

Date 28/9/2020

Expt. No. 2031MDC

Sneha Srivastava
(1BM19CS158)

Page No. 1

LAB PROGRAM-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 message for stack overflow, stack underflow.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int top = -1, stack[MAX];
void push();
void pop();
void display();

int main()
{
    int ch;
    while (1)
    {
        printf ("\n*** STACK MENU ***");
        printf ("\n1.PUSH\n2. POP\n3. DISPLAY\n4.EXIT");
        printf ("\nEnter YOUR CHOICE (1-4):");
        scanf ("%d", &ch);
    }
}
```

Teacher's Signature : _____

Date

Expt. No.....

Sneha Srivastava
(1BM19CS158)

Page No. 2

```

switch(ch)
{
    case 1: push();
    break;
    case 2: pop();
    break;
    case 3: display();
    break;
    case 4: exit(0);
    default: printf("In wrong choice");
}
return 0;
}

void push()
{
    int val;
    if (top == MAX - 1)
    {
        printf ("In STACK IS FULL");
    }
    else
    {
        printf ("In ENTER ELEMENTS TO PUSH:");
        scanf ("%d", &val);
        top = top + 1;
        stack [top] = val;
    }
}

```

Teacher's Signature : _____

Date.....

Expt. No.....

Sneha Srivastava
(1BM19CS158)

Page No..... 3

```
void pop()
{
    if (top == -1)
    {
        printf ("In STACK IS EMPTY");
    }
    else
    {
        printf ("In DELETED ELEMENT IS %d", stack [top]);
        top = top - 1;
    }
}

void display()
{
    int i;
    if (top == -1)
    {
        printf ("In STACK IS EMPTY");
    }
    else
    {
        printf ("In STACK IS ... ");
        for (i = top; i >= 0; --i)
            printf ("%d\n", stack [i]);
    }
}
```

Teacher's Signature : _____

OUTPUT SCREENSHOTS:

```
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc stack.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):2
```

```
STACK IS EMPTY
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):1
```

```
ENTER YOUR CHOICE(1-4):1
```

```
ENTER ELEMENTS TO PUSH:5
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):1
```

```
ENTER ELEMENTS TO PUSH:67
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):3
```

```
STACK IS...67
```

```
5
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):1
```

```
ENTER ELEMENTS TO PUSH:67
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):3
```

```
STACK IS...67
```

```
5
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):2
```

```
DELETED ELEMENT IS 67
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):3
```

```
STACK IS...5
```

```
***STACK MENU***
```

- 1.PUSH
- 2.POP
- 3.DISPLAY
- 4.EXIT

```
ENTER YOUR CHOICE(1-4):4
```

```
Snehas-MacBook-Pro:ccp snehasrivastava$ []
```

Date 5/10/2020

Expt. No.

Sneha Srivastava
(1BM19CS158)

Page No. 4

LAB PROGRAM-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).

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

```
int G (char symbol)
{
```

Teacher's Signature : _____

Date

Expt. No.

Sneha Srivastava
(1BM19 CS158)

Page No. 5

switch (symbol)

{

case '+':

case '-': return 1;

case '*':

case '/': return 3;

case '^':

case '\$': return 6;

case '(': return 9;

case ')': return 0;

default : return 7 ;

{}

3

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 - 1];

j++;

{

Teacher's Signature : _____

Date _____

Expt. No. _____

Sneha Srivastava
(1BM19CS158)

Page No. _____ 6

```
if (F(s [top]) != Gc (symbol))
```

```
    S [++top] = symbol;
```

```
else
```

```
    top --;
```

```
}
```

```
while (s [top] != '#')
```

```
{
```

```
    postfix [j++] = s [top--];
```

```
    postfix [j] = '10';
```

```
}
```

```
}
```

```
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);
```

```
    return 0;
```

```
}
```

Teacher's Signature : _____

OUTPUT SCREENSHOTS:

The screenshot shows a terminal window with tabs for TERMINAL, DEBUG CONSOLE, PROBLEMS, and OUTPUT. The terminal content is as follows:

```
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc postfix.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out
Enter the valid infix expression : (a+b)*(c-d)*(e/f)
The postfix expression is :
ab+cd-*ef/*
Snehas-MacBook-Pro:ccp snehasrivastava$ █
```

Date 19/10/2020

Expt. No.

Page No. 16

LINEAR QUEUE

```
#include <stdio.h>
#include <stdlib.h>
#define qsize 5
int item, front=0, rear=-1, q[10];
void insertrear()
{
    if (rear == qsize - 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()
```

Teacher's Signature : _____

Date.....

St. No.....

Page No. 17.....

```
{  
    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 ch;  
    for (;;)  
    {  
        printf ("1. insert-rear\n2. delete-front\n3. display\n4. exit\n");  
        printf ("enter choice\n");  
        scanf ("%d", &ch);  
        switch (ch)  
        {  
            Case 1: printf ("enter the item to be inserted\n");  
            scanf ("%d", &item);  
            insertrear ();  
            break;  
        }  
    }  
}
```

Teacher's Signature : _____

Date

Expt. No.

Page No. ... 18

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

return 0;
}
```

Teacher's Signature : _____

Scanned with CamScanner

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS 2 OUTPUT
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc linearq.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Queue is empty

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
1
enter the item to be inserted
12

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
1
enter the item to be inserted
56

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Item deleted=12

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Item deleted=56

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
3
Queue is empty

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
4
Snehas-MacBook-Pro:ccp snehasrivastava$ █
```

Date 19/10/2020

Expt. No.

Page No. 13

CIRCULAR QUEUE

```

#include <stdio.h>
#include <stdlib.h>
#define qsize 5
int item, f=0, r=-1, q[qsize], count=0, ch;
void insertrear()
{
    if (count==qsize)
    {
        printf("Queue overflow\n");
        return;
    }
    r=(r+1)%qsize;
    q[r]=item;
    count++;
}
int deletefront()
{
    if (count==0) return -1;
    item=q[f];
    f=(f+1)%qsize;
    count=count-1;
    return item;
}
void display()
{
}

```

Teacher's Signature : _____

Date.....

Expt. No.....

Page No... 14.....

```

int i, front;
if (count == 0)
{
    printf("Queue is empty\n");
    return;
}

front = 0;
printf("Contents of queue\n");
for (i = 1; i <= count; i++)
{
    printf("%d\n", q[front]);
    front = (front + 1) % q.size;
}

int main()
{
    for (;;)
    {
        printf("1.insert_rear\n2.delete_front\n3.display\n4.exit\n");
        printf("Enter choice\n");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1: printf("Enter the item to be inserted\n");
                      scanf("%d", &item);
                      insertrear();
                      break;
        }
    }
}

```

Teacher's Signature : _____

Date _____

Expt. No. _____

Page No. 15 _____

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

return 0;
```

Teacher's Signature : _____

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc circularq.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Queue is empty

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
1
enter the item to be inserted
47

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
1
enter the item to be inserted
93

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Item deleted=47

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
Item deleted=93

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
3
Queue is empty

1.insert_rear
2.delete_front
3.display
4.exit
enter choice
4
Snehas-MacBook-Pro:ccp snehasrivastava$ █
```

LAB PROGRAM:- SINGLE LINKED LIST

UNIVERSITY
Date _____
Page _____

```
#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 insertFront (NODE first, int item)
```

```
{
```

```
    NODE temp;
    temp = getNode();
    temp->info = item;
    temp->Link = NULL;
    if (first == NULL)
```

Date _____
 Page _____

```

getwen temp;
temp-> link = first;
first = temp;
getwen 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 delete-front(NODE first)

```

{
  NODE temp;
  if (first == NULL)
    present("List is empty cannot delete \n");
  getwen first;
  temp = first;
  temp = temp-> Link;
  print(" item deleted at front-end is=%d \n", first-> info);
  free(first);
  getwen temp;
}
  
```

NODE insert rear (NODE first, int item)

{

 NODE temp, cur;

 temp = getnode();

 temp->info = item;

 temp->Link = NULL;

 if (first == NULL)

 return temp;

 cur = first;

 while (cur->link != NULL)

 cur = cur->Link;

 cur->Link = temp;

 return first;

}

NODE 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-rear (NODE first)

{

 NODE cur, prev;

 if (first == NULL)

 printf ("List is empty cannot delete\n");

 return first;

}

 if (first->link == NULL)

 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 insert-pos (int item, int pos, NODE first)

{

 NODE temp;

 NODE prev, cur;

 int count;

 temp = getnode();

Page _____

```

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;
pPrev = NULL;
cur = first;
while (cur != NULL && count != pos)
{
    pPrev = cur;
    cur = cur->link;
    Count++;
}
if (count == pos)
{
    pPrev->link = temp;
    temp->link = cur;
    return first;
}
printf("invalid position\n");
return first;
}

```

Page

```
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;
freemode (cur);
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;
    }
}

```

Date _____
Page _____

```
else
{
    while ( prev == NULL )
    {
        cur = pprev->link;
        while ( cur != NULL )
        {
            if ( pprev->info > cur->info )
            {
                temp = pprev->info;
                pprev->info = cur->info;
                cur->info = temp;
            }
            cur = cur->link;
        }
        prev = pprev->link;
    }
    return first;
}
```

NODE des (NODE first)

```
{}
NODE pprev = first;
NODE cur = NULL;
int temp;
if ( first == NULL )
{
    return 0;
}
else
{}
```

Date _____
Page _____

```

while( pprev != NULL )
{
    cur = pprev->link;
    while( cur != NULL )
    {
        if( pprev->info < cur->info )
        {
            temp = pprev->info;
            pprev->info = cur->info;
            cur->info = temp;
        }
        cur = cur->link;
    }
    pprev = pprev->link;
}
return first;
}

```

```

NODE concate( 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;
}

```

```

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, option, choice2, item1, num;
    NODE first = NULL;
    NODE second = NULL;
    for (ii)
    {
        printf ("1. Insert-front\n 2: Delete-front\n 3: Insert-rear\n 4: Delete-rear\n 5. random-position\n 6: Reverse\n 7: Sort\n 8: concat\n 9. display-list\n 10. 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: printf ("Enter the item at rear-end\n");
        }
    }
}

```

CLASSMATE
Date _____
Page _____

```

scanf ("%d", &item);
first = insert_rear (first, item);
break;

case 4: first = delete_rear (first);
break;

case 5:
printf ("press 1 to insert or 2 to delete at any desired position\n");
scanf ("%d", &element);
if (element == 1)
{
    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);
}

if (element == -2)
{
    printf ("enter the position to delete\n");
    scanf ("%d", &pos);
    first = delete_pos (pos, first);
}

break;

case 6:
first = reverse (first);
break;

case 7:
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 = desc (first);
}

```

```

first = des(first);
break;

case 8:
printf ("Create a second list\n");
printf ("Enter the number of elements in second list\n");
scanf ("%d", &num);
for (int i=1; i<=num; i++)
{
    printf ("\nEnter 1 to insert front and 2 to insert rear\n");
    scanf ("%d", &choice2);
    if (choice2 == 1)
        second = IF (second, item1);
    else
        second = IR (second, item1);
}

if (choice2 == 2)
{
    printf ("Enter the item at rear end\n");
    scanf ("%d", &item1);
    second = IR (second, item1);
}

first = concat (first, second);
break;

case 9: display (first);
break;

default: exit (0);
}

return 0;
}

```

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concatenate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
4

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concatenate
9:display_list
10:Exit
enter the choice
3
enter the item at rear-end
7

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concatenate
9:display_list
10:Exit
enter the choice
5
press 1 to insert or 2 to delete at any desired position
1
enter the position to insert
2
enter the item to insert
34

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concatenate
9:display_list
10:Exit
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
_____
10:Exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
1

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8:concat
9:display_list
10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
2

press 1 to insert front and 2 to insert rear
1
enter the item at front-end
56

press 1 to insert front and 2 to insert rear
2
enter the item at rear-end
56

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8:concat
9:display_list
10:Exit
enter the choice
9
4
7
34
56
56

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8:concat
9:display_list
10:Exit
```

LAB 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_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;
}
```

```

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 length (NODE first)
{

```

```

    NODE cur;
    int count = 0;
    if (first == NULL)
        return 0;
    cur = first;
    while (cur != NULL)
    {
        count++;
        cur = cur->link;
    }
    return count;
}

```

```

void search ( int key, NODE first )
{

```

```

    NODE cur;
    if (first == NULL)

```

```

        printf ("List is empty\n");
}

```

```

return;
}
cur = first;
while (cur != NULL)
{
    if (key == cur->info)
        break;
    cur = cur->link;
}
if (cur == NULL)
    printf ("Search is unsuccessful\n");
    return;
}
printf ("Search successful\n");

```

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;

```

curr->info = temp;
}

curr = curr->link;
}

prev = prev->link;
}

return first;
}

int main()
{
    int item, choice, count, key, option;
    NODE first = NULL;
    for (;;)
    {
        printf("1: Insert-front\n 2: Delete-rear\n 3: Display-list\n 4:
              Count items\n 5: Search item\n 6: Order-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:
                first = delete_rear(first);
                break;
            case 3:
                display(first);
        }
    }
}

```

Date _____
Page _____

```
break;  
case 4:  
    count = length(first);  
    printf("Length of items in the list is %d\n", count);  
    break;  
case 5:  
    printf("enter the item to be searched\n");  
    scanf("%d", &key);  
    search(key, first);  
    break;  
case 6:  
    printf("In1-ascending ordered-list In2-descending ordered-list");  
    scanf("%d", &option);  
    if (option == 1)  
    {  
        first = asc(first);  
        display(first);  
    }  
    else  
    {  
        first = des(first);  
        display(first);  
    }  
    break;  
default:  
    exit(0);  
}  
return 0;  
}
```

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc ll.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1:Insert_front
2:Delete_rear
3:Display_list
4:Count_items
5:Search_items
6:Order_list
7:Exit
enter the choice
1
enter the item at front-end
12

1:Insert_front
2:Delete_rear
3:Display_list
4:Count_items
5:Search_items
6:Order_list
7:Exit
enter the choice
1
enter the item at front-end
23

1:Insert_front
2:Delete_rear
3:Display_list
4:Count_items
5:Search_items
6:Order_list
7:Exit
enter the choice
1
enter the item at front-end
56

1:Insert_front
2:Delete_rear
3:Display_list
4:Count_items
5:Search_items
6:Order_list
7:Exit
enter the choice
3
56
23
12

1:Insert_front
2:Delete_rear
3:Display_list
4:Count_items
5:Search_items
6:Order_list
7:Exit
enter the choice
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
_____
7:Exit
enter the choice
2
item deleted at rear-end is 12
1:Insert_front
2:Delete_rear
3:Display_list
4:Count items
5:Search items
6:Order_list
7:Exit
enter the choice
4
Length of items in the list is 2

1:Insert_front
2:Delete_rear
3:Display_list
4:Count items
5:Search items
6:Order_list
7:Exit
enter the choice
5
enter the item to be searched
12
Search is unsuccessful

1:Insert_front
2:Delete_rear
3:Display_list
4:Count items
5:Search items
6:Order_list
7:Exit
enter the choice
6

1:ascending_ordered_list
2:descending_ordered_list
1
23
56

1:Insert_front
2:Delete_rear
3:Display_list
4:Count items
5:Search items
6:Order_list
7:Exit
enter the choice
6

1:ascending_ordered_list
2:descending_ordered_list
2
56
23
```

```
1:Insert_front
2:Delete_rear
3:Display_list
4:Count items
5:Search items
6:Order_list
7:Exit
enter the choice
7
Snehas-MacBook-Pro:ccp snehasrivastava$ []
```

classmate

Date _____
Page _____

1BM19CS158

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

struct node
{
    int info;
    struct node *llink; // left link
    struct node *rlink; // right link
};

typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE)malloc(sizeof(struct node));
    if (x == NULL)
        {
            printf("Memory FULL\n");
            exit(0);
        }
    return x;
}

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

NODE 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 ("Doubly linked list empty\n");
```

```
    return head;
```

```
}
```

```
cur = head->rlink;
```

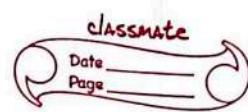
```
next = cur->rlink;
```

```
head->rlink = next;
```

```
next->llink = head;
```

```
printf ("In The node deleted is %d", cur->info);
```

```
freenode (cur);
```



```
return head;  
}  
  
NODE delete_rear(NODE head)  
{  
    NODE cur, prev;  
    if (head->rlink == head)  
    {  
        printf("Doubly linked list 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;  
}  
  
void display(NODE head)  
{  
    NODE temp;  
    if (head->rlink == head)  
    {  
        printf("Doubly linked list empty\n");  
        return;  
    }  
  
    printf("Contents of the doubly linked list\n");  
    temp = head->rlink;  
    while (temp != head)  
    {  
        printf("%d ", temp->info);  
        temp = temp->rlink;  
    }  
}
```

CLASSMATE

Date _____
Page _____

```
    temp = temp->rlink;
```

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

```
NODE delete_all_key (int item, NODE head)
```

```
{ NODE prev, cur, next;
```

```
    int count;
```

```
    if (head->rlink == head)
```

```
        printf ("Doubly linked list empty \n");
```

```
        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;
```

```
            free node (cur);
```

```
            cur = next;
```

```
}
```

```
}
```

```
if (count == 0)
```

```
    printf ("Key not found \n");
```

classmate
Date _____
Page _____

```

else
printf ("Key found at %d positions and are deleted\n", count);
return head;
}

NODE insert_leftpos (int item, NODE head)
{
    NODE temp, cur, prev;
    if (head->rlink == head)
        printf ("Doubly Linked 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 ("In 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 insert_right_pos (int item, NODE head)

 NODE temp, cur, prev;
 if (head->rlink == head)

 printf ("Doubly linked List empty\n");
 return head;

{

 cur = head->llink;

 while (cur != head)

{

 if (item == cur->info) break;

 cur = cur->llink;

}

 if (cur == head)

{

 printf ("Key not found\n");

 return head;

}

 prev = cur->rlink;

 printf ("\nEnter towards right of %d = ", item);

 temp = getnode();

 scanf ("%d", &temp->info);

 prev->llink = temp;

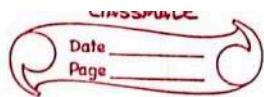
 temp->rlink = prev;

 cur->rlink = temp;

 temp->llink = cur;

 return head;

{



Void search (NODE head)

{

```
struct node *ptr;
int item, i=0, flag;
ptr = head;
if (ptr == NULL)
```

```
} printf ("Doubly linked list empty\n");
```

```
else
```

{

```
printf ("Enter item which you want to search ?\n");
scanf ("%d", &item);
while (ptr != NULL)
```

{

```
if (ptr->info == item)
```

```
printf ("Item found at location %d", i+1);
```

```
flag = 0;
```

```
break;
```

}

```
else
```

{

```
flag = 1;
```

```
printf ("Item not found\n");
```

```
break;
```

}

}

]

}

```
int main()
```

{

classmate

Date _____
Page _____

```
NODE head, last;
int item, choice, choice1;
head = getnode();
head->rlink = head;
head->llink = head;

for(;;)

printf("1: insert front\n2: insert rear\n3: delete front\n"
       "4: delete rear\n5: display\n6: delete all occurrences\n"
       "7: simple search\n8: insert node before & after the
       key node\n9: exit\n");
printf("enter the choice\n");
scanf("%d", &choice);
switch(choice)
{
    case 1: printf("enter the item at front\n");
              scanf("%d", &item);
              last = dinsert_front(item, head);
              break;
    case 2: printf("enter the item at rear\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: display(head);
              break;
    case 6: printf("enter element to be deleted\n");
              scanf("%d", &item);
```

CLASSMATE
Date _____
Page _____

```
delete_all_key(item, head);
break;
case 7: search(head);
break;
Case 8:
printf("In 1. insert towards left In 2. insert towards right\n");
printf("enter the choice\n");
scanf("%d", &choice1);
if (choice1 == 1)
{
    printf("enter the key element\n");
    scanf("%d", &item);
    last = insert_leftpos(item, head);
    break;
}
else
{
    printf("enter the key element\n");
    scanf("%d", &item);
    last = insert_rightpos(item, head);
    break;
}
default: exit(0);
}
return 0;
```

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
1
enter the item at front end
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
1
enter the item at front end
78

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
2
enter the item at rear end
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
2
enter the item at rear end
56
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
5
Contents of the doubly linked list
78      56      56      56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
6
enter element to be deleted
5
Key not found

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
Snehas-MacBook-Pro:ccp snehasrivastava$ 
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc dll.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
1
enter the item at front end
23

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
2
enter the item at rear end
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
2
enter the item at rear end
89

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
5
Contents of the doubly linked list
56      12      56      89

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
8

1:insert towards left
2:insert towards right
enter the choice
1
enter the key element
12
```

```
.. TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
9:exit
enter the choice
2
enter the item at rear end
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
8

1:insert towards left
2:insert towards right
enter the choice
2
enter the key element
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc dll.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
1
enter the item at front end
12

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
1
enter the item at front end
56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurences
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

enter the key element
12

Enter towards left of 12=56

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
5

Contents of the doubly linked list
56 56 12 56 89

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
6
enter element to be deleted
56

Key found at 3 positions and are deleted

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
5

Contents of the doubly linked list

12 89

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:delete all occurrences
7:simple search
8:insert node before & after the key node
9:exit
enter the choice
9

Snehas-MacBook-Pro:ccp snehasrivastava\$ █

CLASSMATE
Date 21/12/2020
Page _____

1BM19CS158

BINARY TREE

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

struct node
{
    int info;
    struct node *llink;
    struct node *rlink;
};

typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if (x == NULL)
        printf ("Memory not available!");
    exit(0);
}

return x;
}

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

NODE insert (int item, NODE root)
{
    NODE temp, cur, prev;
    char direction [10];
    int i;
```

classmate
Date _____
Page _____

```
temp = getnode();
temp->info = item;
temp->lLink = NULL;
temp->rLink = NULL;
if (root == NULL)
    return temp;
printf ("Give direction to insert.. \n");
scanf ("%s", direction);
prev = NULL;
cur = root;
for (i=0; i<strlen (direction) && cur != NULL; i++)
{
    prev = cur;
    if (direction[i] == 'L')
        cur = cur->lLink;
    else
        cur = cur->rLink;
}
if (cur == NULL || i == strlen (direction))
{
    printf ("Insertion not possible \n");
    freeNode (temp);
    return (root);
}
if (cur == NULL)
    if (direction[i-1] == 'L')
        prev->lLink = temp;
    else
        prev->rLink = temp;
return (root);
}
```

Date _____
Page _____

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

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

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

```
void display (NODE root, int i)
{
    int j;
    if (root != NULL)
```

classmate

Date _____

Page _____

```

display (root->link, i+1);
for (j= 1; j <= i; j++)
print (" ");
printf ("%d\n", root->info);
display (root->link, i+1);
}
}

int main ()
{
NODE root = NULL;
int choice, i, item;
for(;;)
printf ("1. Insert \n 2. Preorder \n 3. Inorder \n 4. Postorder \n 5. Display \n");
printf ("Enter the choice : \n");
scanf ("%d", &choice);
switch (choice)
{
case 1 : printf ("Enter the item : \n");
scanf ("%d", &item);
root = insert (item, root);
break;
case 2 : if (root == NULL)
{
printf ("Tree is empty !");
}
else
{
printf ("Given tree is . . . \n");
display (root, 1);
printf ("The preorder traversal is : \n");
preorder (root);
}
break;
}
}

```

```
case 3: if (root==NULL)
{
    printf ("Tree is empty!");
}
else
{
    printf ("Given tree is.. \n");
    display (root, 1);
    printf ("The inorder traversal is: \n");
    inorder (root);
}
break;

case 4: if (root==NULL)
{
    printf ("Tree is empty!");
}
else
{
    printf ("Given tree is.. \n");
    display (root, 1);
    printf ("The postorder traversal is: \n");
    postorder (root);
}
break;

case 5: display (root, 1);
break;

default: printf ("Invalid choice entered.\n");
exit (0);
}

return 0;
}
```

OUTPUT SCREENSHOTS:

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
5.Display
Enter the choice:
5
    6
    78
        3
    23
        56
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
2
Given tree is..
    6
    78
        3
    23
        56
The preorder traversal is:
the item is 23
the item is 56
the item is 78
the item is 3
the item is 6
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
3
Given tree is..
    6
    78
        3
    23
        56
The inorder traversal is
The item is 56
The item is 23
The item is 3
The item is 78
The item is 6
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
4
Given tree is..
    6
    78
        3
    23
        56
The postorder traversal is
```

```
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Snehas-MacBook-Pro:ccp snehasrivastava$ gcc BT.c
Snehas-MacBook-Pro:ccp snehasrivastava$ ./a.out
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
23
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
56
Give direction to insert..
l
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
78
Give direction to insert..
r
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
3
Give direction to insert..
rl
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
6
Give direction to insert..
rr
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
```

56

The postorder traversal is

The item is 56

The item is 3

The item is 6

The item is 78

The item is 23

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice: