

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



DATA STRUCTURE LAB RECORD

Submitted by

DEEPTHI L (1BM19ET014)

Under the Guidance of

**Prof. SHEETAL VA
Assistant Professor, BMSCE**

*in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING*



**B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2020 to Jan-2021**

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the LAB RECORD carried out by **XYZ (1BM19CS000)** who is the bonafide students of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide Signature of the HOD Prof. Prof. Sheelal VA Dr. Umadevi V Assistant Professor Associate Prof.& Head, Dept. of CSE BMSCE, Bengaluru BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

F Laboratory Plan (if applicable)**INDEX**

Lab Program	Unit #	Program Details
1	1	<p>Write a program to simulate the working of stack using an array with the following :</p> <p>a) Push b) Pop c) Display</p> <p>The program should print appropriate messages for stack overflow, stack underflow</p>
2	1	<p>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)</p>
3	2	<p>WAP to simulate the working of a queue of integers using an array. Provide the following operations</p> <p>a) Insert b) Delete c) Display</p> <p>The program should print appropriate messages for queue empty and queue overflow conditions</p>
4	2	<p>WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.</p> <p>a) Insert b) Delete c) Display</p> <p>The program should print appropriate messages for queue empty and queue overflow conditions</p>
5	3	<p>WAP to Implement Singly Linked List with following operations</p> <p>a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.</p>
6	3	<p>WAP to Implement Singly Linked List with following operations</p> <p>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.</p>
7	3	<p>WAP Implement Single Link List with following operations</p> <p>a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists</p>
8	3	<p>WAP to implement Stack & Queues using Linked Representation</p>
9	4	<p>WAP Implement doubly link list with primitive operations</p> <p>a) Create a doubly linked list. b) Insert a new node to the left of the node.</p> <p>c) Delete the node based on a specific value. c) Display the contents of the list</p>
10	5	<p>Write a program</p> <p>a) To construct a binary Search tree.</p> <p>b) To traverse the tree using all the methods i.e., in-order, preorder and post order</p> <p>c) To display the elements in the tree</p>

1) wap to accept student details and print

```
1 #include<stdio.h>
2 struct student
3 {
4     int id,age,marks;
5 };
6 int main()
7 {
8     struct student data[100];
9     int n;
10    printf("enter the number of students:\n");
11    scanf("%d",&n);
12    int i;
13    for(i=0;i<n;i++)
14    {
15        printf("enter the data of the student %d \n",i+1);
16        printf("id: ");
17        scanf("%d",&data[i].id);
18        printf("enter the age of the student %d \n",i+1);
19        printf("age: ");
20        scanf("%d",&data[i].age);
21        printf("enter marks of the student %d \n",i+1);
22        printf("marks: ");
23        scanf("%d",&data[i].marks);
24    }
25    printf("the students who are qualified for admission are:\n");
26    for(i=0;i<n;i++)
27    {
28        if(data[i].age>=20 && data[i].marks<=100)
29            printf("student id:%d age:%d marks:%d\n",data[i].id,data[i].age,data[i].marks);
30    }
31    return 0;
32 }
```

```
enter the number of students:
2
enter the data of the student 1
id: 101
enter the age of the student 1
age: 25
enter marks of the student 1
marks: 80
enter the data of the student 2
id: 102
enter the age of the student 2
age: 19
enter marks of the student 2
marks: 70
the students who are qualified for admission are:
student id:101 age:25 marks:80

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

wap to implement stack using push and pop functions

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 # define size 5
4 int top=-1;
5 void push(int stack[],int item)
6 {
7     if(top==size-1)
8         printf("STACK OVERFLOW \n");
9     else
10    {
11        top++;
12        stack[top]=item;
13    }
14 }
15 int pop(int stack[])
16 {
17     if(top==-1){
18         printf("STACK UNDERFLOW\n");
19         return -1;
20     }
21     else
22    {
23        int item=stack[top];
24        top--;
25        return item;
26    }
27 }
28 void display(int stack[])
29 {
30 }
```

```
29 void display(int stack[])
30 {
31     printf("contents of the stack \n");
32     for(int i=top;i>=0;i--)
33         printf("%d\n",stack[i]);
34 }
35 int main()
36 {
37     int stack[size];
38     int choice,element;
39     for(;;)
40     {
41         printf("\n1.push\n2.pop\n3.print\n4.exit\n");
42         printf("enter your choice\n");
43         scanf("%d",&choice);
44         switch(choice){
45             case 1:
46                 printf("enter element\n");
47                 scanf("%d",&element);
48                 push(stack,element);
49                 break;
50             case 2:
51                 element=pop(stack);
52                 if(element== -1)
53                     printf("STACK UNDERFLOW\n");
54                 else
55                     printf("element popped :%d",element);
56                 break;
57             case 3:
58                 display(stack);
59 }
```

```
59         display(stack);
60         break;
61     default :
62         printf("end of operation\n");
63         exit(0);
64     }
65 }
```

```
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
1  
enter element  
1  
  
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
1  
enter element  
2  
  
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
1  
enter element  
3  
  
1.push
```

```
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
1  
enter element  
5  
  
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
3  
contents of the stack  
5  
4  
3  
2  
1  
  
1.push  
2.pop  
3.print  
4.exit  
enter your choice  
2  
element popped :5
```

```
4.exit
enter your choice
2
element popped :5
1.push
2.pop
3.print
4.exit
enter your choice
2
element popped :4
1.push
2.pop
3.print
4.exit
enter your choice
2
element popped :3
1.push
2.pop
3.print
4.exit
enter your choice
3
contents of the stack
2
1

1.push
```

```
contents of the stack
2
1

1.push
2.pop
3.print
4.exit
enter your choice
4
end of operation

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

wap to convert infix to postfix

```
1 #include<stdio.h>
2 #include<string.h>
3 int F(char symbol)
4 {
5     switch(symbol)
6     {
7         case '+':
8             case '-':return 2;
9             case '*':
10            case '/':return 4;
11            case '^':
12            case '$':return 5;
13            case '(':return 0;
14            case ')':return -1;
15            default:return 8;
16     }
17 }
18 int G(char symbol)
19 {
20     switch(symbol)
21     {
22         case '+':
23             case '-':return 1;
24             case '*':
25             case '/':return 3;
26             case '^':
27             case '$':return 6;
28             case '(':return 9;
29             case ')':return 0;
30     }
31     default:return 7;
32 }
33 }
34 void infix_postfix(char infix[],char postfix[])
35 {
36     int top,i,j;
37     char s[30],symbol;
38     top=-1;
39     s[++top]='#';
40     j=0;
41     for(i=0;i<strlen(infix);i++)
42     {
43         symbol=infix[i];
44         while(F(s[top])>G(symbol))
45         {
46             postfix[j]=s[top--];
47             j++;
48         }
49         if(F(s[top])!=G(symbol))
50             s[++top]=symbol;
51         else
52             top--;
53         }
54         while(s[top]!='#')
55         {
56             postfix[j++]=s[top--];
57         }
58         postfix[j]='\0';
59     }
60 }
```

```
void main()
```

```
60     void main()
61     {
62         char infix[20];
63         char postfix[20];
64         printf("enter valid infix expression:\n");
65         scanf("%s",infix);
66         infix_postfix(infix,postfix);
67         printf("the postfix expression is:\n");
68         printf("%s\n",postfix);
69     }
70
71
```

```
enter valid infix expression:
a$b*c-d+e/f/(g+h)
the postfix expression is:
ab$c*d-e f/g h + / +

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

wap to check if a word is palindrome or not

```
1 #include<stdio.h>
2 #include<string.h>
3 int top=-1;
4 char stack[30];
5 void push(char b)
6 {
7     top++;
8     stack[top]=b;
9 }
10 char pop(char word[])
11 {
12     return word[top--];
13 }
14 }
15 int main()
16 {
17     char word[30];
18     char newword[30];
19     int flag=0;
20     printf("enter the string \n");
21     scanf("%s",word);
22     int size=strlen(word);
23     for(int i=0;i<size;i++)
24         push(word[i]);
25     for(int i=0;i<size;i++)
26         newword[i]=pop(word);
27     for(int i=0;i<size;i++)
28     {
29         if(stack[i]==newword[i])
30             continue;
31         else{
32             flag=1;
33             break;
34         }
35     }
36     if(flag==1)
37         printf("NON PALINDROME \n");
38     else
39         printf("PALINDROME \n");
40 }
```

```
input
enter the string
madam
PALINDROME

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

Tower of hanoi:

```
1 #include<stdio.h>
2 void tower(int,char,char,char);
3 void tower(int n,char src,char temp,char dest)
4 {
5     if(n==1)
6     {
7         printf("move disc %d from %c to %c \n",n,src,dest);
8         return;
9     }
10    tower(n-1,src,dest,temp);
11    printf("move disc %d from %c to %c \n",n,src,dest);
12    tower(n-1,temp,src,dest);
13    return;
14 }
15 int main()
16 {
17     int element;
18     printf("enter the number of discs: \n");
19     scanf("%d",&element);
20     tower(element,'s','t','d');
21
22 }
```

```
enter the number of discs:
4
move disc 1 from s to t
move disc 2 from s to d
move disc 1 from t to d
move disc 3 from s to t
move disc 1 from d to s
move disc 2 from d to t
move disc 1 from s to t
move disc 4 from s to d
move disc 1 from t to d
move disc 2 from t to s
move disc 1 from d to s
move disc 3 from t to d
move disc 1 from s to t
move disc 2 from s to d
move disc 1 from t to d

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

Fibonacci series :

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

```
int fib ( int n )
```

```
{ if ( n == 0 )
```

```
    return 0 ;
```

```
    if ( n == 1 )
```

```
        return 1 ;
```

```
    return fib ( n - 1 ) + fib ( n - 2 ) ;
```

```
 }
```

```
int main ( )
```

```
{
```

```
    int i , n ;
```

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

```
    scanf ( "% . d " , & n ) ;
```

```
    printf ( " . d fibonacci numbers are : " ) ;
```

```
    for ( i = 0 ; i < n / i + 1 ) {
```

```
        printf ( " fib ( . d ) = . d ( n ) , i is fib ( i ) " ) ;
```

```
}
```

Enter n

9

9 fibonacci numbers are:

$\text{fib}(0)=0$

$\text{fib}(1)=1$

$\text{fib}(2)=1$

$\text{fib}(3)=2$

$\text{fib}(4)=3$

$\text{fib}(5)=5$

$\text{fib}(6)=8$

$\text{fib}(7)=13$

$\text{fib}(8)=21$

[Program finished] █

Factorial of n numbers:

#include <stdio.h>

int fact (int n)

{ if (n == 0)

 return 1;

 return n * fact (n - 1);

}

int main()

{ int n;

 printf ("enter the value of n\n");

 scanf ("%d", &n);

 printf ("The factorial of %d = %d\n",

 n, fact (n));

}

GCD of 2 nos

#include <stdio.h>

int gcd (int m, int n)

{ if (n == 0) return m;

 if (m < n) return gcd (n, m);

 return gcd (n, m % n);

}

int main()

{ int m, n, res;

 printf ("enter m & n\n");

 scanf ("%d %d", &m, &n);

 res = gcd (m, n);

 printf ("gcd (%d, %d) = %d\n", m, n, res);

}

Enter n

5

The factorial of 5=120

[Program finished]

Enter m and n

4 3

gcd(4,3)=1

[Program finished] █

Binary search using recursion:

```
#include <stdio.h>
void binary_search (int[], int, int, int);
void bubble_sort (int[], int);
int main ()
{
    int key, size, i;
    int list[25];
    printf ("Enter size of a list : ");
    scanf ("%d", &size);
    printf ("Enter elements in ");
    for (i = 0; i < size; i++)
        scanf ("%d", &list[i]);
    bubble_sort (list, size);
    printf ("\n");
    printf ("Enter key to search in ");
    scanf ("%d", &key);
    binary_search (list, 0, size, key);
}
```

```
void bubble_sort (int list[], int size)
{
```

```
    int temp, i, j;
    for (i = 0; i < size; i++)

```

```
        for (j = 1; j < size; j++)

```

```
            if (list[i] > list[j])

```

```
                temp = list[i];

```

```
list[i] = list[j];  
list[j] = temp;  
y  
y  
y
```

```
void binary_search (int list[], int lo, int hi,  
                    int key)  
{
```

```
    int mid;
```

```
    if ((lo > hi))
```

```
        printf ("key not found (%d)\n");
```

```
        return;
```

```
    y
```

```
    mid = (lo + hi) / 2;
```

```
    if (list[mid] == key)
```

```
    {
```

```
        printf ("key found (%d)\n");
```

```
        else if (list[mid] > key)
```

```
    {
```

```
        binary_search (list, lo, mid - 1, key);
```

```
    } else if (list[mid] < key)
```

```
    { binary_search (list, mid + 1, hi, key);
```

```
    }
```

Enter size of a list: 4

Enter elements

1 2 3 4

Enter key to search

3

Key found

[Program finished]

1)wap to implement linear queue

2)wap to implement circular queue

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

23 void displayQ()
24 {
25     int i;
26     if(front>rear)
27     printf("queue empty\n");
28     else
29     for(i=front;i<=rear;i++)
30     {
31         printf("%d\n",q[i]);
32     }
33 }
34 void main()
35 {
36     int choice;
37
38     for(;;)
39     {
40         printf("\n1.insertrear\n2.deletefront\n3.display\n4.exit\n");
41         printf("enter the choice\n");
42         scanf("%d",&choice);
43         switch(choice)
44         {
45             case 1: printf("enter the item to be inserted\n");
46             scanf("%d",&item);
47             insert_rear();
48             break;
49             case 2:
50                 item=delete_front();
51                 if(item==-1)

52                     item=delete_front();
53                     if(item==-1)
54                         printf("empty queue\n");
55                     else
56                         printf("element deleted: %d",item);
57                         break;
58             case 3:
59                 displayQ();
56                 break;
60             default:
61                 printf("end of operation\n");
62
63         }
64     }
65 }
66 }
67 }
```

```
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  
30  
  
1.insertrear  
2.deletefront  
2 display
```

```
enter the choice  
3  
0  
10  
20  
30  
40  
50  
  
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
2  
element deleted: 0  
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
2  
element deleted: 10  
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
2  
element deleted: 20  
1.insertrear
```

```
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
2  
element deleted: 20  
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
3  
30  
40  
50  
  
1.insertrear  
2.deletefront  
3.display  
4.exit  
enter the choice  
4  
end of operation  
  
1.insertrear  
2.deletefront  
3.display  
4.exit
```

```
1 #include<stdio.h>  
2 #include<stdlib.h>  
3 # define q_size 5  
4 int item,front=0,rear=-1,q[q_size],count=0;  
5 void insert_rear()  
6 {  
7     if(count==q_size)  
8     {  
9         printf("queue underflow\n");  
10        return;  
11    }  
12    rear=(rear+1)%q_size;  
13    q[rear]=item;  
14    count++;  
15 }  
16 int delete_front()  
17 {  
18     if(count==0)  
19     return -1;  
20     item=q[front];  
21     front=(front+1)%q_size;  
22     count=count-1;  
23     return item;  
24 }  
25 void displayQ()  
26 {  
27     int i,f;  
28     if(count==0)  
29     {  
30         printf("queue is empty\n");  
31         return;  
32     }
```

```

32     }
33     f=front;
34     printf("contents of queue\n");
35     for(i=1;i<=count;i++)
36     {
37         printf("%d\n",q[f]);
38         f=(f+1)%q_size;
39     }
40 }
41 void main()
42 {
43     int choice;
44
45     for(;;)
46     {
47         printf("\n1.insertrear\n2.deletefront\n3.display\n4.exit\n");
48         scanf("%d",&choice);
49         switch(choice)
50         {
51             case 1: printf("enter the item to be inserted\n");
52             scanf("%d",&item);
53             insert_rear();
54             break;
55             case 2: item=delete_front();
56             if(item==-1)
57                 printf("\n empty queue");
58             else
59                 printf("item deleted=%d\n",item);
60             break;
61             case 3:
62                 displayQ();
63
64             case 3:
65                 displayQ();
66                 break;
67             default:
68                 printf("end of operation");
69                 exit(0);
70         }
71     }
72 }
73
74

```

```

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

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

1.insertrear
2.deletefront
3.display
4.exit
1
enter the item to be inserted
5

```

```
enter the item to be inserted
5

1.insertrear
2.deletefront
3.display
4.exit
1
enter the item to be inserted
7

1.insertrear
2.deletefront
3.display
4.exit
1
enter the item to be inserted
9

1.insertrear
2.deletefront
```

```
1.insertrear
2.deletefront
3.display
4.exit
3
contents of queue
1
3
5
7
9

1.insertrear
2.deletefront
3.display
4.exit
2
item deleted=1

1.insertrear
2.deletefront
3.display
```

```
2.deletefront
3.display
4.exit
2
item deleted=3

1.insertrear
2.deletefront
3.display
4.exit
3
contents of queue
5
7
9

1.insertrear
2.deletefront
3.display
4.exit
4
enf of operation
```

Multiple priority queue:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<conio.h>
4 #define N 3
5 void pqinsert(int);
6 void pqdelete();
7 void display();
8 int queue[3][N];
9 int front[3]={0,0,0};
10 int rear[3]={-1,-1,-1};
11 int item,pr;
12 int main()
13 {
14     int ch;
15     while(1)
16     {
17         printf("\nPRIORITY QUEUE\n");
18         printf("*****\n");
19         printf("\n\t1:PQinsert\n");
20         printf("\n\t2:PQdelete\n");
21         printf("\n\t3:PQdisplay\n");
22         printf("\n\t4:Exit\n");
23         printf("\nEnter the choice\n");
24         scanf("%d",&ch);
25         switch(ch)
26         {
27             case 1:printf("\nEnter the priority number\n");
28                 scanf("%d",&pr);
29                 if(pr>0 && pr<4)
30                     pqinsert(pr-1);
31                 else
32                     printf("\n only 3 priority exists 1 2 3\n");
33                     break;
34             case 2:pqdelete();
35                 break;
36             case 3:display();
37                 break;
38             case 4:exit(0);
39         }
40     }
41     getch();
42 }
43 void pqinsert(int pr)
44 {
45     if(rear[pr]==N-1)
46         printf("\n Queue overflow\n");
47     else
48     {
49         printf("\nEnter the item\n");
50         scanf("%d",&item);
51         rear[pr]++;
52         queue[pr][rear[pr]]=item;
53     }
54 }
55 void pqdelete()
56 {
57     int i;
58     for(i=0;i<3;i++)
59     {
60         if(front[i]==-1)
61             continue;
62         printf("\nItem deleted from queue is %d",queue[i][front[i]]);
63         front[i]++;
64     }
65 }
```

```

58 int i;
59 for(i=0;i<3;i++)
60 {
61     if(rear[i]==front[i]-1)
62     printf("\n queue empty\n");
63     else
64     {
65         printf("deleted item is %d of queue %d\n",queue[i][front[i]],i+1);
66         front[i]++;
67     }
68 }
69 }
70 }
71 void display()
72 {
73     int i,j;
74     for(i=0;i<3;i++)
75     {
76         if(rear[i]==front[i]-1)
77             printf("\n queue empty %d\n",i+1);
78         else
79         {
80             printf("\nQUEUE %d:",i+1);
81             for(j=front[i];j<=rear[i];j++)
82                 printf("%d\t",queue[i][j]);
83         }
84     }
85 }
86 }

```

```

PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay
4:Exit

enter the choice
1

enter the priority number
2

enter the item
6

PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay

```

```
3:PQdisplay
4:Exit
enter the choice
1
enter the priority number
3
enter the item
8
PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay
4:Exit
enter the choice
1
enter the priority number
1
```

```
enter the priority number
1
enter the item
4
PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay
4:Exit
enter the choice
3
QUEUE 1:4
QUEUE 2:6
QUEUE 3:8
PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
```

```
2:PQdelete
3:PQdisplay
4:Exit
enter the choice
2
deleted item is 4 of queue 1
deleted item is 6 of queue 2
deleted item is 8 of queue 3

PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay
4:Exit
enter the choice
3
queue empty 1
queue empty 2
```

```
queue empty 2
queue empty 3

PRIORITY QUEUE
*****
1:PQinsert
2:PQdelete
3:PQdisplay
4:Exit
enter the choice
4

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

PRIORITY QUEUE:

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 struct node
4 {
5     int priority;
6     int info;
7     struct node *link;
8 }*front=NULL;
9 void insert(int item, int item_priority);
10 int del();
11 void display();
12 int isEmpty();
13 int main()
14 {
15     int choice,item,item_priority;
16     while(1)
17     {
18         printf("\n1.Insert\n");
19         printf("2.Delete\n");
20         printf("3.Display\n");
21         printf("4.Quit\n");
22         printf("\nEnter your choice : ");
23         scanf("%d", &choice);
24         switch(choice)
25         {
26             case 1:
27                 printf("\nEnter the item to be added in the queue : ");
28                 scanf("%d",&item);
29                 printf("\nEnter its priority : ");
30                 scanf("%d",&item_priority);
31                 insert(item, item_priority);
32             break;
33         }
34     }
35     return 0;
36 }
37 void insert(int item,int item_priority)
38 {
39     struct node *tmp,*p;
40     tmp=(struct node *)malloc(sizeof(struct node));
41     if(tmp==NULL)
42     {
43         printf("\nMemory not available\n");
44         return;
45     }
46     tmp->info=item;
47     tmp->priority=item_priority;
48     if( isEmpty() || item_priority < front->priority )
49     {
50         front=tmp;
51     }
52     else
53     {
54         p=front;
55         while(p->link!=NULL)
56         {
57             p=p->link;
58         }
59         p->link=tmp;
60     }
61 }
```

```

31         insert(item, item_priority);
32     break;
33     case 2:
34         printf("\nDeleted item is %d\n",del());
35         break;
36     case 3:
37         display();
38         break;
39     case 4:
40         exit(1);
41     default :
42         printf("\nWrong choice\n");
43     }
44 }
45
46 return 0;
47 }
48 void insert(int item,int item_priority)
49 {
50     struct node *tmp,*p;
51     tmp=(struct node *)malloc(sizeof(struct node));
52     if(tmp==NULL)
53     {
54         printf("\nMemory not available\n");
55         return;
56     }
57     tmp->info=item;
58     tmp->priority=item_priority;
59     if( isEmpty() || item_priority < front->priority )
60     {
61         front=tmp;
62     }
63     else
64     {
65         p=front;
66         while(p->link!=NULL)
67         {
68             p=p->link;
69         }
70         p->link=tmp;
71     }
72 }
```

```

61     if( isEmpty() || item_priority < front->priority )
62     {
63         tmp->link=front;
64         front=tmp;
65     }
66     else
67     {
68         p = front;
69         while( p->link!=NULL && p->link->priority<=item_priority )
70             p=p->link;
71         tmp->link=p->link;
72         p->link=tmp;
73     }
74 }
75 int del()
76 {
77     struct node *tmp;
78     int item;
79     if( isEmpty() )
80     {
81         printf("\nQueue Underflow\n");
82         exit(1);
83     }
84     else
85     {
86         tmp=front;
87         item=tmp->info;
88         front=front->link;
89         free(tmp);
90     }
91     return item;
92 }
```

```

93 int isEmpty()
94 {
95     if( front == NULL )
96         return 1;
97     else
98         return 0;
99 }
100 void display()
101 {
102     struct node *ptr;
103     ptr=front;
104     if( isEmpty() )
105         printf("\nQueue is empty\n");
106     else
107     {
108         printf("\nQueue is :\n");
109         printf("\nPriority      Item\n");
110         while(ptr!=NULL)
111         {
112             printf("%5d      %5d\n",ptr->priority,ptr->info);
113             ptr=ptr->link;
114         }
115     }
116 }
```

```
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 1  
  
Input the item to be added in the queue : 2  
  
Enter its priority : 3  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 1  
  
Input the item to be added in the queue : 5  
  
Enter its priority : 2  
  
1.Insert  
2.Delete  
3.Display  
4.Quit
```

```
4.Quit  
  
Enter your choice : 1  
  
Input the item to be added in the queue : 4  
  
Enter its priority : 2  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 1  
  
Input the item to be added in the queue : 3  
  
Enter its priority : 3  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 3  
  
Queue is :
```

```
Queue is :  
Priority      Item  
  2          5  
  2          4  
  3          2  
  3          3  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 2  
  
Deleted item is 5  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 1  
  
Input the item to be added in the queue : 6  
  
Enter its priority : 1
```

```
Enter its priority : 1  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 3  
  
Queue is :  
Priority      Item  
  1          6  
  2          4  
  3          2  
  3          3  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 2  
  
Deleted item is 6  
  
1.Insert  
2.Delete
```

```
3.Display  
4.Quit  
  
Enter your choice : 2  
  
Deleted item is 4  
  
1.Insert  
2.Delete  
3.Display  
4.Quit  
  
Enter your choice : 3
```

```

Queue is :

Priority      Item
 3            2
 3            3

1.Insert
2.Delete
3.Display
4.Quit

Enter your choice : 4

...Program finished with exit code 1
Press ENTER to exit console.■

```

Dequeue:

```

1 #include<stdio.h>
3 #include<conio.h>
4 #include<stdlib.h>
5 #define qsize 5
6 int f=0,r=-1,ch;
7 int item,q[10];
8
9 int isfull()
10 {
11     return(r==qsize-1)?1:0;
12 }
13 int isempty()
14 {
15     return(f>r)?1:0;
16 }
17 void insert_rear()
18 {
19     if(isfull())
20     {
21         printf("queue overflow\n");
22         return;
23     }
24     r=r+1;
25     q[r]=item;
26 }
27 void delete_front()
28 {
29     if(isempty())
30     {
31         printf("queue empty\n");
32         return;
33     }
34     printf("item deleted is %d\n",q[(f)+]);
35     if(f>r)
36     {
37         f=0;
38         r=-1;
39     }
40 }
41 void insert_front()
42 {
43     if(f!=0)
44     {
45         f=f-1;
46         q[f]=item;
47         return;
48     }
49 else if((f==0)&&(r==0))
50 {
51     q[++(r)]=item;
52     return;
53 }
54 else
55     printf("insertion not possible\n");
56 }
57 void delete_rear()
58 {
59     if(isempty())
60     {
61         printf("queue is empty\n");
62         return;
63     }
64 }

```

```
61  printf("queue is empty\n");
62  return;
63 }
64  printf("item deleted is %d\n",q[(r)--]);
65  if(f>r)
66 {
67  f=0;
68  r=-1;
69 }
70 }
71 void display()
72 {
73  int i;
74  if(isempty())
75 {
76  printf("queue empty\n");
77  return;
78 }
79  for(i=f;i<=r;i++)
80 printf("%d\n",q[i]);
81 }
82 void main()
83 {
84 {for();}
85 printf("1.insert_rear\n2.insert_front\n3.delete_rear\n4.delete_front\n5.display\n6.exit\n");
86 printf("enter choice\n");
87 scanf("%d",&ch);
88 switch(ch)
89 {
90  case 1:printf("enter the item\n");
91  scanf("%d",&item);
92  insert_rear();
93  break;
94  case 2:printf("enter the item\n");
95  scanf("%d",&item);
96  insert_front();
97  break;
98  case 3:delete_rear();
99  break;
100 case 4:delete_front();
101 break;
102 case 5:display();
103 break;
104 default:exit(0);
105 }}}
```

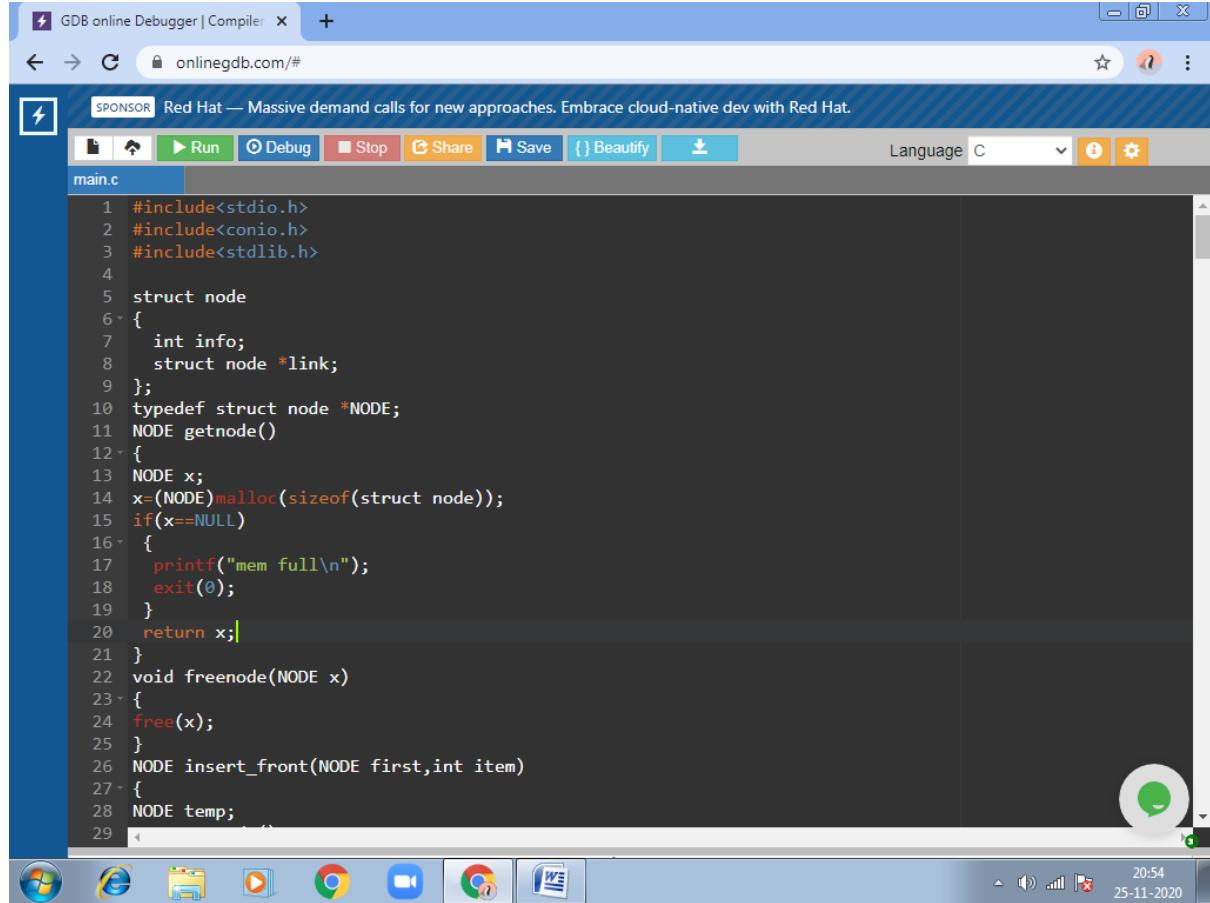
```
-  
enter the item  
6  
insertion not possible  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit  
enter choice  
1  
enter the item  
9  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit  
enter choice  
5  
2  
9  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit  
enter choice  
4
```

```
-  
enter choice  
4  
item deleted is 2  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit  
enter choice  
4  
item deleted is 9  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit
```

```
enter choice  
5  
queue empty  
1.insert_rear  
2.insert_front  
3.delete_rear  
4.delete_front  
5.display  
6.exit  
enter choice  
6  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

wap to implement linked list and its various operations

CODE:



The screenshot shows a web-based GDB debugger interface. At the top, there's a header bar with a logo, the text "GDB online Debugger | Compiler", and a search bar containing "onlinegdb.com/#". Below the header is a toolbar with icons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to C. A banner from Red Hat is visible. The main area displays the "main.c" file content:

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert_front(NODE first,int item)
27 {
28     NODE temp;
29 }
```

The code implements a simple singly linked list with functions for creating nodes, inserting at the front, and freeing memory. The interface includes a status bar at the bottom showing system icons and the date/time (25-11-2020, 20:54).

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
28 NODE temp;
29 temp=getnode();
30 temp->info=item;
31 temp->link=NULL;
32 if(first==NULL)
33 return temp;
34 temp->link=first;
35 first=temp;
36 return first;
37 }
38 NODE IF(NODE second,int item)
39 {
40 NODE temp;
41 temp=getnode();
42 temp->info=item;
43 temp->link=NULL;
44 if(second==NULL)
45 return temp;
46 temp->link=second;
47 second=temp;
48 return second;
49 }
50 NODE delete_front(NODE first)
51 {
52 NODE temp;
53 if(first==NULL)
54 {
55 printf("list is empty cannot delete\n");
56 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:54 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
56 return first;
57 }
58 temp=first;
59 temp=temp->link;
60 printf("item deleted at front-end is=%d\n",first->info);
61 free(first);
62 return temp;
63 }
64 NODE insert_rear(NODE first,int item)
65 {
66 NODE temp,cur;
67 temp=getnode();
68 temp->info=item;
69 temp->link=NULL;
70 if(first==NULL)
71 return temp;
72 cur=first;
73 while(cur->link!=NULL)
74 cur=cur->link;
75 cur->link=temp;
76 return first;
77 }
78 NODE IR(NODE second,int item)
79 {
80 NODE temp,cur;
81 temp=getnode();
82 temp->info=item;
83 temp->link=NULL;
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:55 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
84 if(second==NULL)
85     return temp;
86 cur=second;
87 while(cur->link!=NULL)
88     cur=cur->link;
89 cur->link=temp;
90 return second;
91 }
92 NODE delete_rear(NODE first)
93 {
94 NODE cur,prev;
95 if(first==NULL)
96 {
97 printf("list is empty cannot delete\n");
98 return first;
99 }
100 if(first->link==NULL)
101 {
102 printf("item deleted is %d\n",first->info);
103 free(first);
104 return NULL;
105 }
106 prev=NULL;
107 cur=first;
108 while(cur->link!=NULL)
109 {
110 prev=cur;
111 cur=cur->link;
112 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:55 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
111 cur=cur->link;
112 }
113 printf("item deleted at rear-end is %d",cur->info);
114 free(cur);
115 prev->link=NULL;
116 return first;
117 }
118 NODE insert_pos(int item,int pos,NODE first)
119 {
120 NODE temp;
121 NODE prev,cur;
122 int count;
123 temp=getnode();
124 temp->info=item;
125 temp->link=NULL;
126 if(first==NULL && pos==1)
127 return temp;
128 if(first==NULL)
129 {
130 printf("invalid pos\n");
131 return first;
132 }
133 if(pos==1)
134 {
135 temp->link=first;
136 return temp;
137 }
138 count=1;
139 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:55 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
139 prev=NULL;
140 cur=first;
141 while(cur!=NULL && count!=pos)
142 {
143     prev=cur;
144     cur=cur->link;
145     count++;
146 }
147 if(count==pos)
148 {
149     prev->link=temp;
150     temp->link=cur;
151     return first;
152 }
153 printf("Invalid Position \n");
154 return first;
155 }
156 NODE delete_pos(int pos, NODE first)
157 {
158     NODE cur;
159     NODE prev;
160     int count;
161     if(first==NULL || pos<=0)
162     {
163         printf("invalid position \n");
164         return NULL;
165     }
166     if (pos==1) |
167 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:56 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
166 if (pos==1)
167 {
168     cur=first;
169     first=first->link;
170     freenode(cur);
171     return first;
172 }
173 prev=NULL;
174 cur=first;
175 count=1;
176 while(cur!=NULL)
177 {
178     if(count==pos)
179         break; //if found
180     prev=cur;
181     cur=cur->link;
182     count++;
183 }
184 if(count!=pos)
185 {
186     printf("invalid position\n");
187     return first;
188 }
189 if(count!=pos)
190 {
191     printf("invalid position specified\n");
192     return first;
193 }
194 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:56 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
193 }
194 prev->link=cur->link;
195 freenode(cur);
196 return first;
197 }
198 NODE reverse(NODE first)
199 {
200     NODE cur,temp;
201     cur=NULL;
202     while(first!=NULL)
203     {
204         temp=first;
205         first=first->link;
206         temp->link=cur;
207         cur=temp;
208     }
209     return cur;
210 }
211 NODE asc(NODE first)
212 {
213     NODE prev=first;
214     NODE cur=NULL;
215     int temp;
216
217     if(first== NULL) {
218         return 0;
219     }
220 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download, Help, Settings

20:56 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
220 }
221 else {
222     while(prev!= NULL) {
223
224         cur = prev->link;
225
226         while(cur!= NULL) {
227
228             if(prev->info > cur->info) {
229                 temp = prev->info;
230                 prev->info = cur->info;
231                 cur->info = temp;
232             }
233             cur = cur->link;
234         }
235         prev= prev->link;
236     }
237 }
238 return first;
239 }
240 NODE des(NODE first)
241 {
242     NODE prev=first;
243     NODE cur=NULL;
244     int temp;
245
246     if(first==NULL) {
247         return 0;
248 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download, Help, Settings

20:57 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
247     return 0;
248 }
249 else {
250     while(prev!= NULL) {
251         cur = prev->link;
252         while(cur!= NULL) {
253             if(prev->info < cur->info) {
254                 temp = prev->info;
255                 prev->info = cur->info;
256                 cur->info = temp;
257             }
258             cur = cur->link;
259         }
260         prev= prev->link;
261     }
262     return first;
263 }
264 NODE concate(NODE first,NODE second)
265 {
266     NODE cur;
267     if(first==NULL)
268         return second;
269     if(second==NULL)
270         return first;
271 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:57 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
274     return first;
275     cur=first;
276     while(cur->link!=NULL)
277     {
278         cur=cur->link;
279     }
280     cur->link=second;
281     return first;
282 }
283 }
284 void display(NODE first)
285 {
286     NODE temp;
287     if(first==NULL)
288         printf("list empty cannot display items\n");
289     for(temp=first;temp!=NULL,temp=temp->link)
290     {
291         printf("%d\n",temp->info);
292     }
293 }
294 void main()
295 {
296     int item,choice,pos;element,option,choice2,item1,num;
297     NODE first=NULL;
298     NODE second=NULL;
299
300     for(;;)
301 }
```

Language: C

Run, Debug, Stop, Share, Save, Beautify, Download

20:57 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
301 for(;;)
302 {
303     printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:reverse\n");
304     printf("enter the choice\n");
305     scanf("%d",&choice);
306     switch(choice)
307     {
308         case 1:printf("enter the item at front-end\n");
309             scanf("%d",&item);
310             first=insert_front(first,item);
311             break;
312         case 2:first=delete_front(first);
313             break;
314         case 3:printf("enter the item at rear-end\n");
315             scanf("%d",&item);
316             first=insert_rear(first,item);
317             break;
318         case 4:first=delete_rear(first);
319             break;
320         case 5:
321             printf("press 1 to insert or 2 to delete at any desired position \n");
322             scanf("%d",&element);
323             if(element==1){
324                 printf("enter the position to insert \n");
325                 scanf("%d",&pos);
326                 printf("enter the item to insert \n");
327                 scanf("%d",&item);
328                 first=insert_pos(item,pos,first);
329             }
330             if(element==2){
331                 printf("enter the position to delete \n");
332                 scanf("%d",&pos);
333                 first=delete_pos(pos,first);
334             }
335             break;
336         case 6:
337             first=reverse(first);
338             break;
339         case 7:
340             printf("press 1 for ascending sort and 2 for descending sort:\n");
341             scanf("%d",&option);
342             if(option==1)
343                 first=asc(first);
344             if(option==2)
345                 first=des(first);
346             break;
347         case 8:
348             printf("create a second list\n");
349             printf("enter the number of elements in second list\n");
350
351             scanf("%d",&num);
352             for(int i=1;i<num;i++){
353                 printf("\n press 1 to insert front and 2 to insert rear \n");
354                 scanf("%d",&choice2);
355             }
356     }
357 }
```

Language C

20:58 25-11-2020

GDB online Debugger | Compiler

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

main.c

```
328         first=insert_pos(item,pos,first);
329     }
330     if(element==2){
331         printf("enter the position to delete \n");
332         scanf("%d",&pos);
333         first=delete_pos(pos,first);
334     }
335     break;
336 case 6:
337     first=reverse(first);
338     break;
339 case 7:
340     printf("press 1 for ascending sort and 2 for descending sort:\n");
341     scanf("%d",&option);
342     if(option==1)
343         first=asc(first);
344     if(option==2)
345         first=des(first);
346     break;
347 case 8:
348     printf("create a second list\n");
349     printf("enter the number of elements in second list\n");
350
351     scanf("%d",&num);
352     for(int i=1;i<num;i++){
353         printf("\n press 1 to insert front and 2 to insert rear \n");
354         scanf("%d",&choice2);
355     }
356 }
```

Language C

20:58 25-11-2020

GDB online Debugger | Compiler +

SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.

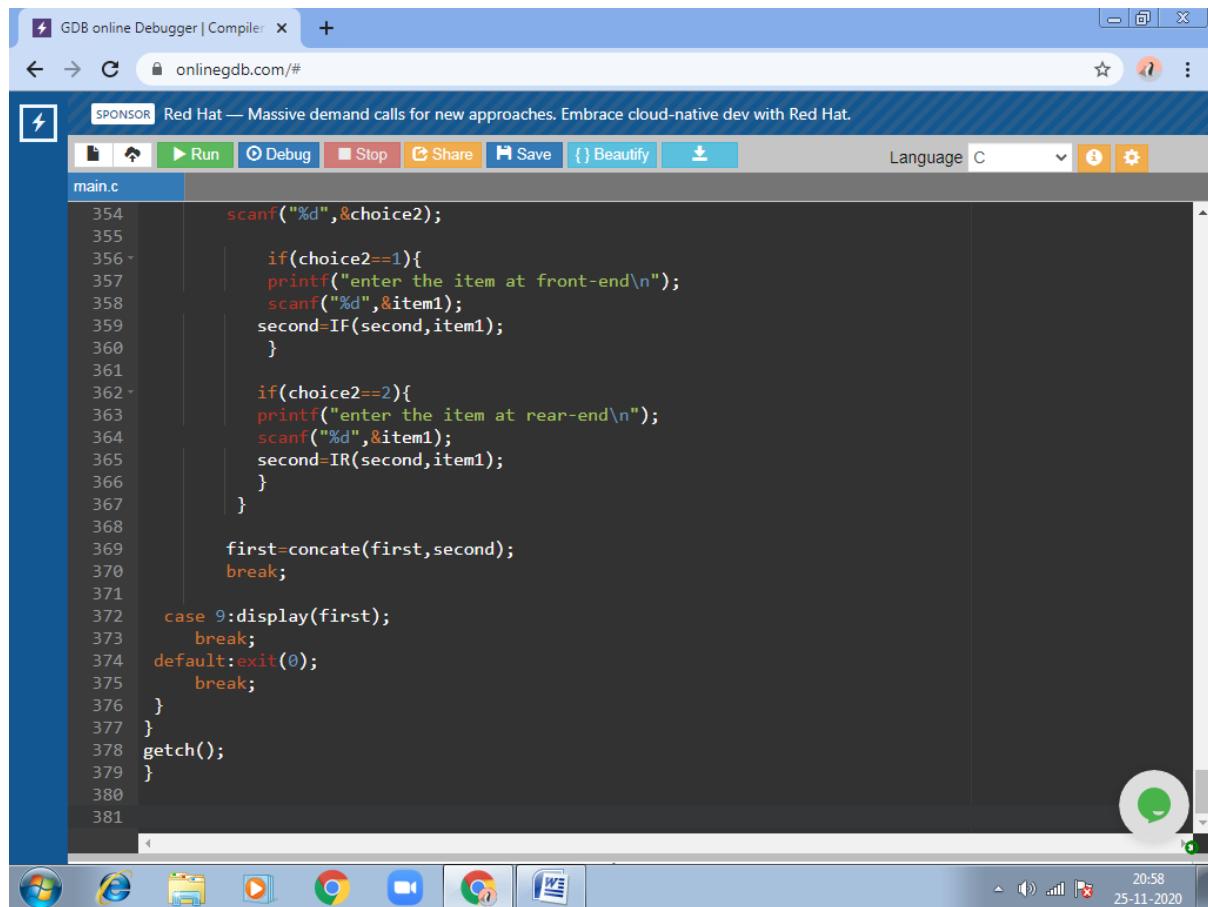
main.c

```
354     scanf("%d",&choice2);
355
356     if(choice2==1){
357         printf("enter the item at front-end\n");
358         scanf("%d",&item1);
359         second=IF(second,item1);
360     }
361
362     if(choice2==2){
363         printf("enter the item at rear-end\n");
364         scanf("%d",&item1);
365         second=IR(second,item1);
366     }
367
368     first=concat(first,second);
369     break;
370
371     case 9:display(first);
372     break;
373     default:exit(0);
374     break;
375 }
376 }
377 getch();
378 }
```

Language: C

Run | Debug | Stop | Share | Save | Beautify |

20:58 25-11-2020



OUTPUT:

The screenshot shows a Windows operating system interface. At the top, there is a taskbar with several icons: Start, Internet Explorer, File Explorer, Task View, Google Chrome, and Microsoft Word. In the center, a browser window titled "GDB online Debugger | Compiler" displays the URL "onlinegdb.com/#". The main content area of the browser shows a terminal window titled "input". The terminal displays the following text:

```
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  
1  
  
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  
2
```

The bottom right corner of the screen shows the date and time as "25-11-2020 20:45".

GDB online Debugger | Compiler

onlinegdb.com/#

```
enter the item at front-end
2

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
3

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
```

input

20:46 25-11-2020

GDB online Debugger | Compiler

onlinegdb.com/#

```
3
enter the item at rear-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
9
3
2
1
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
```

input

20:46 25-11-2020

GDB online Debugger | Compiler

onlinegdb.com/#

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

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
4
item deleted at rear-end is 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
9
2
```

GDB online Debugger | Compiler

onlinegdb.com/#

```
enter the choice
9
2
1

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
6

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
```

GDB online Debugger | Compiler

onlinegdb.com/#

```
input
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
8

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
9
8
6
2
1

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
```

20:48
25-11-2020

GDB online Debugger | Compiler

onlinegdb.com/#

```
input
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
3
enter the item to insert
6

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
9
8
6
2
1
```

20:49
25-11-2020

GDB online Debugger | Compiler

onlinegdb.com/#

```
input
2
1

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
6

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
9
1
2
6
```

GDB online Debugger | Compiler

onlinegdb.com/#

```
input
6
8

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
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.concatenate
9:display_list
10:Exit
enter the choice
9
1
```

GDB online Debugger | Compiler

onlinegdb.com/#

```
enter the choice
9
1
2
6
6
8

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
8
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
1

press 1 to insert front and 2 to insert rear
1
```



20:50
25-11-2020

GDB online Debugger | Compiler

onlinegdb.com/#

```
press 1 to insert front and 2 to insert rear
1
enter the item at front-end
3

press 1 to insert front and 2 to insert rear
2
enter the item at rear-end
5

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
9
1
2
6
6
8
3
1
5
```

GDB online Debugger | Compiler

onlinegdb.com/#

```
7:sort
8.concatenate
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5

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
10

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

Q1)wap to implement queues using linked lists:

The screenshot shows a web-based GDB debugger interface. The title bar reads '(2) WhatsApp' and 'GDB online Debugger | Compiler'. The main window displays a C program named 'main.c'.

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4 struct node
5 {
6     int info;
7     struct node *link;
8 };
9 typedef struct node *NODE;
10 NODE getnode()
11 {
12 NODE x;
13 x=(NODE)malloc(sizeof(struct node));
14 if(x==NULL)
15 {
16     printf("mem full\n");
17     exit(0);
18 }
19 return x;
20 }
21 void freenode(NODE x)
22 {
23 free(x);
24 }
25 NODE insert_rear(NODE first,int item)
26 {
27 NODE temp,cur;
28 temp=getnode();
29 temp->info=item;
30 temp->link=NULL;
31 if(first==NULL)
32     first=temp;
```

The code implements a simple queue using a linked list. It defines a 'node' structure with 'info' and 'link' fields. A 'NODE' type is defined as a pointer to a 'node'. The 'getnode' function allocates memory for a new node and returns it. The 'freenode' function frees the memory of a node. The 'insert_rear' function inserts a new node at the end of the queue. The program starts by defining the queue structure and then calls 'insert_rear' with an initial value of 10. The rest of the code is commented out.

(2) WhatsApp GDB online Debugger | Compiler

onlinergdb.com

main.c

```
1 if(first==NULL)
2     return temp;
3 cur=first;
4 while(cur->link!=NULL)
5     cur=cur->link;
6 cur->link=temp;
7 return first;
8 }
9
10 NODE delete_front(NODE first)
11 {
12 NODE temp;
13 if(first==NULL)
14 {
15 printf("list is empty cannot delete\n");
16 return first;
17 }
18 temp=first;
19 temp=temp->link;
20 printf("item deleted at front-end is=%d\n",first->info);
21 free(first);
22 return temp;
23 }
24 void display(NODE first)
25 {
26 NODE temp;
27 if(first==NULL)
28 printf("list empty cannot display items\n");
29 for(temp=first;temp!=NULL;temp=temp->link)
30 {
31     printf("%d\n",temp->info);
32 }
```

Language: C

23:13 03-01-2021

(2) WhatsApp GDB online Debugger | Compiler

onlinergdb.com

main.c

```
58 printf("list empty cannot display items\n");
59 for(temp=first,temp!=NULL,temp=temp->link)
60 {
61     printf("%d\n",temp->info);
62 }
63 }
64 void main()
65 {
66 int item,choice,pos;
67 NODE first=NULL;
68 for(;;)
69 {
70     printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
71     printf("enter the choice\n");
72     scanf("%d",&choice);
73     switch(choice)
74     {
75         case 1:printf("enter the item at rear-end\n");
76             scanf("%d",&item);
77             first=insert_rear(first,item);
78             break;
79         case 2:first=delete_front(first);
80             break;
81         case 3:display(first);
82             break;
83         default:exit(0);
84             break;
85     }
86 }
87 getch();
88 }
89 }
```

Language: C

23:13 03-01-2021

(2) WhatsApp GDB online Debugger | Compiler

onlinegdb.com

input

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

https://www.onlinegdb.com/#tab-stdin

23:13 03-01-2021

(2) WhatsApp GDB online Debugger | Compiler

onlinegdb.com

input

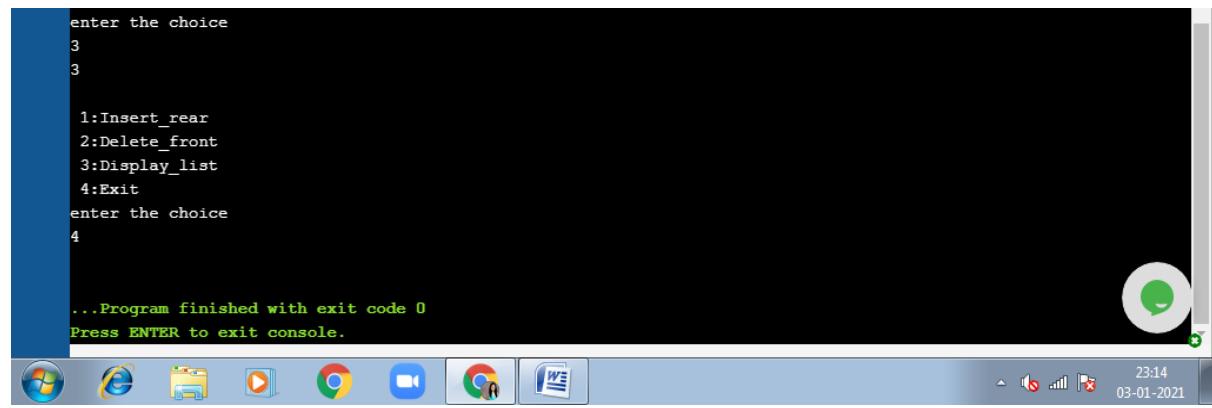
```
2:Delete_front  
3:Display_list  
4:Exit  
enter the choice  
3  
1  
2  
3  
  
1:Insert_rear  
2:Delete_front  
3:Display_list  
4:Exit  
enter the choice  
2  
item deleted at front-end is=1  
  
1:Insert_rear  
2:Delete_front  
3:Display_list  
4:Exit  
enter the choice  
2  
item deleted at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

23:14 03-01-2021

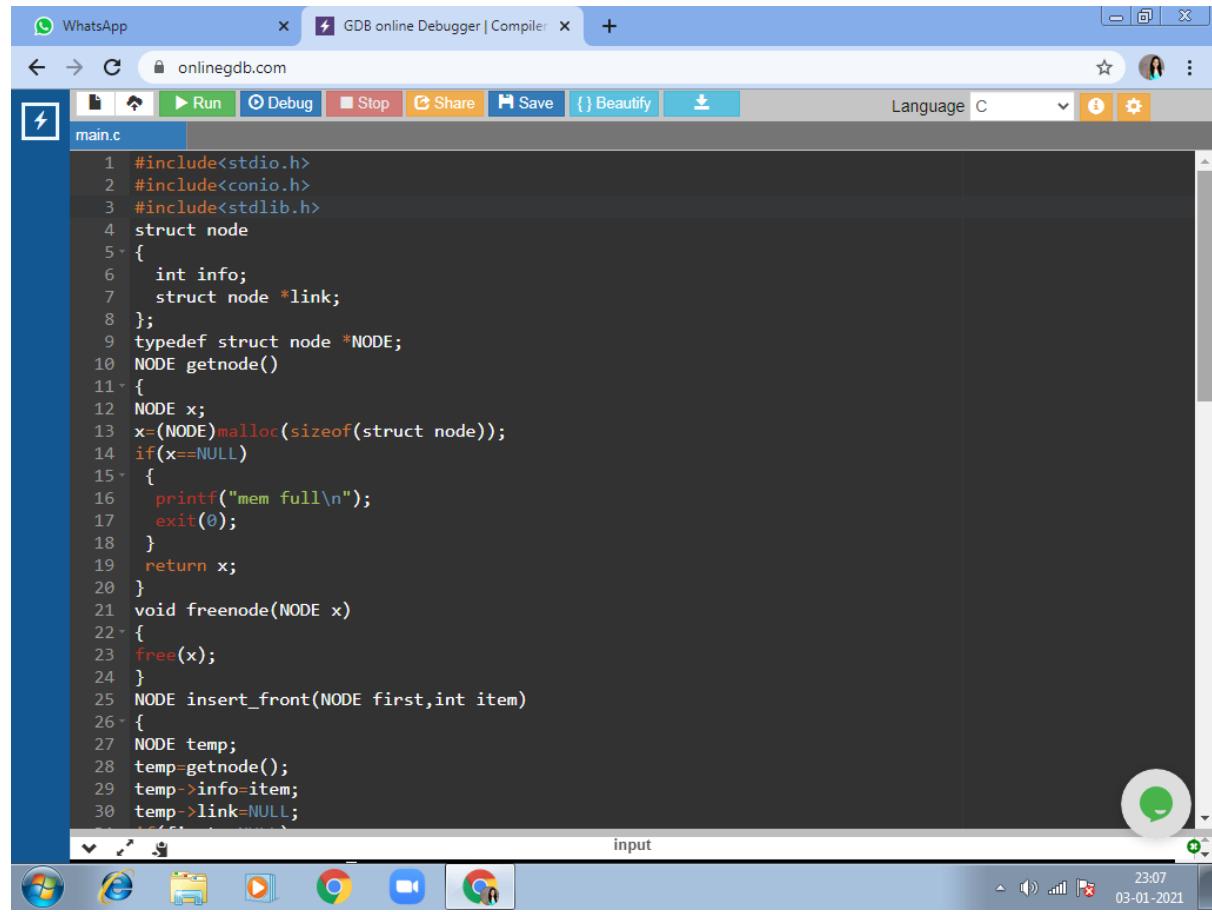
```
enter the choice
3
3

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

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



Q2)wap to implement stacks using linked lists:



```
#include<stdio.h>
#include<conio.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;
    temp->link=first;
```

WhatsApp GDB online Debugger | Compiler +

onlinergdb.com

Run Debug Stop Share Save Beautify Language C

main.c

```
30 temp->link=NULL;
31 if(first==NULL)
32 return temp;
33 temp->link=first;
34 first=temp;
35 return first;
36 }
37 NODE delete_front(NODE first)
38 {
39 NODE temp;
40 if(first==NULL)
41 {
42 printf("stack is empty cannot delete\n");
43 return first;
44 }
45 temp=first;
46 temp=temp->link;
47 printf("item deleted at front-end is=%d\n",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 }
```

input

23:08 03-01-2021

WhatsApp GDB online Debugger | Compiler +

onlinergdb.com

Run Debug Stop Share Save Beautify Language C

main.c

```
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 void main()
62 {
63 int item,choice,pos;
64 NODE first=NULL;
65 for(;;)
66 {
67 printf("\n 1:Insert_front[PUSH]\n 2:Delete_front[pop]\n 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 }
```

input

23:08 03-01-2021

WhatsApp GDB online Debugger | Compiler

onlinegdb.com

input

```
1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
1

1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
2

1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
3

1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
```

23:08 03-01-2021

WhatsApp GDB online Debugger | Compiler

onlinegdb.com

input

```
3
2
1

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

1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
4:Exit
enter the choice
3
2
1

1:Insert_front[PUSH]
2:Delete_front[pop]
3:Display_list
4:Exit
enter the choice
4

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

23:09 03-01-2021

PRACTICE PROG:

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert_front(NODE first,int item)
27 {
28     NODE temp;
29     temp=getnode();
30     temp->info=item;
31     temp->link=NULL;
32
33     if(first==NULL)
34         return temp;
35     temp->link=first;
36     first=temp;
37     return first;
38 }
39 NODE delete_rear(NODE first)
40 {
41     NODE cur,prev;
42     if(first==NULL)
43     {
44         printf("list is empty cannot delete\n");
45         return first;
46     }
47     if(first->link==NULL)
48     {
49         printf("item deleted is %d\n",first->info);
50         free(first);
51         return NULL;
52     }
53     prev=NULL;
54     cur=first;
55     while(cur->link!=NULL)
56     {
57         prev=cur;
58         cur=cur->link;
59     }
60     printf("item deleted at rear-end is %d",cur->info);
61     free(cur);
```

```
60     printf("item deleted at rear-end is %d ", cur->info);
61     free(cur);
62     prev->link=NULL;
63     return first;
64 }
65 NODE asc(NODE first)
66 {
67     NODE prev=first;
68     NODE cur=NULL;
69     int temp;
70
71     if(first==NULL) {
72         return 0;
73     }
74     else {
75         while(prev!=NULL) {
76
77             cur = prev->link;
78
79             while(cur!=NULL) {
80
81                 if(prev->info > cur->info) {
82                     temp = prev->info;
83                     prev->info = cur->info;
84                     cur->info = temp;
85                 }
86                 cur = cur->link;
87             }
88             prev= prev->link;
89         }
90     }
91 }
```

```
91     return first;
92 }
93 NODE des(NODE first)
94 {
95     NODE prev=first;
96     NODE cur=NULL;
97     int temp;
98
99     if(first==NULL) {
100         return 0;
101     }
102     else {
103         while(prev!=NULL) {
104
105             cur = prev->link;
106
107             while(cur!=NULL) {
108
109                 if(prev->info < cur->info) {
110                     temp = prev->info;
111                     prev->info = cur->info;
112                     cur->info = temp;
113                 }
114                 cur = cur->link;
115             }
116             prev= prev->link;
117         }
118     }
119     return first;
120 }
121 int count(NODE first)
```

```
120         }
121     int count(NODE first)
122    {
123        NODE cur;
124        cur=first;
125        int c;
126        if(first==NULL)
127        {
128            c=0;
129        }
130        return c;
131    }
132    if(first->link==NULL)
133    {
134        c=1;
135        return c;
136    }
137    while(cur!=NULL)
138    {
139        c++;
140        cur=cur->link;
141    }
142    return c;
143 }
144 }
145 }
146 void search(NODE first,int ele)
147 {
148     NODE cur;
149     cur=first;
150 }
```

```
150     cur=first;
151     int flag=0;
152     if(first==NULL)
153     {
154         printf("list empty can't search \n");
155         return ;
156     }
157     while(cur!=NULL)
158     {
159         if(ele==cur->info)
160         {
161             flag=1;
162             break;
163         }
164         cur=cur->link;
165     }
166     if(flag==1)
167     printf("SEARCH SUCCESSFULL \n");
168     else
169     printf("SEARCH UNSUCCESSFULL \n");
170
171 }
172 }
173 void display(NODE first)
174 {
175     NODE temp;
176     if(first==NULL)
177     printf("list empty cannot display items\n");
178     for(temp=first,temp!=NULL,temp=temp->link)
179     {
180         printf("%d\n",temp->info);
181     }
182 }
```

```

179    {
180        printf("%d\n",temp->info);
181    }
182 }
183 void main()
184 {
185 int item,choice,choice2,j,key;
186 NODE first=NULL;
187
188 for(;;)
189 {
190 printf("\n 1:Insert_front\n 2:Delete_rear\n 3:count \n 4:sort\n 5.search\n 6:display_list\n 7:Exit\n");
191 printf("enter the choice\n");
192 scanf("%d",&choice);
193 switch(choice)
194 {
195     case 1:
196         printf("enter item to be inserted at front end \n");
197         scanf("%d",&item);
198         first=insert_front(first,item);
199         break;
200     case 2:
201         delete_rear(first);
202         break;
203     case 3:
204         j=count(first);
205         printf("no of items in list: %d ",j);
206         break;
207     case 4:
208         printf("press 1 for ascending order and 2 for descending order \n");
209
210     case 4:
211         printf("press 1 for ascending order and 2 for descending order \n");
212         scanf("%d",&choice2);
213         if(choice2==1)
214             first=asc(first);
215         if(choice2==2)
216             first=des(first);
217         break;
218     case 5:
219         printf("enter element to be searched for \n");
220         scanf("%d",&key);
221         search(first,key);
222         break;
223     case 6:
224         display(first);
225         break;
226     default:exit(0);
227         break;
228     }
229 }
230
231 getch();

```

OUTPUT:

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
5

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
4

1:Insert_front
2:Delete_rear
3:count
4:sort
```

```
4:sort
5.search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
3

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
1
enter item to be inserted at front end
2

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
```

```
enter the choice
1
enter item to be inserted at front end
1

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
6
1
2
3
4
5

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
3
```

```
enter the choice
3
no of items in list: 5
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
4
press 1 for ascending order and 2 for descending order
2

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
6
5
4
3
2
1
```

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
5
enter element to be searched for
20
SEARCH UNSUCCESSFULL

1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
5
enter element to be searched for
3
SEARCH SUCCESSFULL

1:Insert_front
2:Delete_rear
```

```
1:Insert_front
2:Delete_rear
3:count
4:sort
5.search
6:display_list
7:Exit
enter the choice
7

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

wap to implement doubly linked list and its operations

```
main.c
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4 struct node
5 {
6     int info;
7     struct node *llink;
8     struct node *rlink;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE dinsert_front(int item,NODE head)
27 {
28     NODE temp,cur;
29     temp=getnode();
30     temp->info=item;
31     cur=head->rlink;
32     head->rlink=temp;
33     temp->llink=head;
34     temp->rlink=cur;
35     cur->llink=temp;
36     return head;
37 }
38 NODE dinsert_rear(int item,NODE head)
39 {
40     NODE temp,cur;
41     temp=getnode();
42     temp->info=item;
43     cur=head->llink;
44     head->llink=temp;
45     temp->rlink=head;
46     temp->llink=cur;
47     cur->rlink=temp;
48     return head;
49 }
50 NODE ddelete_front(NODE head)
51 {
52     NODE cur,next;
53     if(head->rlink==head)
54     {
55         printf("dq empty\n");
56         return head;
57     }
58     cur=head->rlink;
59     next=cur->rlink;
60     head->rlink=next;
61     next->llink=head;
```

```

61 next->llink=head;
62 printf("the node deleted is %d",cur->info);
63 freenode(cur);
64 return head;
65 }
66 NODE ddelete_rear(NODE head)
67 {
68 NODE cur,prev;
69 if(head->rlink==head)
70 {
71 printf("dq empty\n");
72 return head;
73 }
74 cur=head->llink;
75 prev=cur->llink;
76 head->llink=prev;
77 prev->rlink=head;
78 printf("the node deleted is %d",cur->info);
79 freenode(cur);
80 return head;
81 }
82
83 NODE insert_leftpos(int item,NODE head)
84 {
85 NODE temp,cur,prev;
86 if(head->rlink==head)
87 {
88 printf("list empty\n");
89 return head;
90 }
91 cur=head->rlink;|
92 while(cur!=head)
93 {
94 if(item==cur->info)break;
95 cur=cur->rlink;
96 }
97 if(cur==head)
98 {
99 printf("key not found\n");
100 return head;
101 }
102 prev=cur->llink;
103 printf("enter towards left of %d=%d",item);
104 temp=getnode();
105 scanf("%d",&temp->info);
106 prev->rlink=temp;
107 temp->llink=prev;
108 cur->llink=temp;
109 temp->rlink=cur;
110 return head;
111 }
112
113 NODE insert_rightpos(int item,NODE head)
114 {
115 NODE temp,cur,next;
116 if(head->rlink==head)
117 {
118 printf("list empty\n");
119 return head;
120 }
121 cur=head->rlink;|

```

```
121 cur=head->rlink;
122 while(cur!=head)
123 {
124     if(item==cur->info)break;
125     cur=cur->rlink;
126 }
127 if(cur==head)
128 {
129     printf("key not found\n");
130     return head;
131 }
132 next=cur->rlink;
133 printf("enter towards right of %d=%d",item);
134 temp=getnode();
135 scanf("%d",&temp->info);
136 cur->rlink=temp;
137 temp->llink=cur;
138 next->llink=temp;
139 temp->rlink=next;
140 return head;
141 }
142 NODE search(NODE head,int item)
143 {
144     NODE temp,cur;
145     int flag=0;
146     if(head->rlink==head)
147     {
148         printf("list empty\n");
149         return head;
150     }
151     cur=head->rlink;
```

```
151 cur=head->rlink;
152 while(cur!=head)
153 {
154     if(item==cur->info)
155     {
156         flag=1;
157         break;
158     }
159     cur=cur->rlink;
160 }
161 if(cur==head)
162     printf("search unsuccessfull\n");
163 if(flag==1)
164     printf("search successfull\n");
165 }
166 NODE delete_all_key(int item,NODE head)
167 {
168     NODE prev,cur,next;
169     int count;
170     if(head->rlink==head)
171     {
172         printf("list empty\n");
173         return head;
174     }
175     count=0;
176     cur=head->rlink;
177     while(cur!=head)
178     {
179         if(item!=cur->info)
180             cur=cur->rlink;
181         else|
```

```

181     else
182     {
183         count++;
184         prev=cur->llink;
185         next=cur->rlink;
186         prev->rlink=next;
187         next->llink=prev;
188         freenode(cur);
189         cur=next;
190     }
191 }
192 if(count==0)
193 printf("not found\n");
194 else{
195     printf("found at %d positions and are deleted",count);
196     return head;
197 }
198 }
199
200 void display(NODE head)
201 {
202 NODE temp;
203 if(head->rlink==head)
204 {
205 printf("dq empty\n");
206 return;
207 }
208 printf("contents of dq\n");
209 temp=head->rlink;
210 while(temp!=head)||
211 {
210 while(temp!=head)
211 {
212 printf("%d",temp->info);
213 temp=temp->rlink;
214 }
215 printf("\n");
216 }
217 void main()
218 {
219 NODE head,last;
220 int item, choice;
221 head=getnode();
222 head->rlink=head;
223 head->llink=head;
224 //clrscr();
225 for(;;)
226 {
227     printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of key\n6:display\n7:exit");
228     printf("enter the choice\n");
229     scanf("%d",&choice);
230     switch(choice)
231     {
232         case 1: printf("enter the item at front end\n");
233             scanf("%d",&item);
234             last=dinsert_front(item,head);
235             break;
236         case 2: printf("enter the item at rear end\n");
237             scanf("%d",&item);
238             last=dinsert_rear(item,head);
239             break;
240         case 3:last=ddelete_front(head);

```

```
240     case 3:last=ddelete_front(head);
241         break;
242     case 4: last=ddelete_rear(head);
243         break;
244
245     case 5:
246         printf("enter the key element\n");
247         scanf("%d",&item);
248         last=insert_leftpos(item,head);
249         break;
250     case 6:
251         printf("enter the key element\n");
252         scanf("%d",&item);
253         last=insert_rightpos(item,head);
254         break;
255     case 7:
256         printf("enter the search element\n");
257         scanf("%d",&item);
258         search(head,item);
259         break;
260     case 8: printf("enter element to be deleted\n");
261         scanf("%d",&item);
262         last=delete_all_key(item,head);
263     case 9: display(head);
264         break;
265     default:exit(0);
266 }
267 getch();
268 }
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
1
enter the item at front end
1

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
1
```

```
enter the choice
1
enter the item at front end
2

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
2
enter the item at rear end
3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
```

```
8:delete repeating occurrences
9:display
10:exit
enter the choice
2
enter the item at rear end
3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
1
enter the item at front end
4

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
```

```
-  
4:delete rear  
5:insert left of key element  
6:insert right of key element  
7:search  
8:delete repeating occurrences  
9:display  
10:exit  
enter the choice  
9  
contents of dq  
5421  
  
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:insert left of key element  
6:insert right of key element  
7:search  
8:delete repeating occurrences  
9:display  
10:exit  
enter the choice  
7  
enter the search element  
4  
search successfull
```

```
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:insert left of key element  
6:insert right of key element  
7:search  
8:delete repeating occurrences  
9:display  
10:exit  
enter the choice  
4  
the node deleted is 1  
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:insert left of key element  
6:insert right of key element  
7:search  
8:delete repeating occurrences  
9:display  
10:exit  
enter the choice  
9
```

```
10:exit
enter the choice
9
contents of dq
542

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
6
enter the key element
4
enter towards right of 4=3

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element

6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
9
contents of dq
5432

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
10

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

wap to implement binary search tree

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4 struct node
5 {
6     int info;
7     struct node *rlink;
8     struct node *llink;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE)malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert(NODE root,int item)
27 {
28     NODE temp,cur,prev;
29     temp=getnode();
30     temp->rlink=NULL;
31     temp->llink=NULL;
32     temp->info=item;
33     if(root==NULL)
34     {
35         return temp;
36     }
37     cur=root;
38     while(cur!=NULL)
39     {
40         prev=cur;
41         cur=(item<cur->info)?cur->llink:cur->rlink;
42     }
43     if(item<prev->info)
44     {
45         prev->llink=temp;
46     }
47     else
48     {
49         prev->rlink=temp;
50     }
51     return root;
52 }
53 void display(NODE root,int i)
54 {
55     int j;
56     if(root!=NULL)
57     {
58         display(root->rlink,i+1);
59         for(j=0;j<i;j++)
60             printf("   ");
61         printf("%d\n",root->info);
62         display(root->llink,i+1);
63     }
64 }
```

```
55     printf("   ");
56     printf("%d\n",root->info);
57     display(root->llink,i+1);
58 }
59 }
60 NODE delete(NODE root,int item)
61 {
```

```
60 NODE delete(NODE root,int item)
61 {
62 NODE cur,parent,q,suc;
63 if(root==NULL)
64 {
65 printf("empty\n");
66 return root;
67 }
68 parent=NULL;
69 cur=root;
70 while(cur!=NULL&&item!=cur->info)
71 {
72 parent=cur;
73 cur=(item<cur->info)?cur->llink:cur->rlink;
74 }
75 if(cur==NULL)
76 {
77 printf("not found\n");
78 return root;
79 }
80 if(cur->llink==NULL)
81 q=cur->rlink;
82 else if(cur->rlink==NULL)
83 q=cur->llink;
84 else
85 {
86 suc=cur->rlink;
87 while(suc->llink!=NULL)
88 suc=suc->llink;
89 suc->llink=cur->llink;
90 q=cur->rlink;
91 }
```

```
91 q=cur->rlink;
92 }
93 if(parent==NULL)
94 return q;
95 if(cur==parent->llink)
96 parent->llink=q;
97 else
98 parent->rlink=q;
99 freenode(cur);
100 return root;
101 }
102 void preorder(NODE root)
103 {
104 if(root!=NULL)
105 {
106 printf("%d\n",root->info);
107 preorder(root->llink);
108 preorder(root->rlink);
109 }
110 }
111 void postorder(NODE root)
112 {
113 if(root!=NULL)
114 {
115 postorder(root->llink);
116 postorder(root->rlink);
117 printf("%d\n",root->info);
118 }
119 }
120 }
121 void inorder(NODE root)
```

```

120     }
121     void inorder(NODE root)
122     {
123         if(root!=NULL)
124         {
125             inorder(root->llink);
126             printf("%d\n",root->info);
127             inorder(root->rlink);
128         }
129     }
130 }
131 void main()
132 {
133     int item,choice;
134     NODE root=NULL;
135     for(;;)
136     {
137         printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
138         printf("enter the choice\n");
139         scanf("%d",&choice);
140         switch(choice)
141         {
142             case 1:printf("enter the item\n");
143                 scanf("%d",&item);
144                 root=insert(root,item);
145                 break;
146             case 2:display(root,0);
147                 break;
148             case 3:preorder(root);
149                 break;
150             case 4:postorder(root);
151                 break;

```

```

152             case 5:inorder(root);
153                 break;
154             case 6:printf("enter the item\n");
155                 scanf("%d",&item);
156                 root=delete(root,item);
157                 break;
158             default:exit(0);
159                 break;
160         }
161     }
162 }
163

```

```

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
50

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

1.insert
2.display
3.pre
4.post
5.in
6.delete

```

```
6.delete
7.exit
enter the choice
1
enter the item
40

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
80

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
90
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
15

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
70

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

```
7.exit
enter the choice
1
enter the item
70

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    90
    80
    70
50
    40
    20
    15

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

```
7.exit
enter the choice
3
50
20
15
40
80
70
90

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
15
20
40
50
70
80
90

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

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
15
20
40
50
70
80
90

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

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

wap to traverse nodes using binary tree

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4 #include<string.h>
5 struct node
6 {
7     int info;
8     struct node *llink;
9     struct node *rlink;
10 };
11 typedef struct node*NODE;
12 NODE getnode()
13 {
14     NODE x;
15     x=(NODE)malloc(sizeof(struct node));
16     if(x==NULL)
17     {
18         printf("memory not available");
19         exit(0);
20     }
21     return x;
22 }
23 void freenode(NODE x)
24 {
25     free(x);
26 }
27 NODE insert(int item,NODE root)
28 {
29     NODE temp,cur,prev;
30     char direction[10];
31     int i;
```

```
31     int i;|
32     temp=getnode();
33     temp->info=item;
34     temp->llink=NULL;
35     temp->rlink=NULL;
36     if(root==NULL)
37     {
38         return temp;
39     }
40     printf("give direction to insert\n");
41     scanf("%s",direction);
42     prev=NULL;
43     cur=root;
44     for(i=0;i<strlen(direction)&&cur!=NULL;i++)
45     {
46         prev=cur;
47         if(direction[i]=='l')
48             cur=cur->llink;
49         else
50             cur=cur->rlink;
51     }
52     if(cur!=NULL|i!=strlen(direction))
53     {
54         printf("insertion not possible\n");
55         freenode(temp);
56         return(root);
57     }
58     if(cur==NULL)
59     {
60         if(direction[i-1]=='l')
61             prev->llink=temp;
62         else
63             prev->rlink=temp;
64     }
```



```

124     }
125     else
126     {
127         printf("given tree is \n");
128         display(root,1);
129         printf("the preorder traversal is \n");
130         preorder(root);
131     }
132     break;
133 case 3:if(root==NULL)
134 {
135     printf("tree is empty");
136 }
137 else
138 {
139     printf("given tree is \n");
140     display(root,1);
141     printf("the inorder traversal is \n");
142     inorder(root);
143 }
144 break;
145 case 4:if (root==NULL)
146 {
147     printf("tree is empty");
148 }
149 else
150 {
151     printf("given tree is \n");
152     display(root,1);
153     printf("the postorder traversal is \n");
154     postorder(root);

```

```

155 }
156 break;
157 case 5:display(root,1);
158 break;
159 default:exit(0);
160 }
161 }
162 }
163

```

```

1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
2
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
3
give direction to insert
r

```

```
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
4
give direction to insert
lr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
5
give direction to insert
rl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
```

```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
given tree is
      3
        7
      5
    1
      4
      6
    2
the preorder traversal is
the item is 1
the item is 2
the item is 4
the item is 6
the item is 3
the item is 5
the item is 7
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
given tree is
```

```
1
enter the item
6
give direction to insert
lrl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
7
give direction to insert
rlr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    3
        7
            5
        1
            4
                6
            2
1.insert
```

```
given tree is
    3
        7
        5
    1
        4
        6
    2
the inorder traversal is
the item is2
the item is6
the item is4
the item is1
the item is5
the item is7
the item is3
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
given tree is
    3
        7
        5
    1
        4
        6
    2
```

```
4.postorder
5.display
enter the choice
4
given tree is
    3
        7
        5
    1
        4
        6
    2
the postorder traversal is
the item is6
the item is4
the item is2
the item is7
the item is5
the item is3
the item is1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
6

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