```
  }
```

```
void freenode(NODE x)
```

```
{
```

```
  free(x);
```

```
}
```

```
NODE insert(NODE root, int item)
```

```
{
```

```
  NODE temp, curr, 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->link : cur->rlink;
}
if (temp item < prev->info)
    prev->link = 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->link, i+1);
    }
}

```

```

NODE delete (NODE root, int item)
{
    NODE cur, prev, q, r;
    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->link : cur->rlink;
}
if (cur == NULL)
    pf("not found\n");
return root;
}
if (cur->link == NULL)
    q = cur->rlink;
else if (cur->rlink == NULL)
    q = cur->link;
else
{
    suc = cur->rlink;
    while (suc->link != NULL)
        suc = suc->link;
    suc->link = cur->link;
    q = cur->rlink;
}
freemod(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(;;)
```

```

1
P1 ("1: insert 2: display 3: pre order 4: post
   5: in order 6: delete 7: exit\n");
P1 ("enter the choice\n");
if ("y.d", &choice);
switch (choice)
{
    case 1: printf ("enter the item\n");
             if ("y.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");
             if ("y.d", &item)
                 root = delete (root, item);
             break;
    default: exit (0);
             break;
}
}

```

void preorder(NODE root)

{ if (root == NULL)

{

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

preorder(root->link);

preorder(root->rlink);

}

}

void postorder(NODE root)

{

if (root == NULL)

{

postorder(root->link);

postorder(root->rlink);

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

}

}

void inorder(NODE root)

{

if (root != NULL)

{

inorder(root->link);

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

inorder(root->rlink);

}

}

int main()

{

int item, choice;

NODE root = NULL;

for(;;)



