# 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. _____ | _____ |

INDEX

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

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

# 1)wap to accept student details and print

```c
#include<stdio.h>
struct student
{
    int id,age,marks;
};
int main()
{
    struct student data[100];
    int n;
    printf("enter the number of students:\n");
    scanf("%d",&n);
    int i;
    for(i=0;i<n;i++)
    {
        printf("enter the data of the student %d \n",i+1);
        printf("id: ");
        scanf("%d",&data[i].id);
        printf("enter the age of the student %d \n",i+1);
        printf("age: ");
        scanf("%d",&data[i].age);
        printf("enter marks of the student %d \n",i+1);
        printf("marks: ");
        scanf("%d",&data[i].marks);
    }
    printf("the students who are qualified for admission are:\n");
    for(i=0;i<n;i++)
    {
        if(data[i].age>=20 && data[i].marks<=100)
        printf("student id:%d age:%d marks:%d\n",data[i].id,data[i].age,data[i].marks);
        }
        return 0;
}
```

```
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

```c
#include<stdio.h>
#include<stdlib.h>
# define size 5
int top=-1;
void push(int stack[],int item)
{
    if(top==size-1)
    printf("STACK OVERFLOW \n");
    else
    {
        top++;
        stack[top]=item;
    }
}
int pop(int stack[])
{
    if(top==-1){
    printf("STACK UNDERFLOW\n");
    return -1;

    }
    else
    {
        int item=stack[top];
        top--;
        return item;
    }
}
void display(int stack[])
{

```

```c
void display(int stack[])
{

    printf("contents of the stack \n");
    for(int i=top;i>=0;i--)
    printf("%d\n",stack[i]);
}
int main()
{
    int stack[size];
    int choice,element;
    for(;;)
    {
        printf("\n1.push\n2.pop\n3.print\n4.exit\n");
        printf("enter your choice\n");
        scanf("%d",&choice);
        switch(choice){
            case 1:
            printf("enter element\n");
            scanf("%d",&element);
            push(stack,element);
            break;
            case 2:
            element=pop(stack);
            if(element==-1)
            printf("STACK UNDERFLOW\n");
            else
            printf("element popped :%d",element);
            break;
            case 3:
            display(stack);
```

```c
            display(stack);
            break;
            default :
            printf("end of operation\n");
            exit(0);

        }
    }
}
```

```
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

```c
#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 '(':return 0;
    case '#':return -1;
    default:return 8;
    }

}
int G(char symbol)
{
    switch(symbol)
    {
    case '+':
    case '-':return 1;
    case '*':
    case '/':return 3;
    case '^':
    case '$':return 6;
    case '(':return 9;
    case '#':return 0;
    default:return 7;
    }
}
void infix_postfix(char infix[],char postfix[])
{
    int top,i,j;
    char s[30],symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(F(s[top])>G(symbol))
        {
            postfix[j]=s[top--];
            j++;
        }
        if(F(s[top])!=G(symbol))
        s[++top]=symbol;
        else
        top--;
        }
        while(s[top]!='#')
        {
            postfix[j++]=s[top--];
        }
          postfix[j]='\0';
    }
         void main()
```

```
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-ef/gh)+(/+


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

# wap to check if a word is palindrome or not

```c
#include<stdio.h>
#include<string.h>
int top=-1;
char stack[30];
void push(char b)
{
    top++;
    stack[top]=b;
}
char pop(char word[])
{
    return word[top--];

}
int main()
{
    char word[30];
    char newword[30];
    int flag=0;
    printf("enter the string \n");
    scanf("%s",word);
    int size=strlen(word);
    for(int i=0;i<size;i++)
    push(word[i]);
    for(int i=0;i<size;i++)
    newword[i]=pop(word);
    for(int i=0;i<size;i++)
    {
        if(stack[i]==newword[i])
        continue;
        else{
            flag=1;
            break;
        }
    }
    if(flag==1)
    printf("NON PALINDROME \n");
    else
    printf("PALINDROME \n");
}
```

```
input
enter the string
madam
PALINDROME


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

## Tower of hannoi:

```c
#include<stdio.h>
void tower(int,char,char,char);
void tower(int n,char src,char temp,char dest)
{
    if(n==1)
    {
        printf("move disc %d from %c to %c \n",n,src,dest);
        return;
    }
    tower(n-1,src,dest,temp);
    printf("move disc %d from %c to %c \n",n,src,dest);
    tower(n-1,temp,src,dest);
    return;
}
int main()
{
    int element;
    printf("enter the number of discs: \n");
    scanf("%d",&element);
    tower(element,'s','t','d');

}
```

```
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:
        \n", n);
 for ( i=0; i<n; i++) {
   printf (" fib(%d) = %d\n", i, fib(i));
 }
}
```

```
Enter n
9
9 fibonacci numbers are:
fib(0)=0
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
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]
```

Date: / /

## 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\n");
for(i=0; i<size; i++)
{
scanf("%d", &list[i]);
}
bubble_sort(list, size);
printf("\n");
printf("enter key to search\n");
scanf("%d", &key);
binary_search(list, 0, size, key);
}

void bubble_sort(int list[], int siz)
{
  int temp, i, j;
  for(i=0; i<siz; i++)
  {
    for(j=1; j<size; j++)
    {
    if(list[i] > list[j])
      temp = list[i];
```

```
list[i] = list[j];
list[j] = temp;
}
}
}
void binary_search (int list[], int lo, int hi
                    int key)
{
  int mid;
  if (lo > hi)
  {
printf ("key not found \n");
  return;
}
  mid = (lo + hi) / 2;
if (list[mid] == key)
  {
  printf ("key found \n");
  else if (list[mid > key)
  {
  binary_search (list, lo, mid-1, key)
  }
  else if (dist[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

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

```c
 void displayQ()
 {
     int i;
     if(front>rear)
     printf("queue empty\n");
     else
     for(i=front;i<=rear;i++)
     {
         printf("%d\n",q[i]);
     }
 }
 void main()
 {
     int choice;

     for(;;)
     {
         printf("\n1.insertrear\n2.deletefront\n3.display\n4.exit\n");
         printf("enter the choice\n");
         scanf("%d",&choice);
         switch(choice)
         {
             case 1: printf("enter the item to be inserted\n");
                     scanf("%d",&item);
                     insert_rear();
                     break;
             case 2:
                     item=delete_front();
                     if(item==-1)
```

```c
                     if(item==-1)
                     printf("empty queue\n");
                     else
                     printf("element deleted: %d",item);
                     break;
             case 3:
                     displayQ();
                     break;
             default:
                     printf("end of operation\n");


         }
     }

 }
```

```
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
```

```
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
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
```

```c
#include<stdio.h>
#include<stdlib.h>
# define q_size 5
int item,front=0,rear=-1,q[q_size],count=0;
void insert_rear()
{
    if(count==q_size)
    {
    printf("queue underflow\n");
    return;
    }
    rear=(rear+1)%q_size;
    q[rear]=item;
    count++;
}
int delete_front()
{
    if(count==0)
    return -1;
    item=q[front];
    front=(front+1)%q_size;
    count=count-1;
    return item;
}
void displayQ()
{
    int i,f;
    if(count==0)
    {
        printf("queue is empty\n");
        return;
```

```
    }
    f=front;
    printf("contents of queue\n");
    for(i=1;i<=count;i++)
    {
        printf("%d\n",q[f]);
        f=(f+1)%q_size;
    }
}
void main()
{
    int choice;

    for(;;)
    {
        printf("\n1.insertrear\n2.deletefront\n3.display\n4.exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("enter the item to be inserted\n");
                    scanf("%d",&item);
                    insert_rear();
                    break;
            case 2: item=delete_front();
                    if(item==-1)
                    printf("\n empty queue");
                    else
                    printf("item deleted=%d\n",item);
                    break;
            case 3:
                    displayQ();
```

```
            case 3:
                    displayQ();
                    break;
            default:
                    printf("enf of operation");
                    exit(0);


        }
    }

}
```

```
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:

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define N 3
void pqinsert(int);
void pqdelete();
void display();
int queue[3][N];
int front[3]={0,0,0};
int rear[3]={-1,-1,-1};
int item,pr;
int main()
{
int ch;
while(1)
{
printf("\nPRIORITY QUEUE\n");
printf("*******\n");
printf("\n\t1:PQinsert\n");
printf("\n\t2:PQdelete\n");
printf("\n\t3:PQdisplay\n");
printf("\n\t4:Exit\n");
printf("\nenter the choice\n");
scanf("%d",&ch);
switch(ch)
{
 case 1:printf("\n enter the priority number\n");
        scanf("%d",&pr);
        if(pr>0 && pr<4)
        pqinsert(pr-1);
        pqinsert(pr-1);
        else
        printf("\n only 3 priority exists 1 2 3\n");
        break;
case 2:pqdelete();
       break;
case 3:display();
       break;
case 4:exit(0);
}
}
getch();
}
 void pqinsert(int pr)
{
 if(rear[pr]==N-1)
 printf("\n Queue overflow\n");
 else
 {
 printf("\n enter the item\n");
 scanf("%d",&item);
 rear[pr]++;
 queue[pr][rear[pr]]=item;
 }

}
void pqdelete()
{
int i;
for(i=0;i<3;i++)
```

```
int i;
for(i=0;i<3;i++)
{
  if(rear[i]==front[i]-1)
  printf("\n queue empty\n");
  else
  {
  printf("deleted item is %d of queue %d\n",queue[i][front[i]],i+1);
  front[i]++;

  }
 }
}
void display()
{
int i,j;
for(i=0;i<3;i++)
{
if(rear[i]==front[i]-1)
  printf("\n queue empty %d\n",i+1);
else
  {
  printf("\nQUEUE %d:",i+1);
  for(j=front[i];j<=rear[i];j++)
    printf("%d\t",queue[i][j]);
  }
}

}
```

```
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:***

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int priority;
        int info;
        struct node *link;
}*front=NULL;
void insert(int item, int item_priority);
int del();
void display();
int isEmpty();
int main()
{
        int choice,item,item_priority;
        while(1)
        {
                printf("\n1.Insert\n");
                printf("2.Delete\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("\nEnter your choice : ");
                scanf("%d", &choice);
                switch(choice)
                {
                 case 1:
                        printf("\nInput the item to be added in the queue : ");
                        scanf("%d",&item);
                        printf("\nEnter its priority : ");
                        scanf("%d",&item_priority);
                        insert(item, item_priority);
                        break;
                 case 2:
                        printf("\nDeleted item is %d\n",del());
                        break;
                 case 3:
                        display();
                        break;
                 case 4:
                        exit(1);
                 default :
                        printf("\nWrong choice\n");
                }
        }

        return 0;
}
void insert(int item,int item_priority)
{
        struct node *tmp,*p;

        tmp=(struct node *)malloc(sizeof(struct node));
        if(tmp==NULL)
        {
                printf("\nMemory not available\n");
                return;
        }
        tmp->info=item;
        tmp->priority=item_priority;

        if( isEmpty() || item_priority < front->priority )
```

```
        if( isEmpty() || item_priority < front->priority )
        {
                tmp->link=front;
                front=tmp;
        }
        else
        {
                p = front;
                while( p->link!=NULL && p->link->priority<=item_priority )
                        p=p->link;
                tmp->link=p->link;
                p->link=tmp;
        }
}
int del()
{
        struct node *tmp;
        int item;
        if( isEmpty() )
        {
                printf("\nQueue Underflow\n");
                exit(1);
        }
        else
        {
                tmp=front;
                item=tmp->info;
                front=front->link;
                free(tmp);
        }
        return item;
}
```

```
int isEmpty()
{
        if( front == NULL )
                return 1;
        else
                return 0;
}
void display()
{
        struct node *ptr;
        ptr=front;
        if( isEmpty() )
                printf("\nQueue is empty\n");
        else
        {       printf("\nQueue is :\n");
                printf("\nPriority        Item\n");
                while(ptr!=NULL)
                {
                        printf("%5d         %5d\n",ptr->priority,ptr->info);
                        ptr=ptr->link;
                }
        }
}
```

```
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:

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define qsize 5
int f=0,r=-1,ch;
int item,q[10];

int isfull()
  {
   return(r==qsize-1)?1:0;
  }
int isempty()
  {
   return(f>r)?1:0;
  }
void insert_rear()
  {
   if(isfull())
{
 printf("queue overflow\n");
 return;
}
   r=r+1;
   q[r]=item;
  }
void delete_front()
  {
   if(isempty())
{
 printf("queue empty\n");
```

```c
 printf("queue empty\n");
 return;
}
   printf("item deleted is %d\n",q[(f)++]);
   if(f>r)
{
 f=0;
 r=-1;
}
  }
void insert_front()
  {
   if(f!=0)
{
 f=f-1;
 q[f]=item;
 return;
}
else if((f==0)&&(r==-1))
{
 q[++(r)]=item;
 return;
}
else
 printf("insertion not possible\n");
  }
void delete_rear()
  {
   if(isempty())
{
 printf("queue is empty\n");
```

```
 printf("queue is empty\n");
 return;
}
   printf("item deleted is %d\n",q[(r)--]);
   if(f>r)
{
 f=0;
 r=-1;
}
  }
void display()
  {
   int i;
   if(isempty())
{
 printf("queue empty\n");
 return;
}
   for(i=f;i<=r;i++)
printf("%d\n",q[i]);
  }
void main()
  {for(;;)
   {
printf("1.insert_rear\n2.insert_front\n3.delete_rear\n4.delete_front\n5.display\n6.exit\n");
printf("enter choice\n");
scanf("%d",&ch);
switch(ch)
 {
  case 1:printf("enter the item\n");
 scanf("%d",&item);
```

```
  case 1:printf("enter the item\n");
 scanf("%d",&item);
 insert_rear();
 break;
  case 2:printf("enter the item\n");
 scanf("%d",&item);
 insert_front();
 break;
  case 3:delete_rear();
 break;
  case 4:delete_front();
 break;
  case 5:display();
 break;
  default:exit(0);
 }}}
```

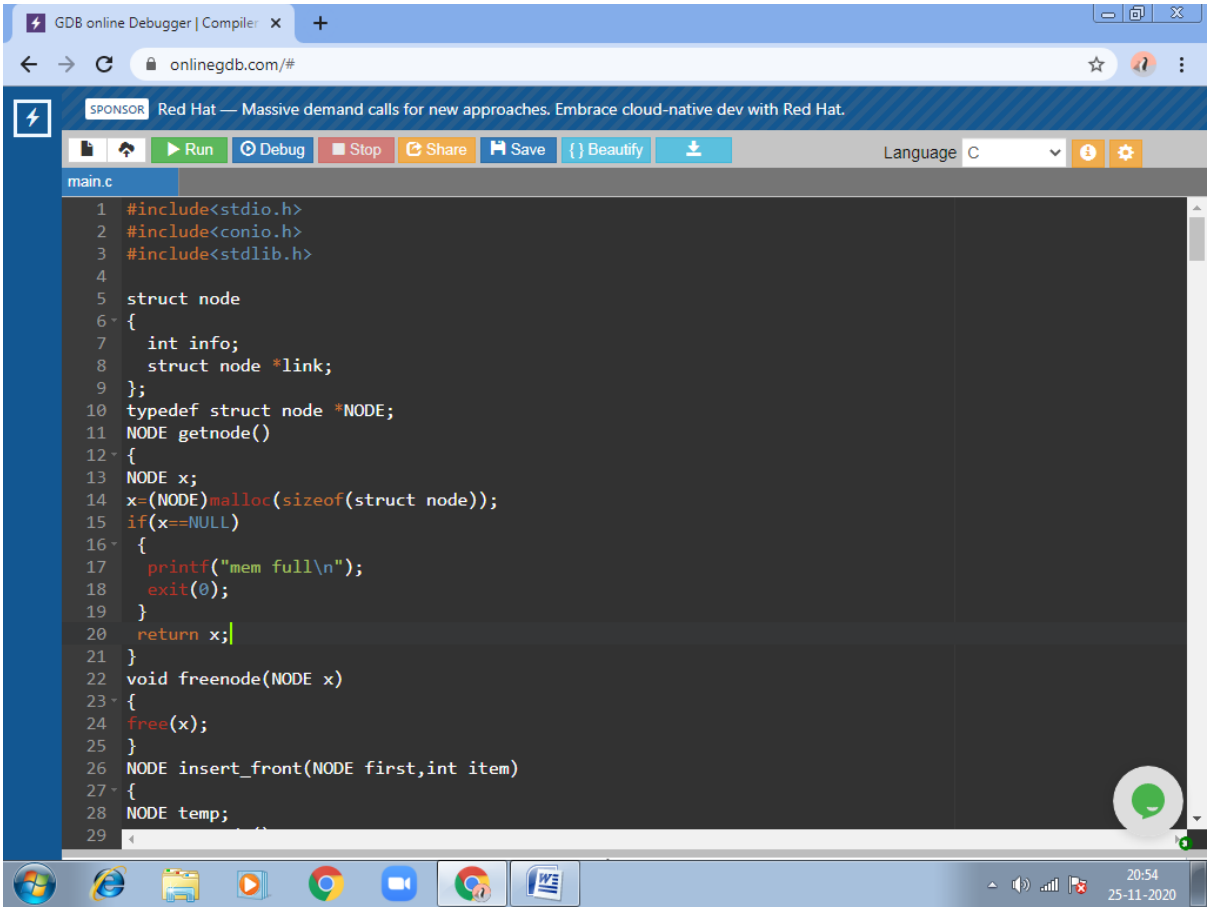
```
enter the item
6
insertion not possible
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
1
enter the item
9
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
5
2
9
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front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

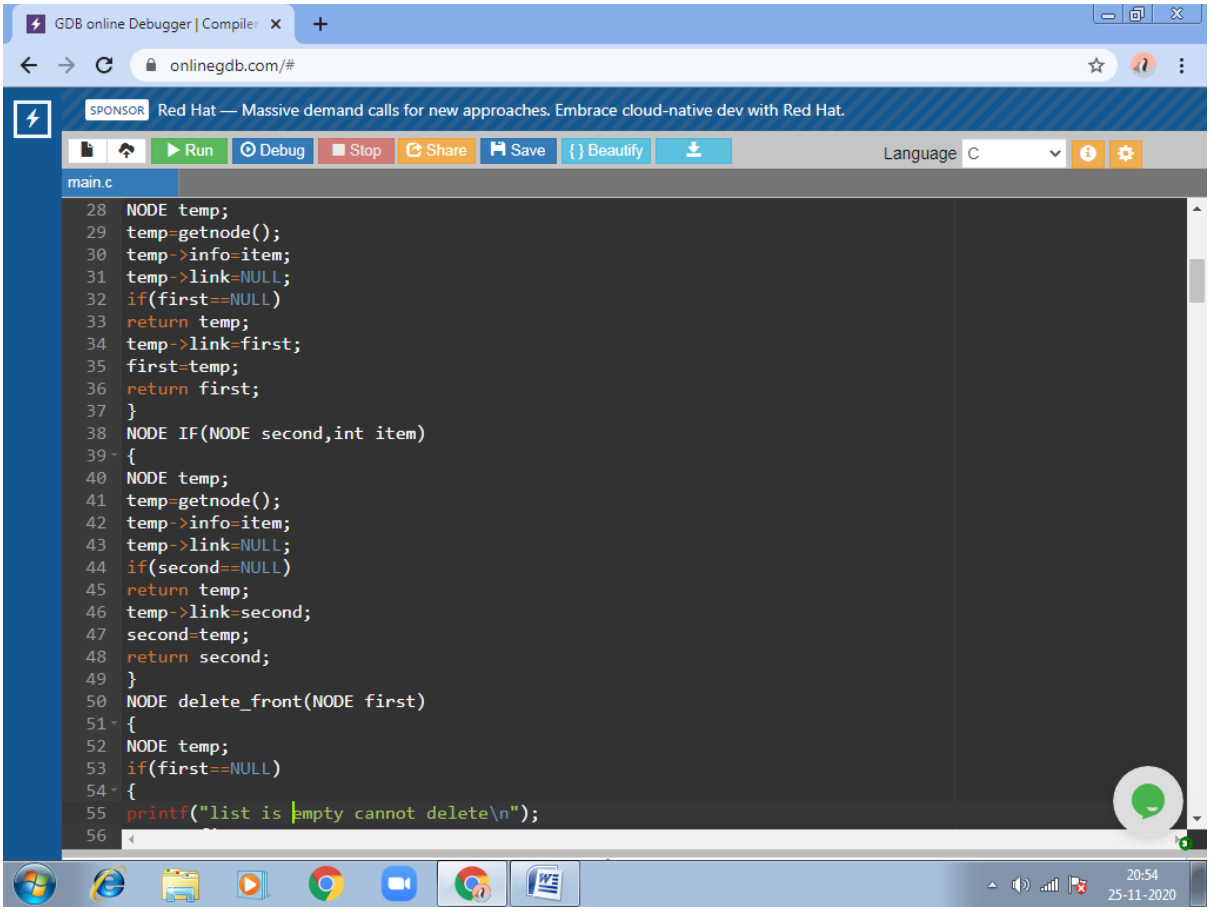
```
enter choice
4
item deleted is 2
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
4
item deleted is 9
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
```

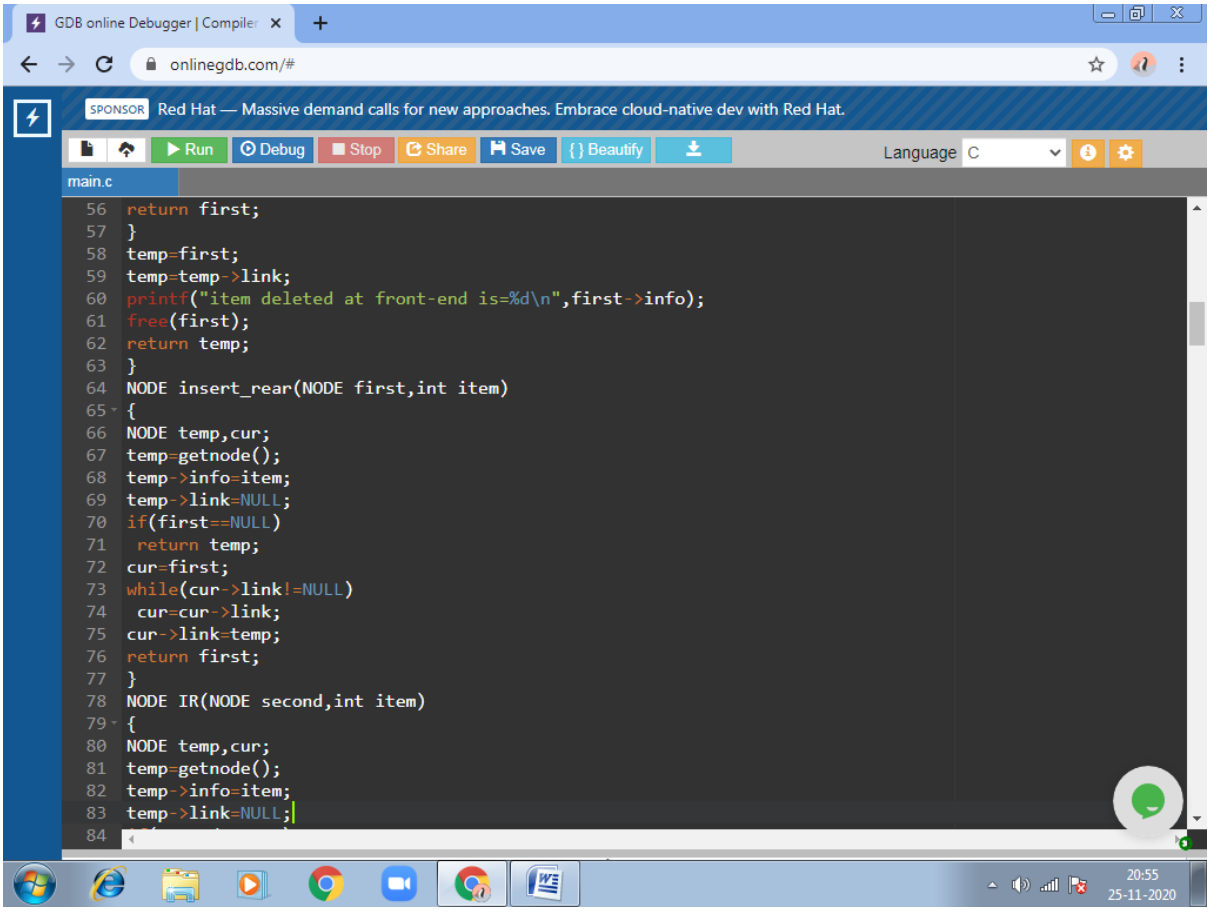
```
enter choice
5
queue empty
1.insert_rear
2.insert_front
3.delete_rear
4.delete_front
5.display
6.exit
enter choice
6


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

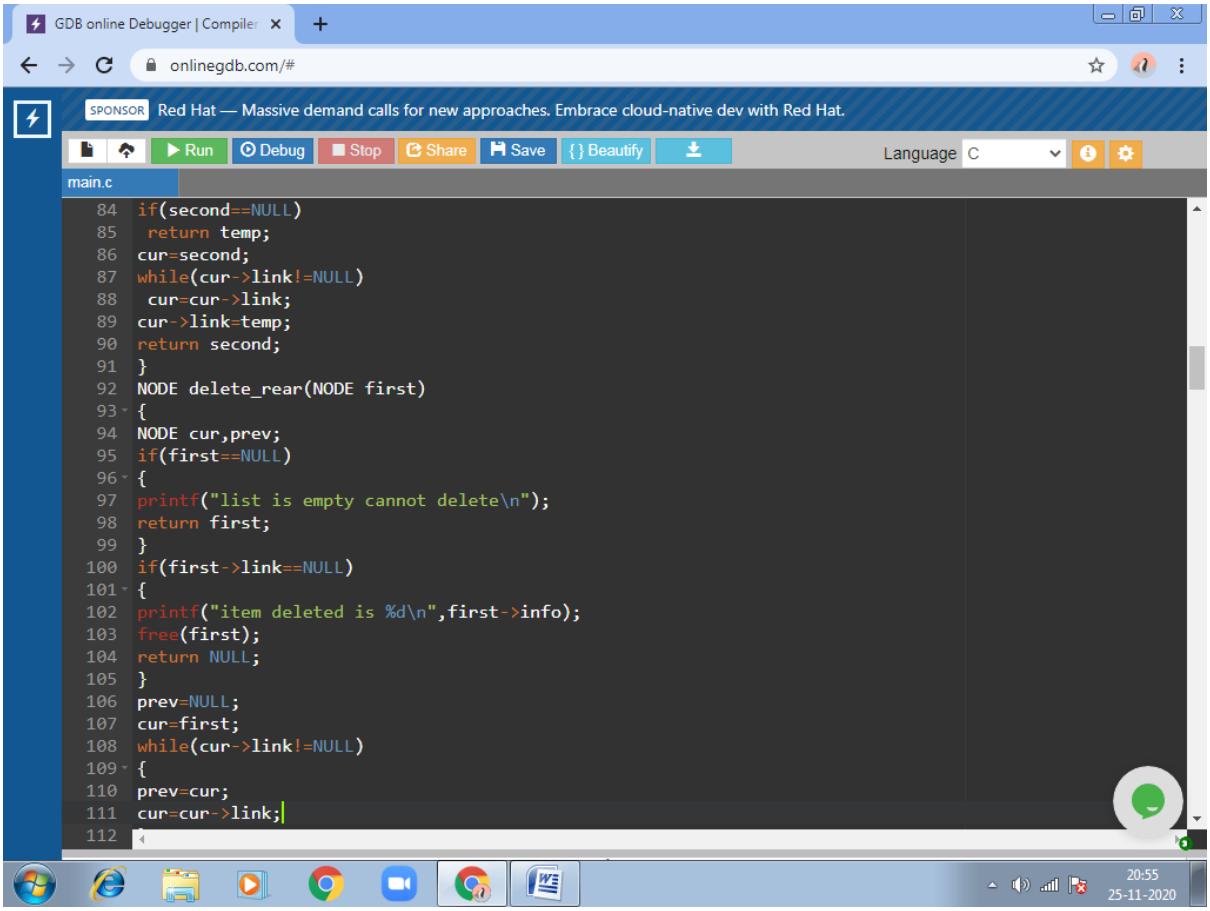
# wap to implement linked list and its various operations

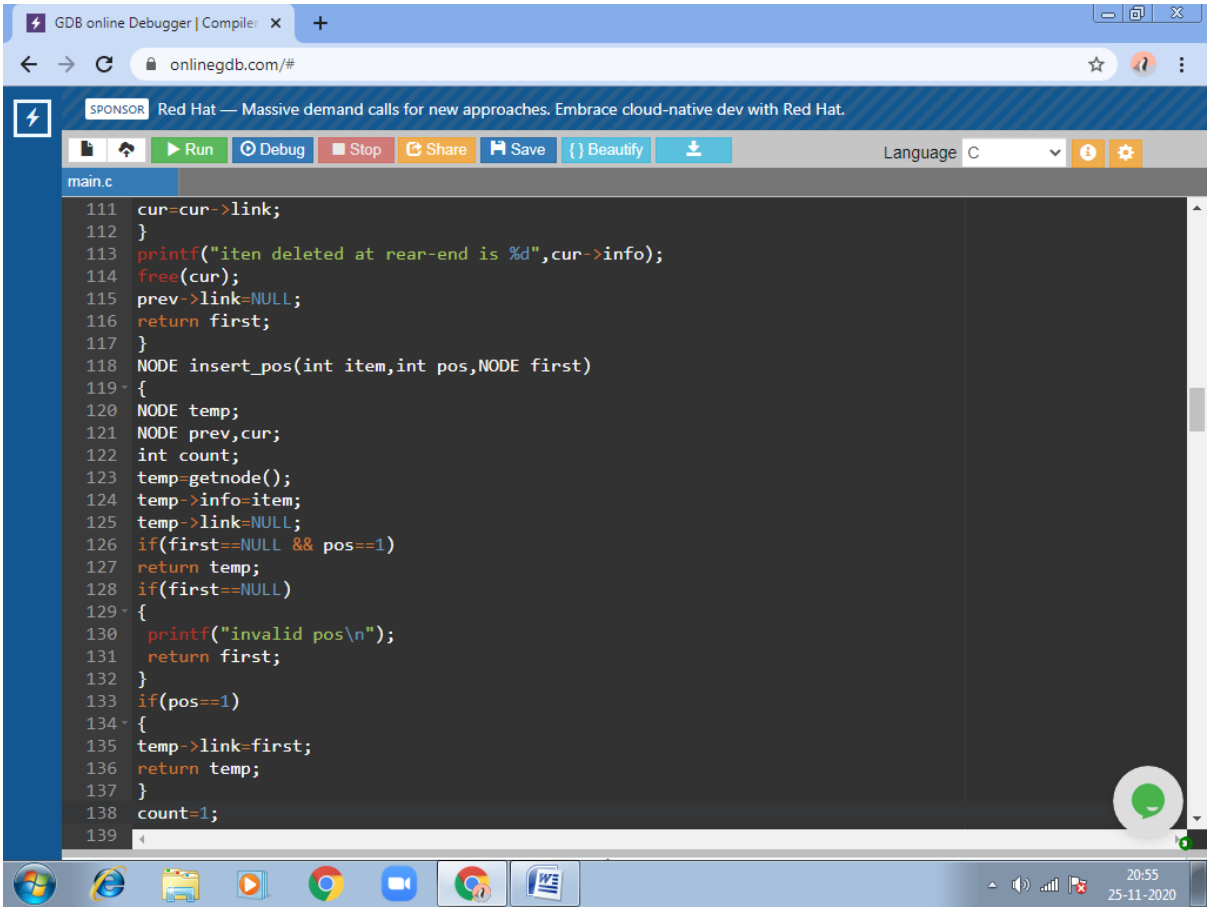
CODE:

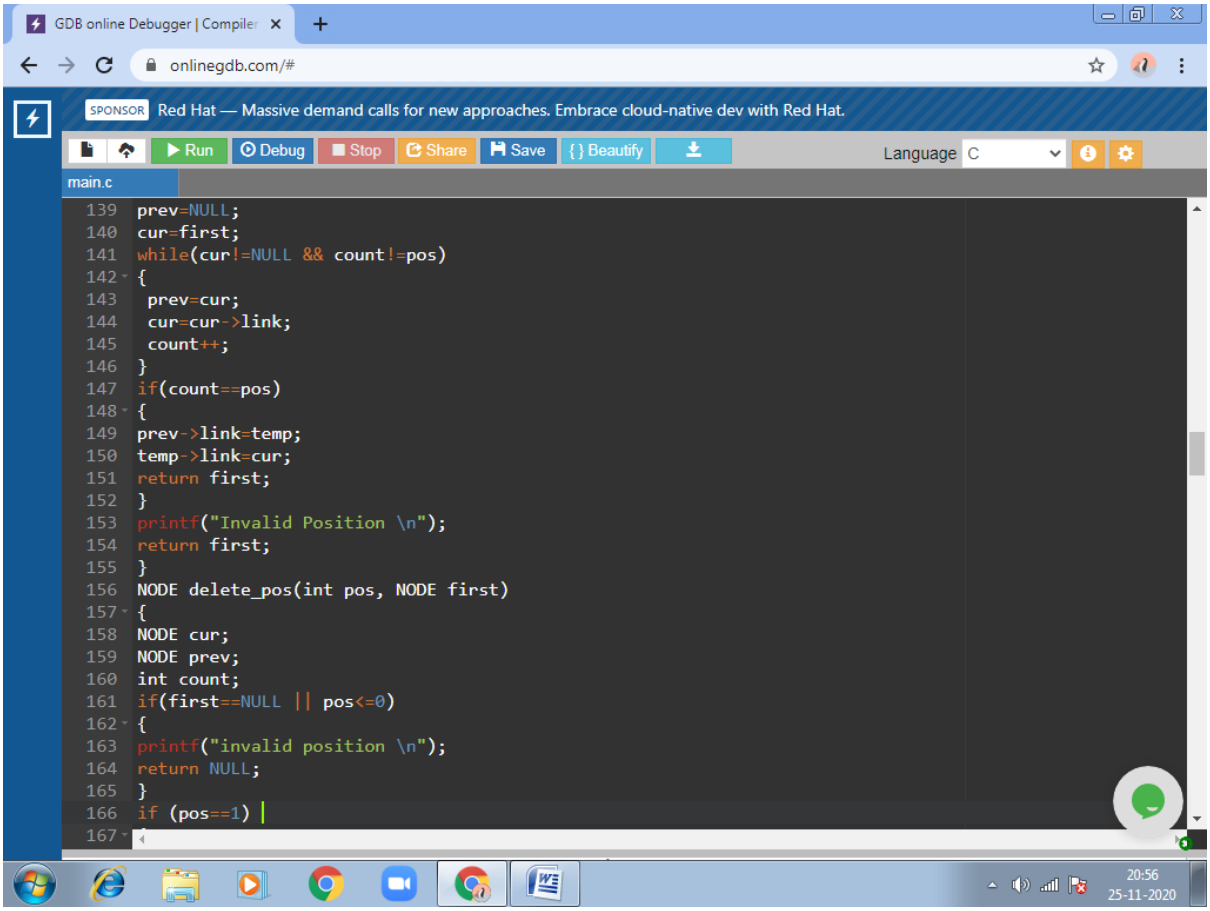


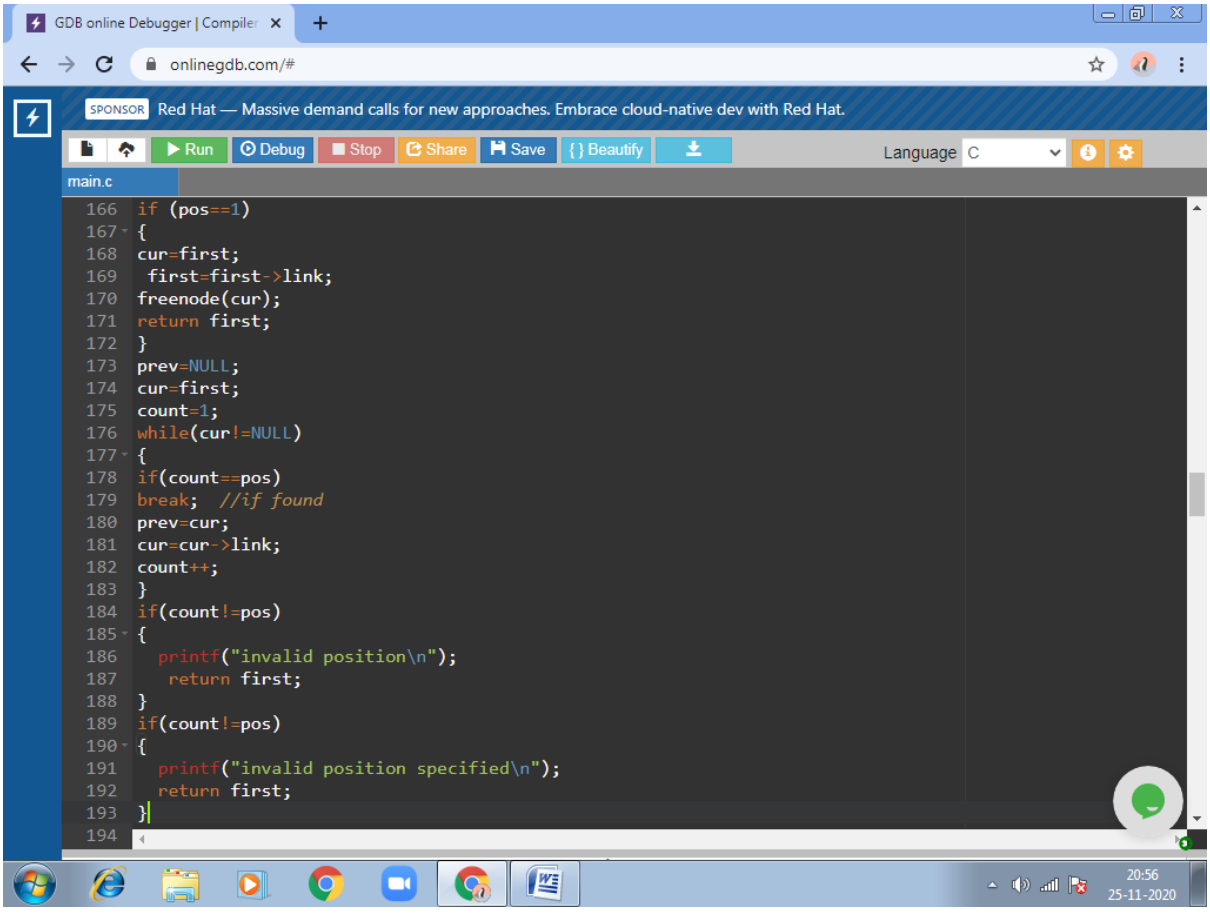

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE IF(NODE second,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(second==NULL)
return temp;
temp->link=second;
second=temp;
return second;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
20:54
25-11-2020

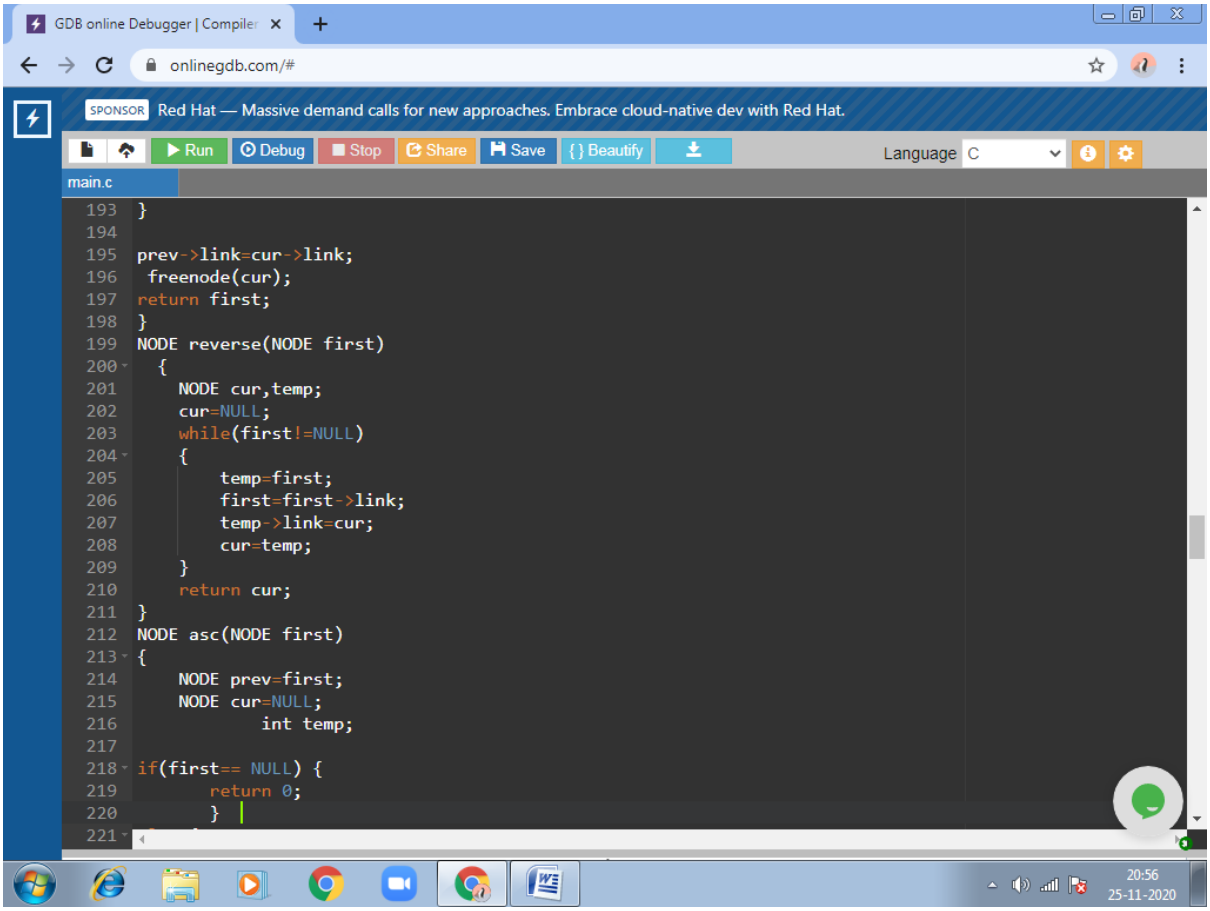

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE IR(NODE second,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
20:55
25-11-2020

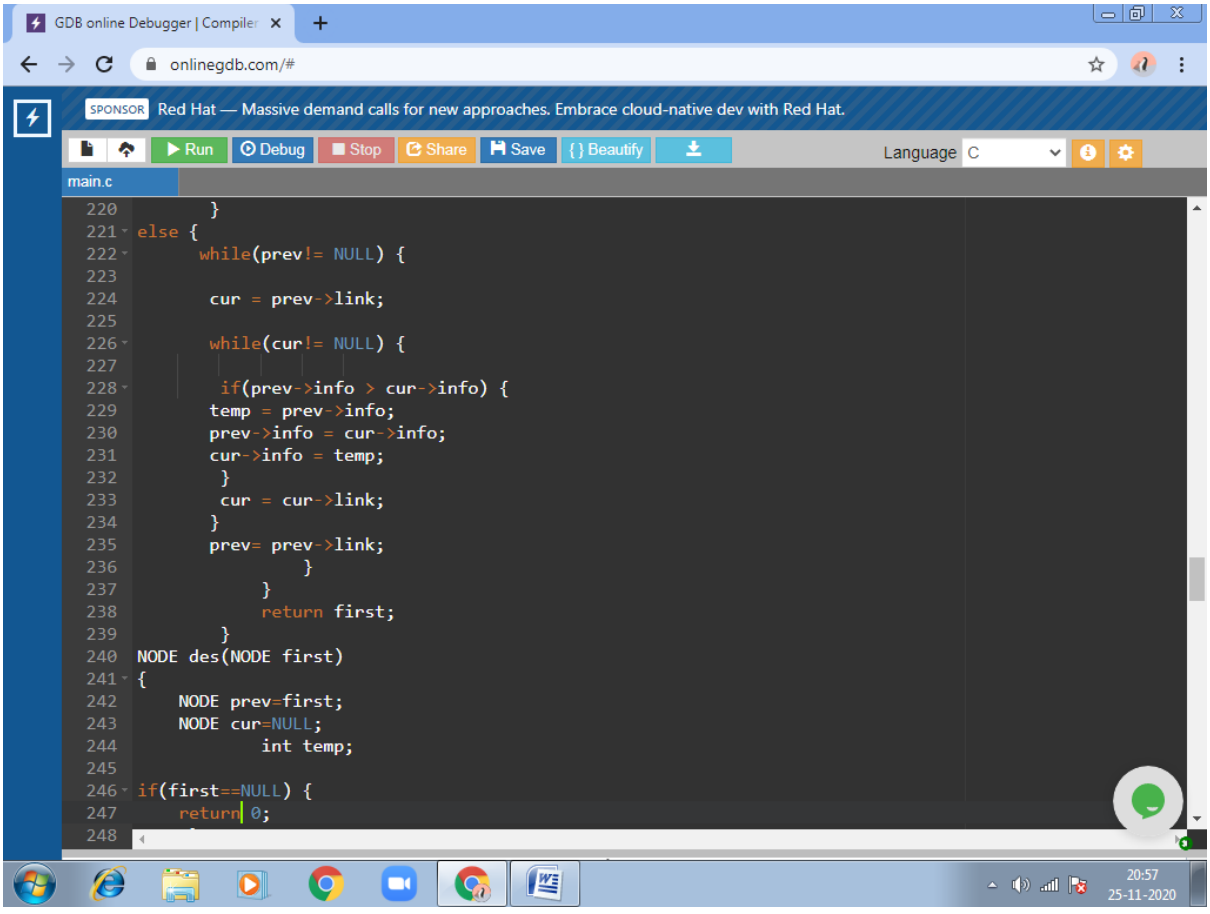

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
84 if(second==NULL)
85 return temp;
86 cur=second;
87 while(cur->link!=NULL)
88 cur=cur->link;
89 cur->link=temp;
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
20:55
25-11-2020

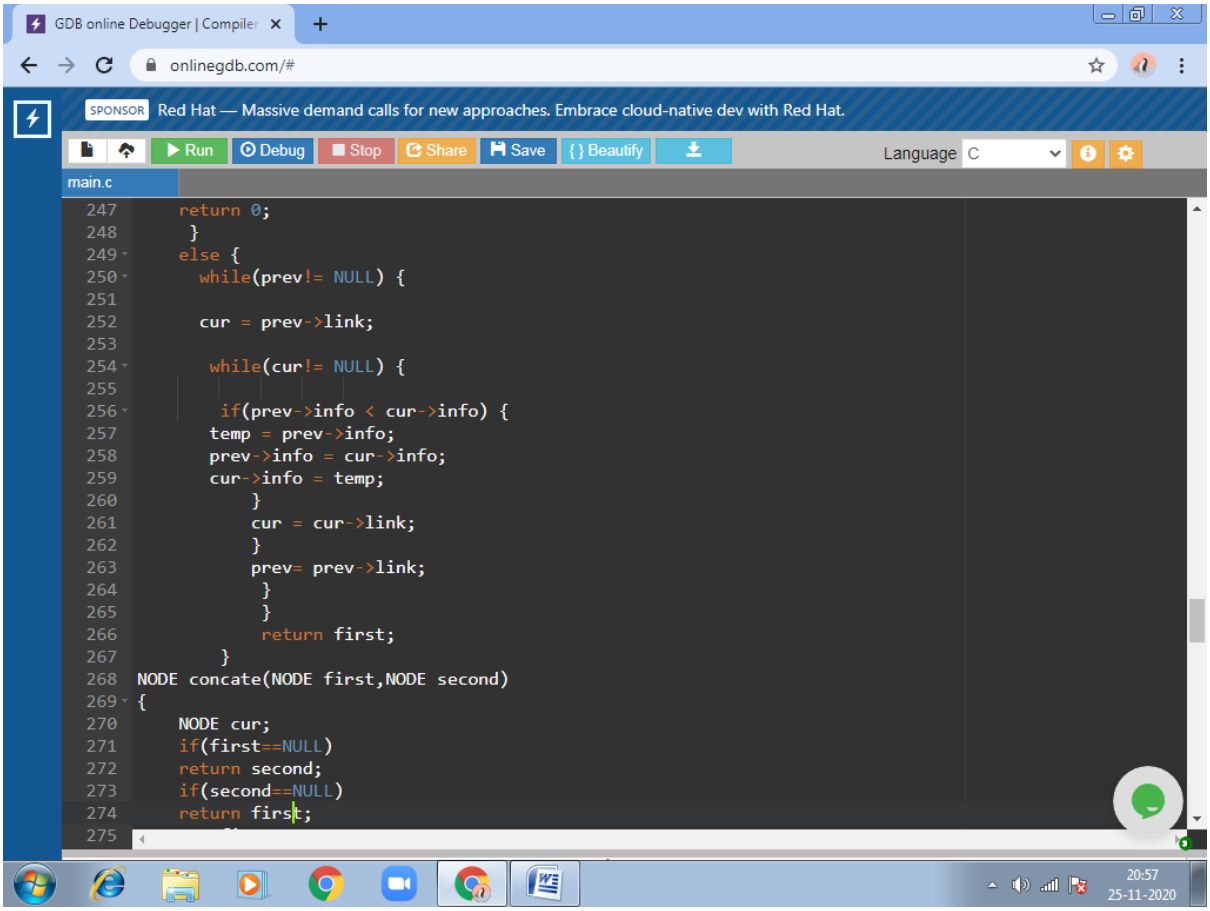

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
111 cur=cur->link;
112 }
113 printf("iten deleted at rear-end is %d",cur->info);
114 free(cur);
115 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
20:55
25-11-2020

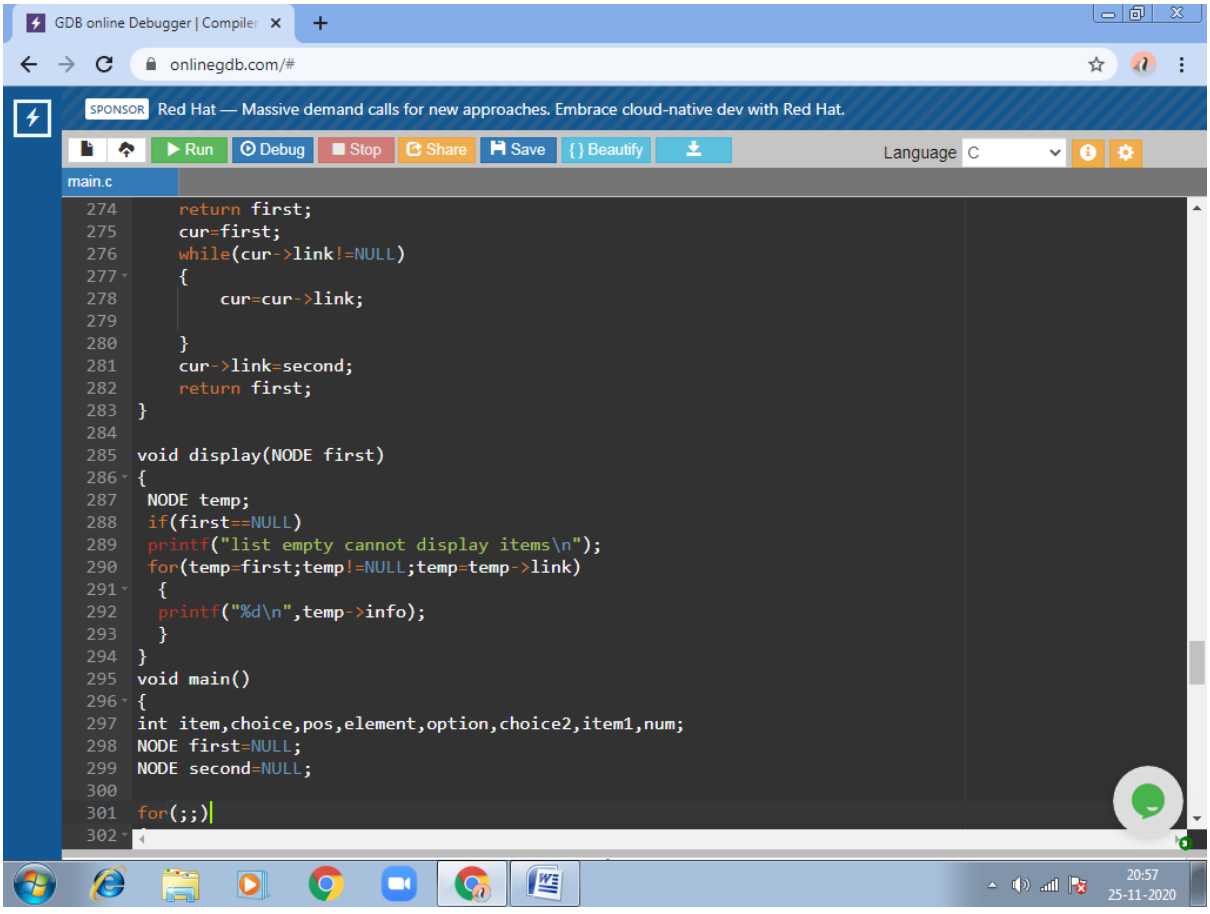

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid Position \n");
return first;
}
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)
20:56
25-11-2020

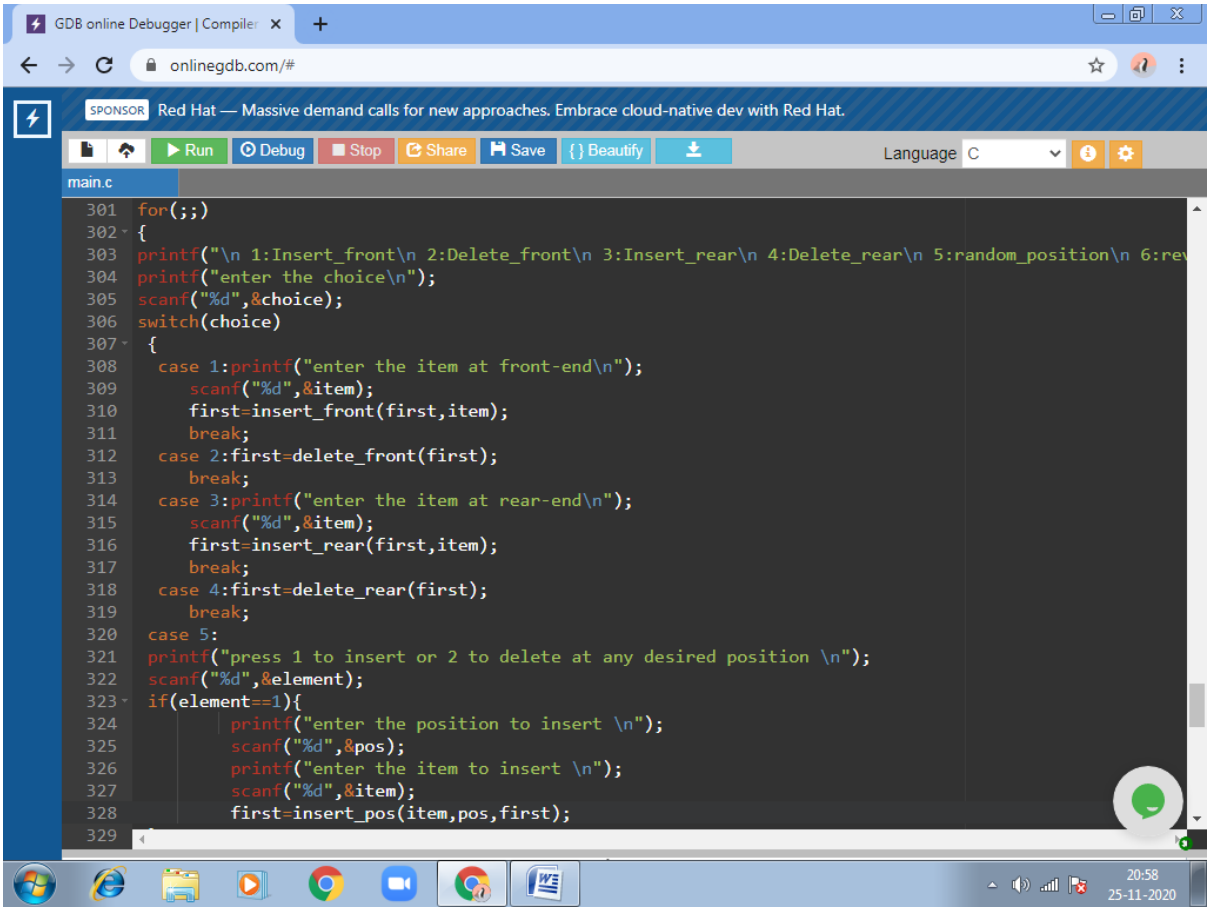

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
if (pos==1)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
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;
}
20:56
25-11-2020

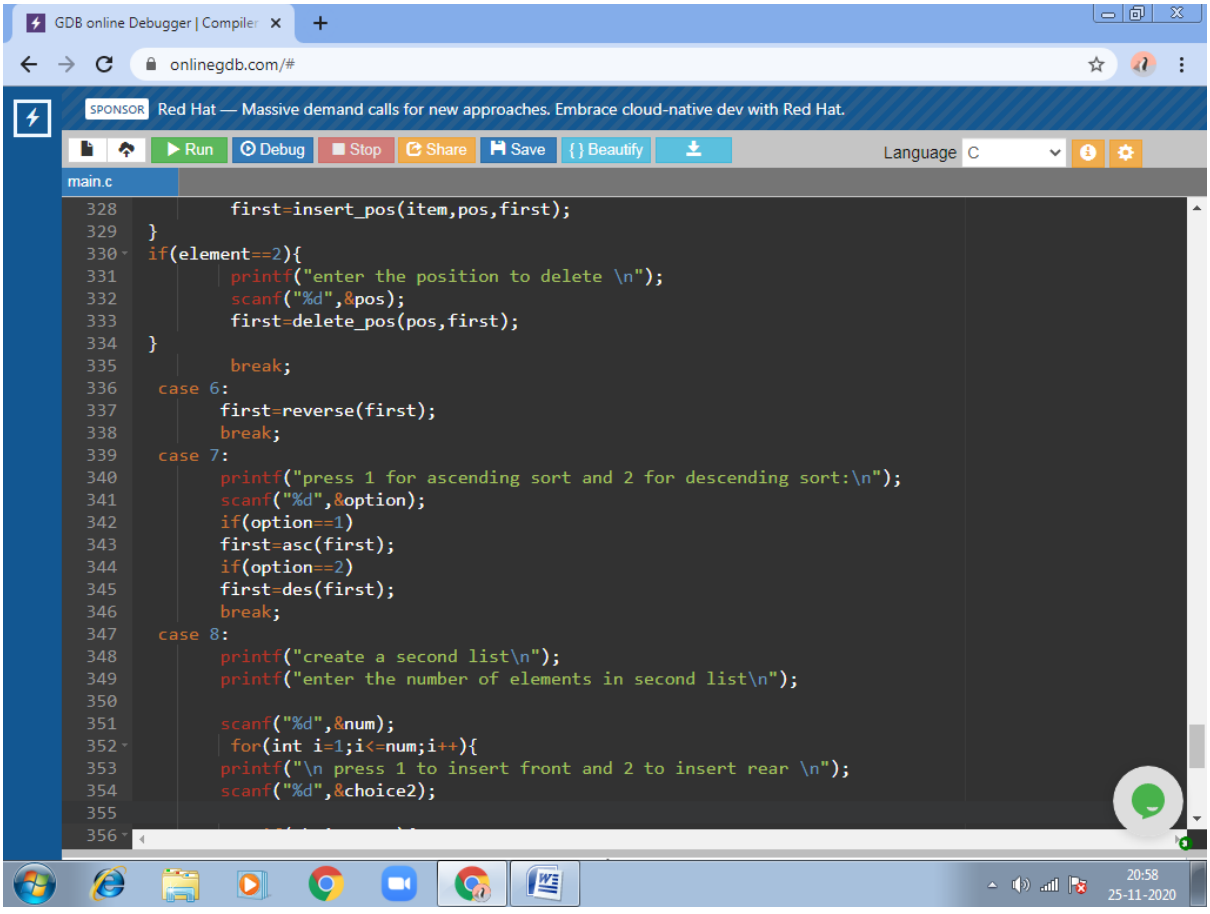

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
}
prev->link=cur->link;
freenode(cur);
return first;
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
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;
}
20:56
25-11-2020

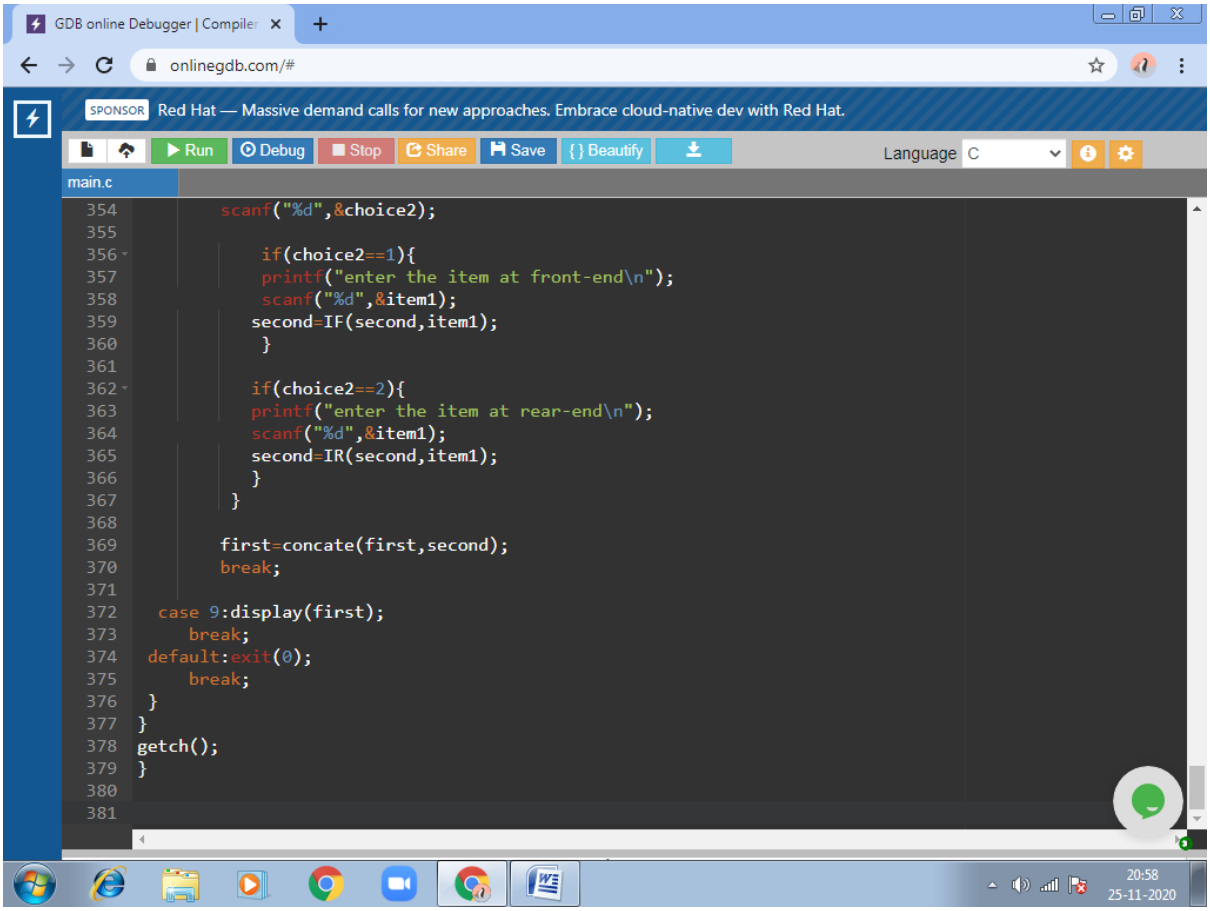

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
}
else {
while(prev!= NULL) {
cur = prev->link;
while(cur!= NULL) {
if(prev->info > cur->info) {
temp = 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;
20:57
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
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 concate(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
20:57
25-11-2020

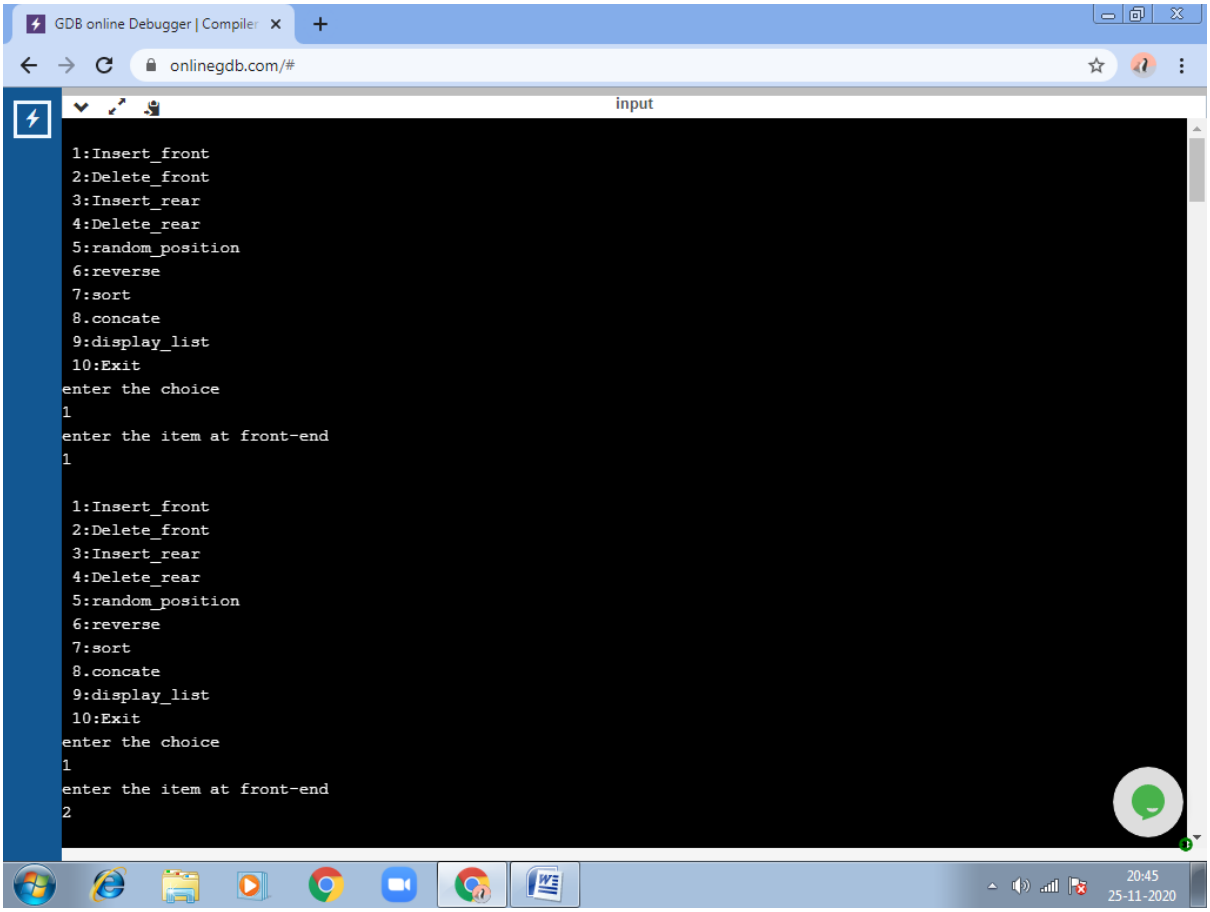

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
return first;
cur=first;
while(cur->link!=NULL)
{
cur=cur->link;
}
cur->link=second;
return first;
}
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);
}
}
void main()
{
int item,choice,pos,element,option,choice2,item1,num;
NODE first=NULL;
NODE second=NULL;
for(;;)
20:57
25-11-2020

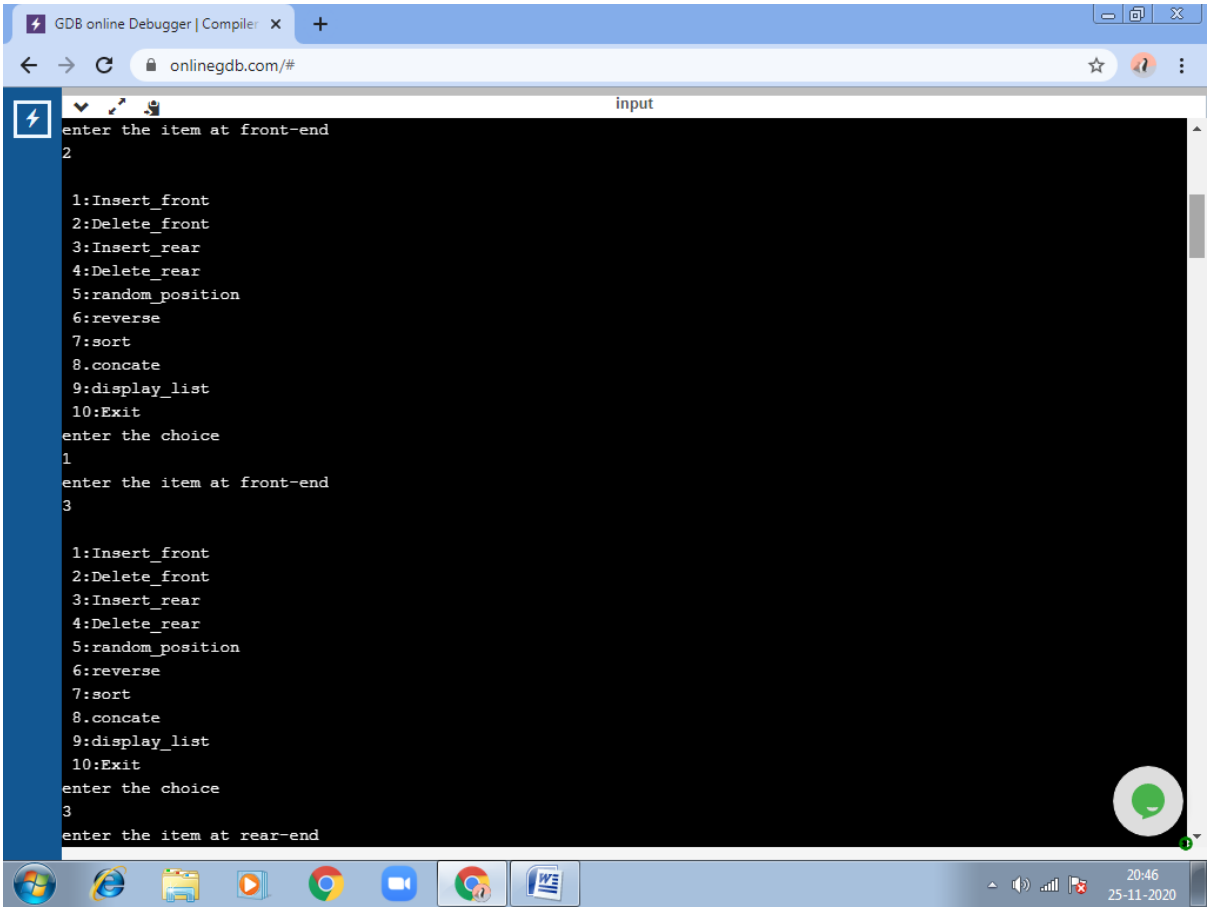

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:rev
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");
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);
20:58
25-11-2020

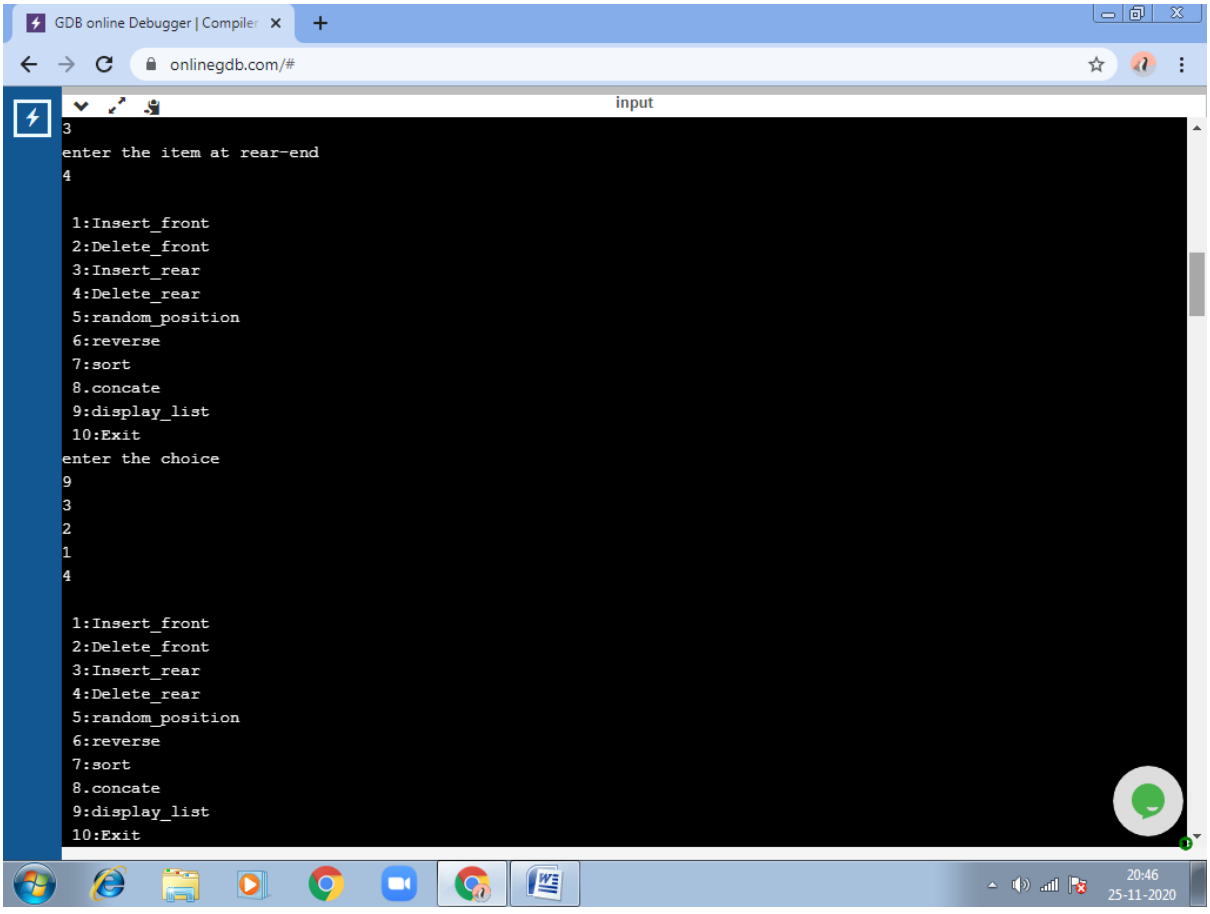

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
first=insert_pos(item,pos,first);
}
if(element==2){
printf("enter the position to delete \n");
scanf("%d",&pos);
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=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("\n press 1 to insert front and 2 to insert rear \n");
scanf("%d",&choice2);
20:58
25-11-2020

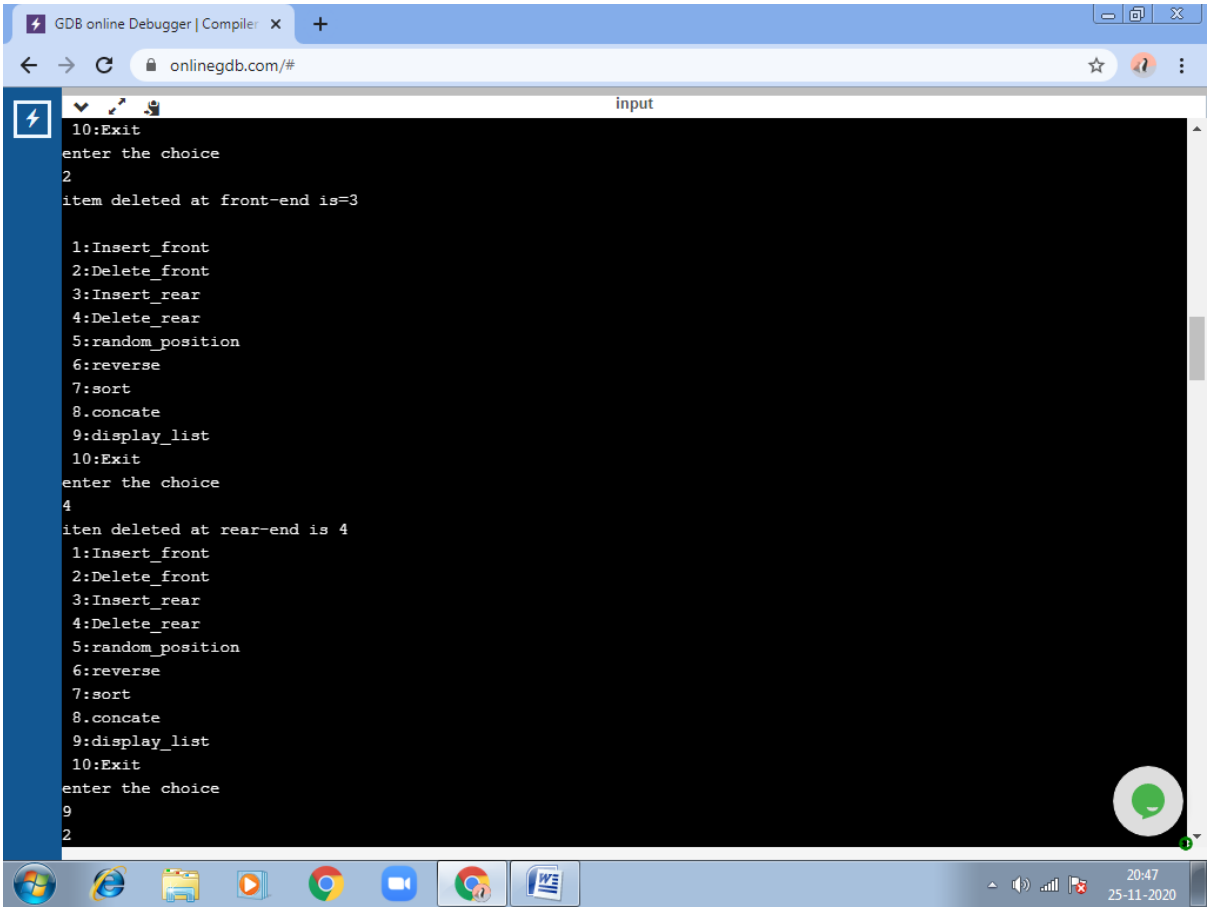

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
scanf("%d",&choice2);
if(choice2==1){
printf("enter the item at front-end\n");
scanf("%d",&item1);
second=IF(second,item1);
}
if(choice2==2){
printf("enter the item at rear-end\n");
scanf("%d",&item1);
second=IR(second,item1);
}
}
first=concate(first,second);
break;
case 9:display(first);
break;
default:exit(0);
break;
}
}
getch();
}
20:58
25-11-2020

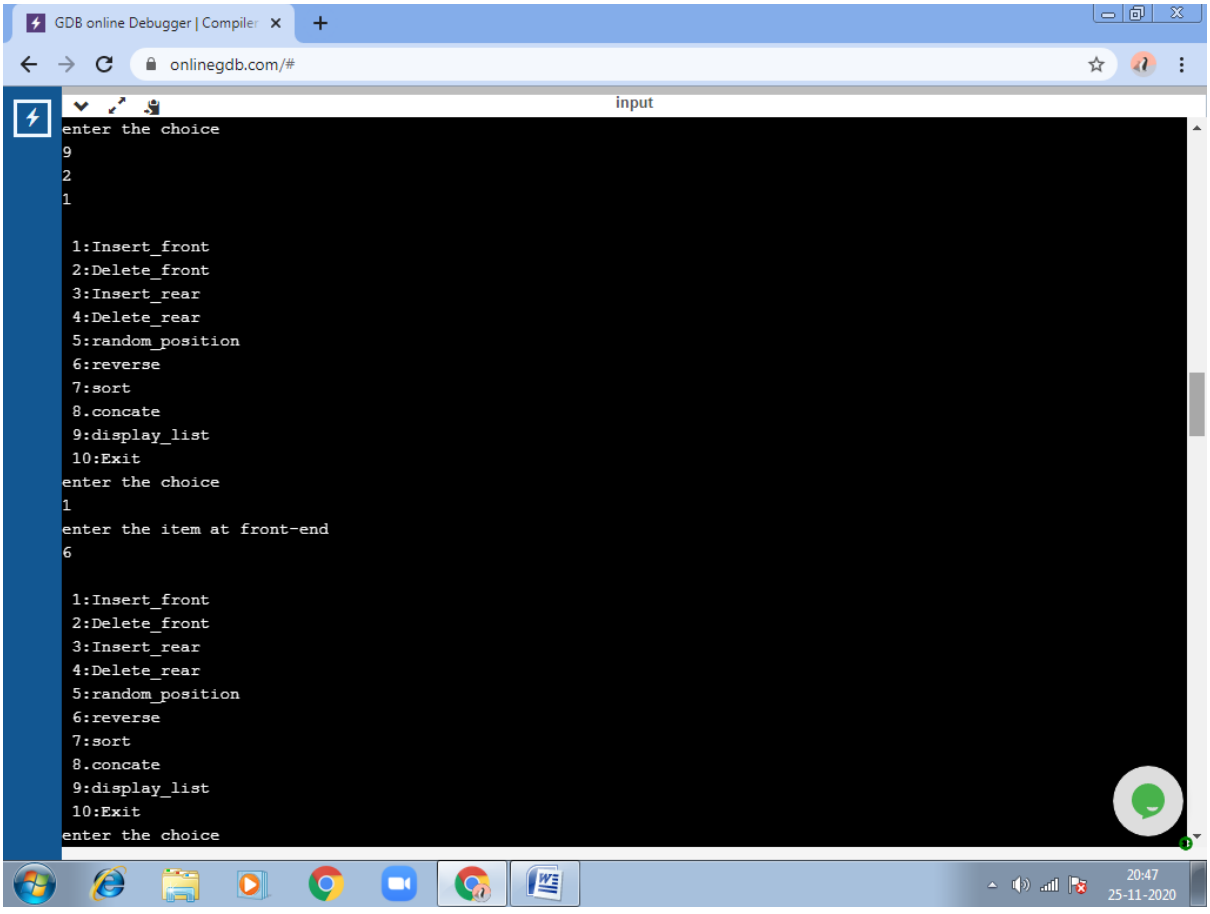
**OUTPUT:**

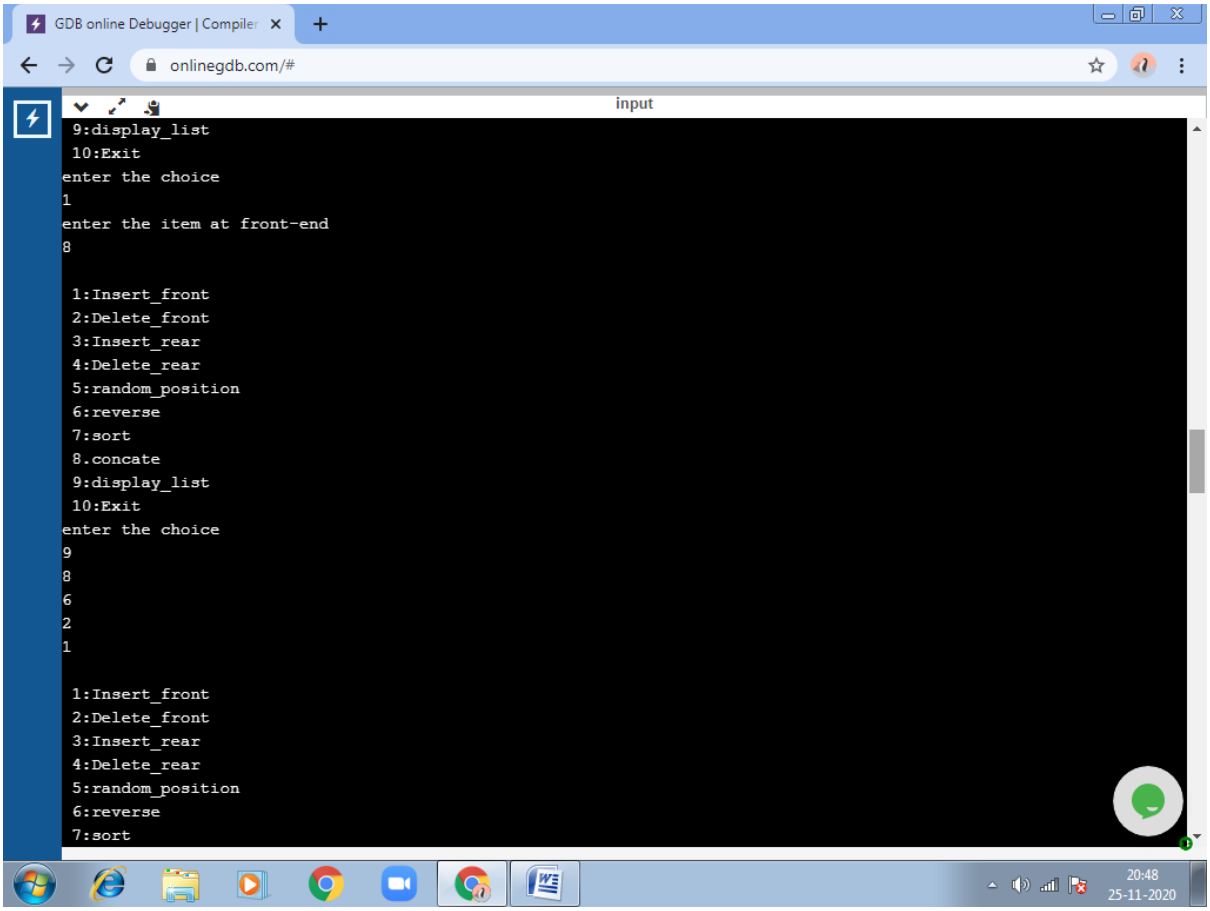

GDB online Debugger | Compiler
onlinegdb.com/#
input
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
2
20:45
25-11-2020

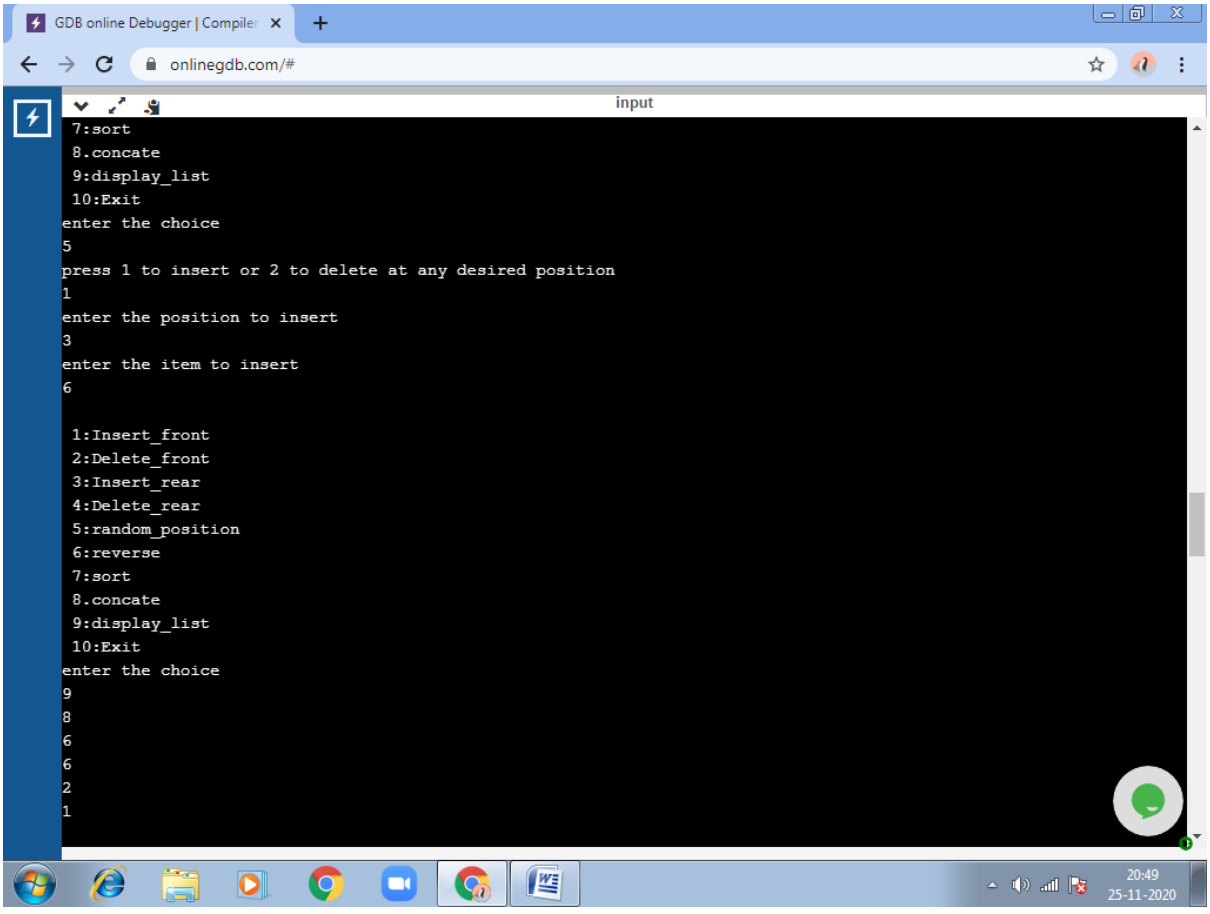

GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the item at front-end
2
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
3
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
3
enter the item at rear-end
20:46
25-11-2020

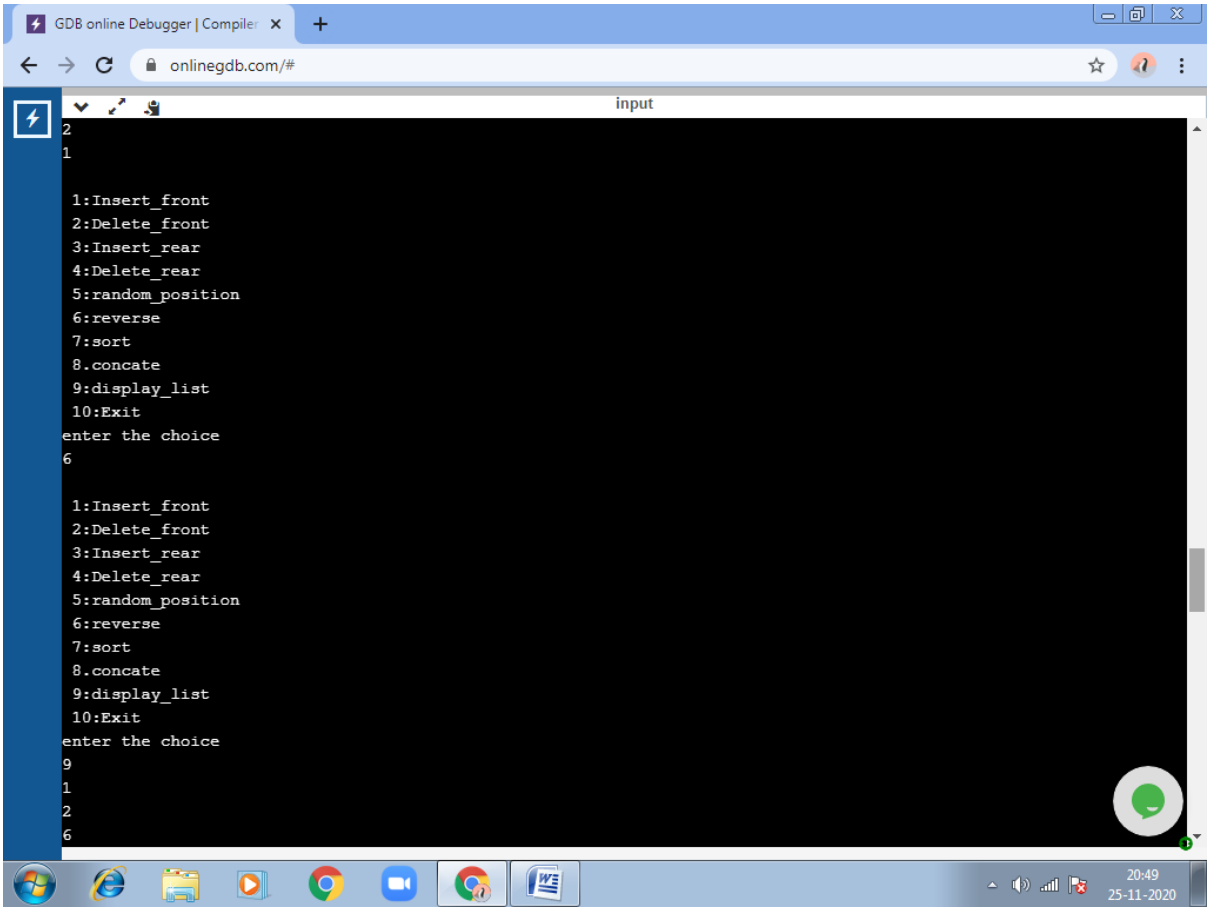

GDB online Debugger | Compiler
onlinegdb.com/#
input
3
enter the item at rear-end
4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
3
2
1
4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
20:46
25-11-2020

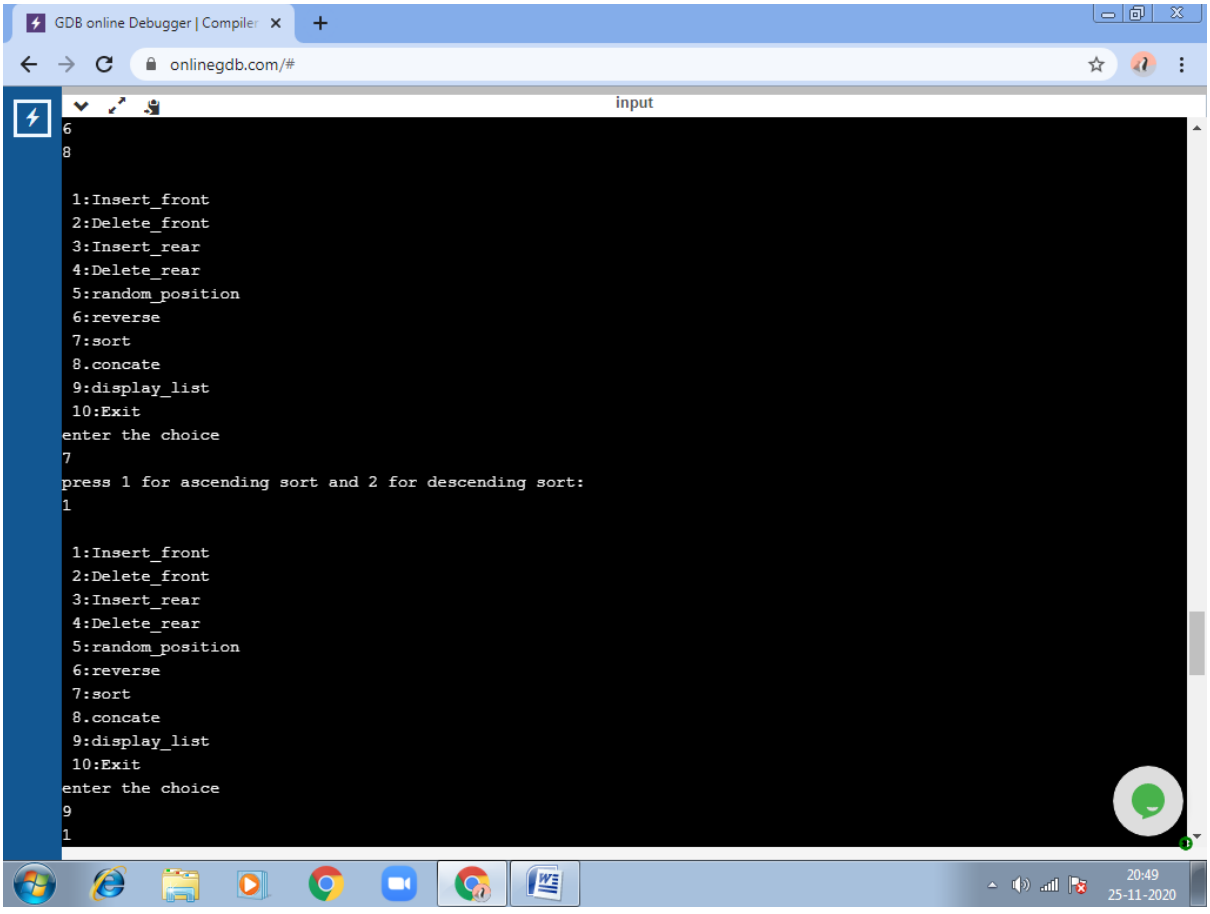

GDB online Debugger | Compiler
onlinegdb.com/#
input
10:Exit
enter the choice
2
item deleted at front-end is=3
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
4
iten deleted at rear-end is 4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
2
20:47
25-11-2020

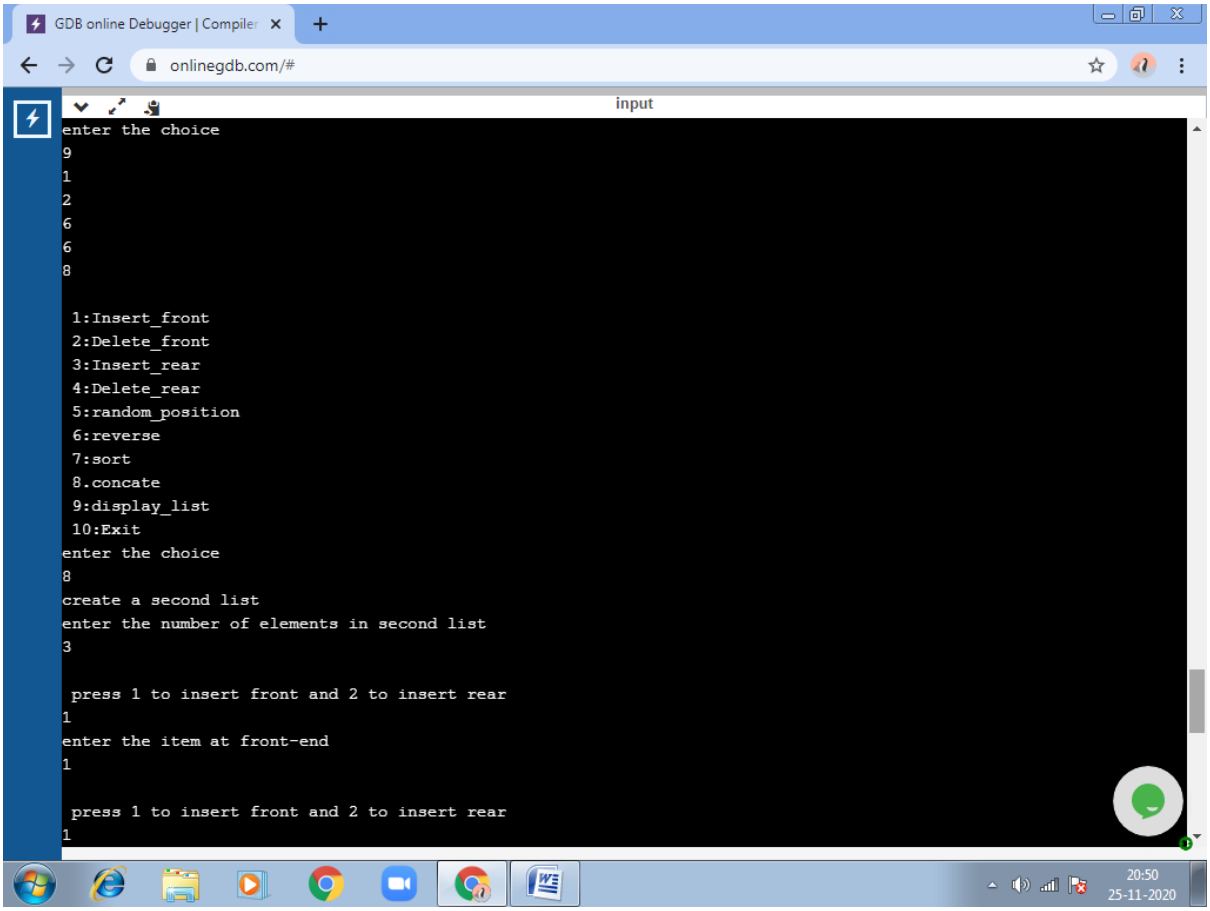

GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the choice
9
2
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
20:47
25-11-2020


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.concate
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

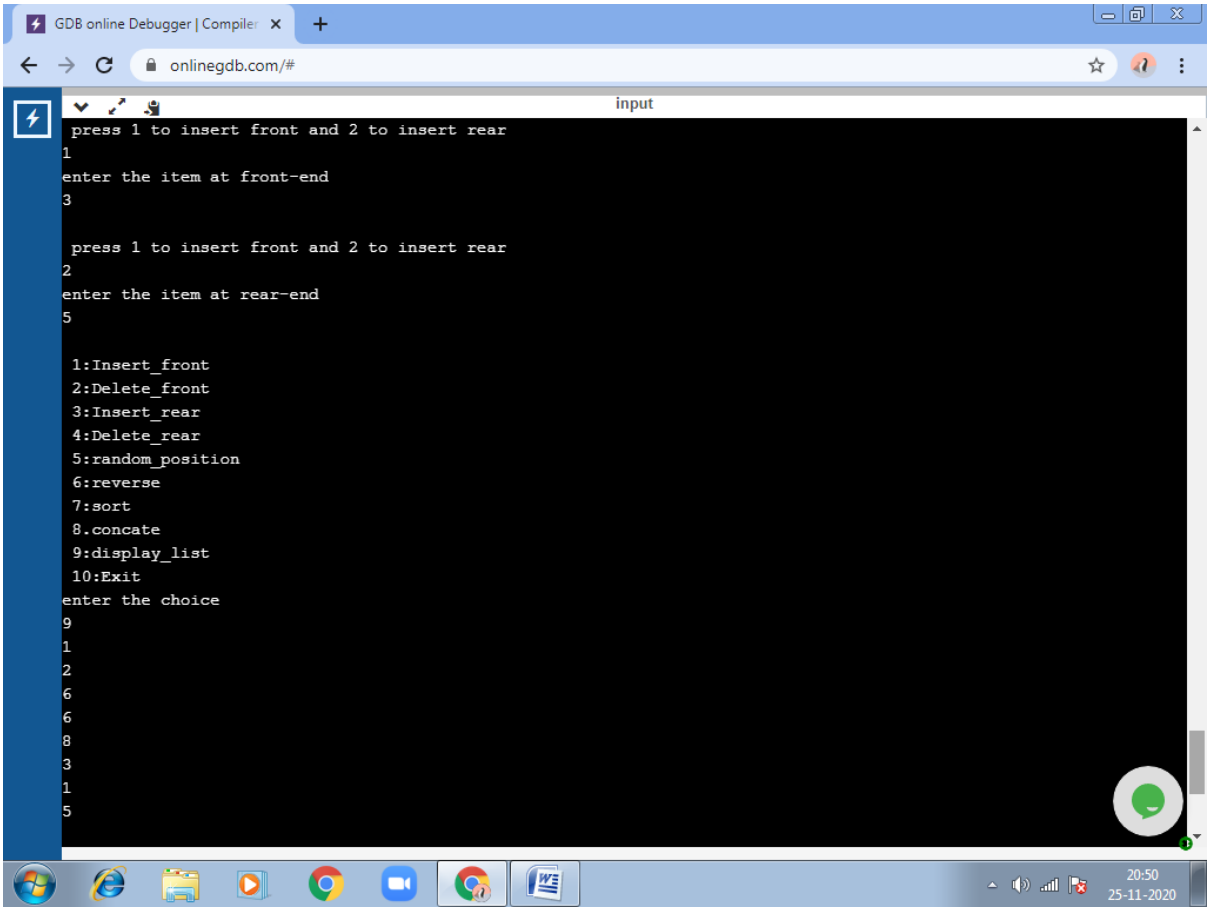

GDB online Debugger | Compiler
onlinegdb.com/#
input
7:sort
8.concate
9:display_list
10:Exit
enter the choice
5
press 1 to insert or 2 to delete at any desired position
1
enter the position to insert
3
enter the item to insert
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
8
6
6
2
1
20:49
25-11-2020

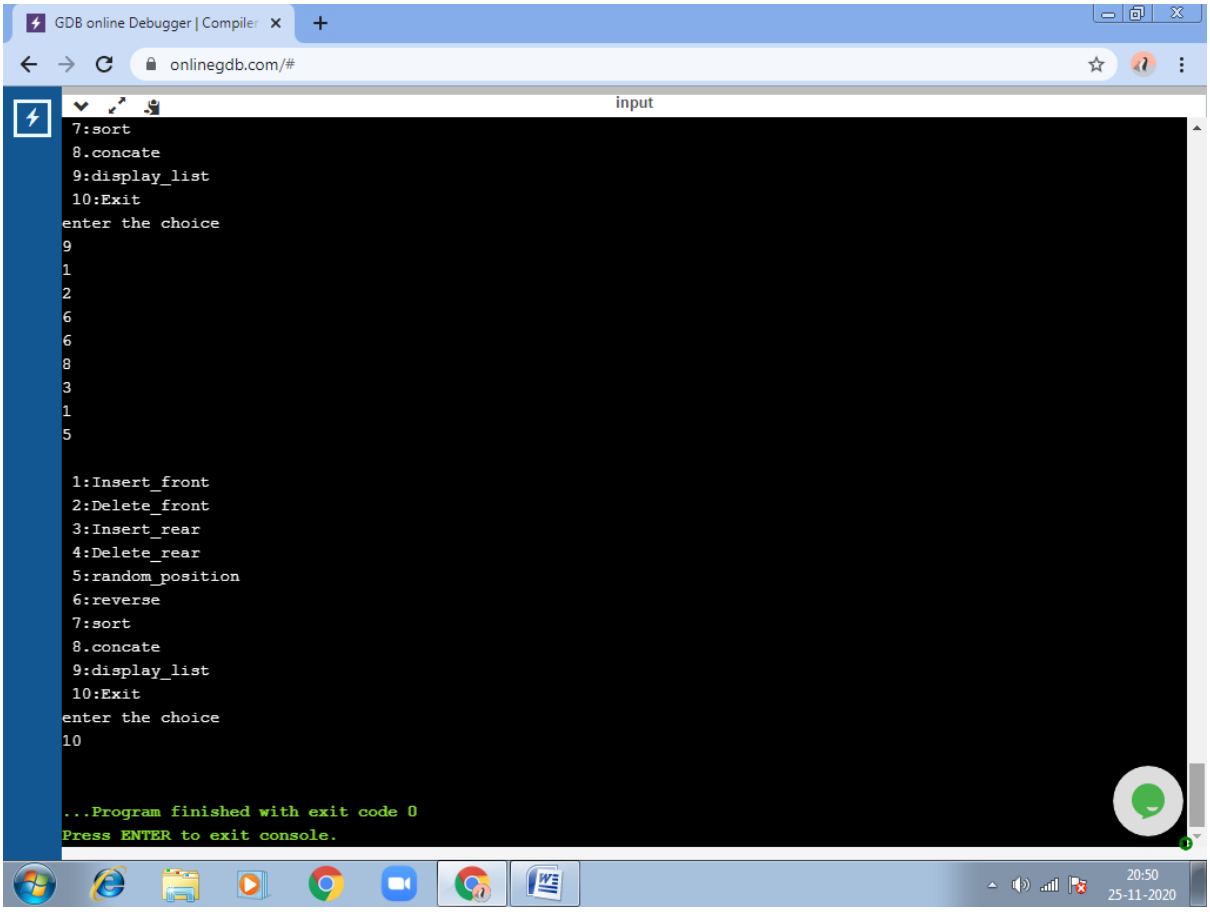

GDB online Debugger | Compiler
onlinegdb.com/#
input
2
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
20:49
25-11-2020


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.concate
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.concate
9:display_list
10:Exit
enter the choice
9
1
20:49
25-11-2020

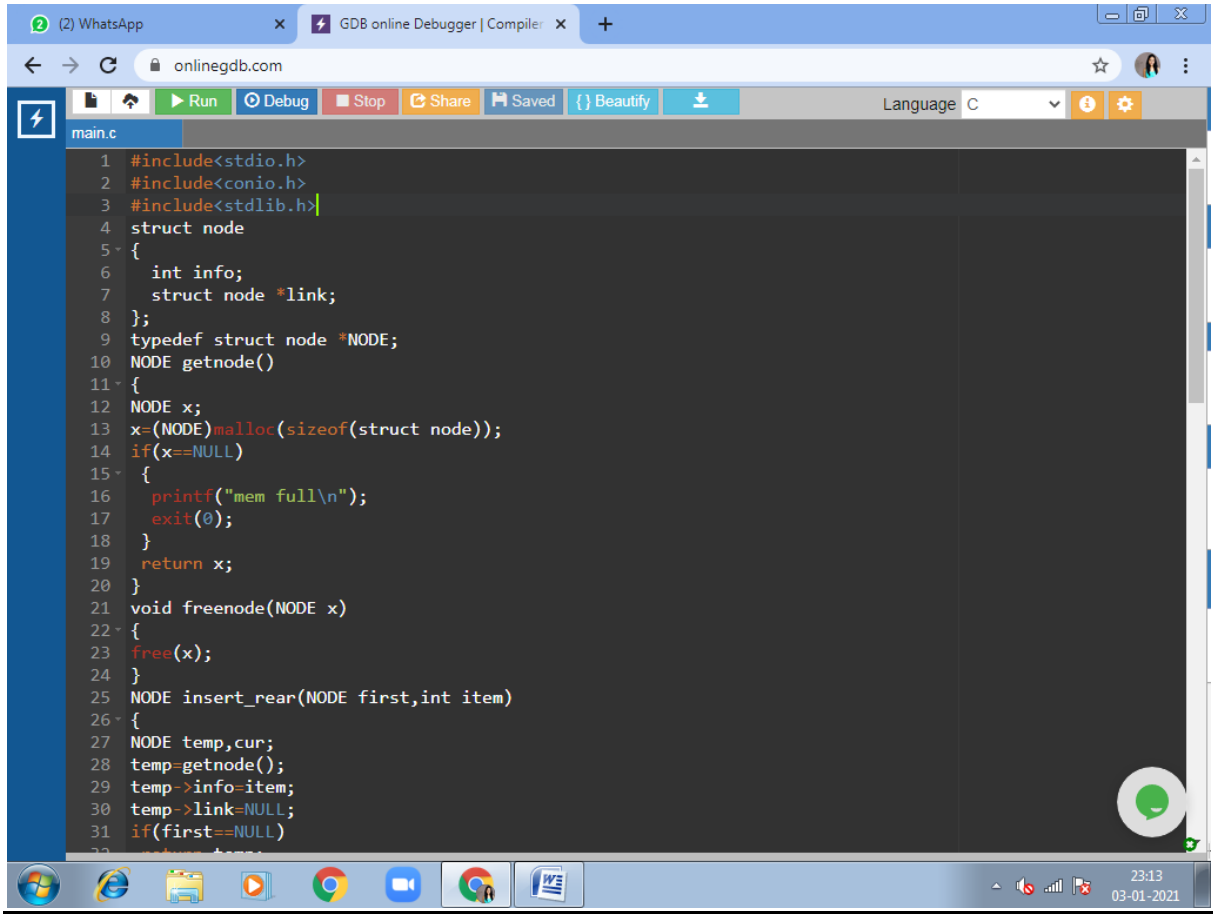

GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the choice
9
1
2
6
6
8
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
3
press 1 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

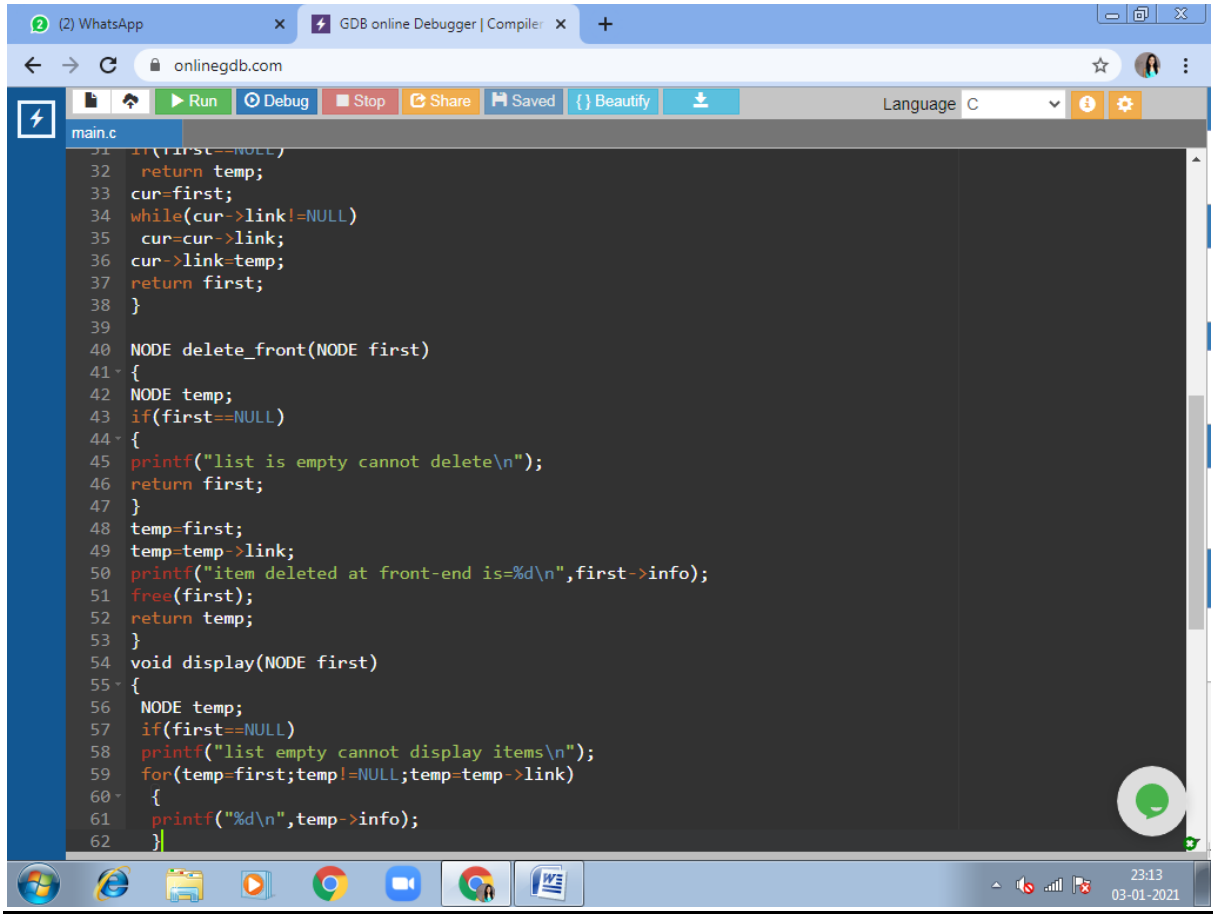
\


GDB online Debugger | Compiler
onlinegdb.com/#
input
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.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5
20:50
25-11-2020

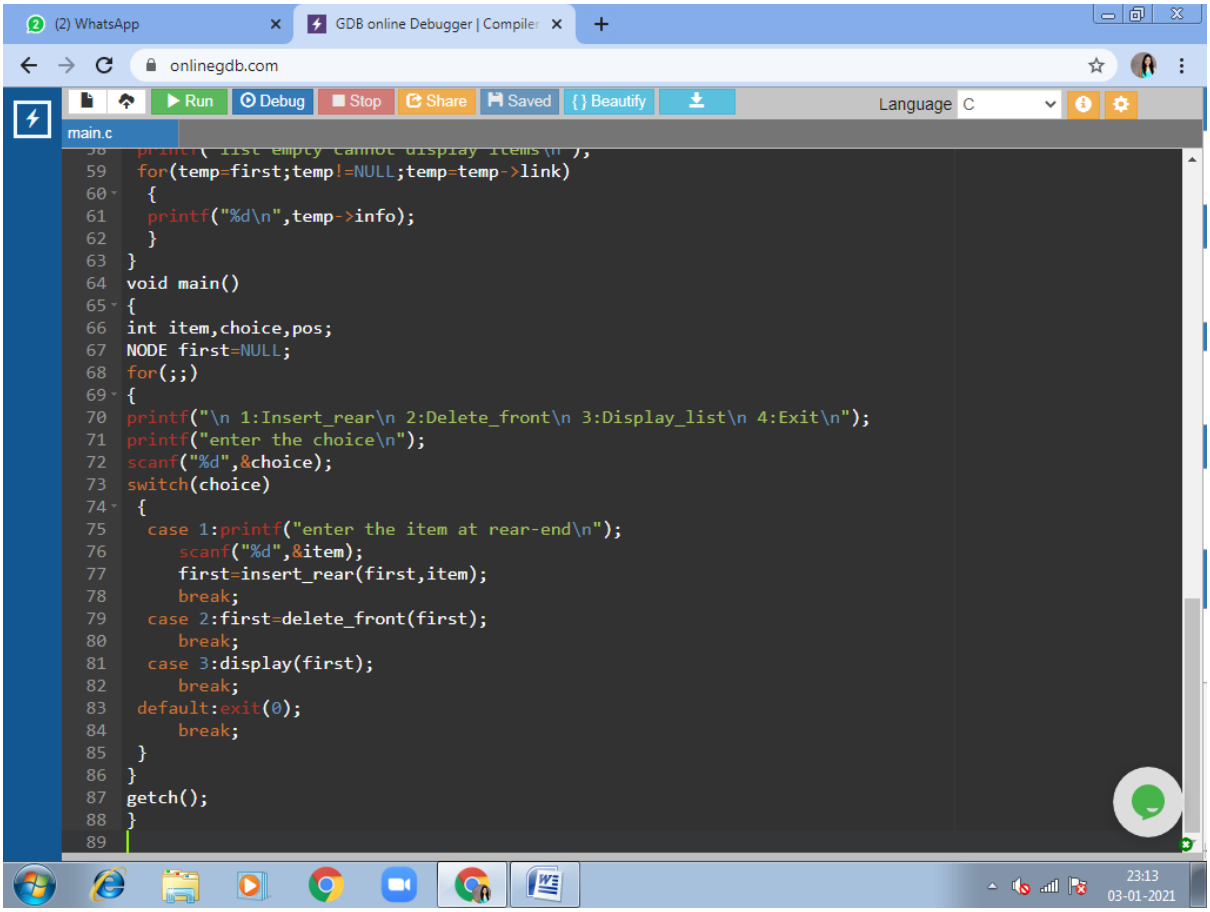

GDB online Debugger | Compiler
onlinegdb.com/#
input
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
10
...Program finished with exit code 0
Press ENTER to exit console.
20:50
25-11-2020

## Q1)wap to implement queues using linked lists:

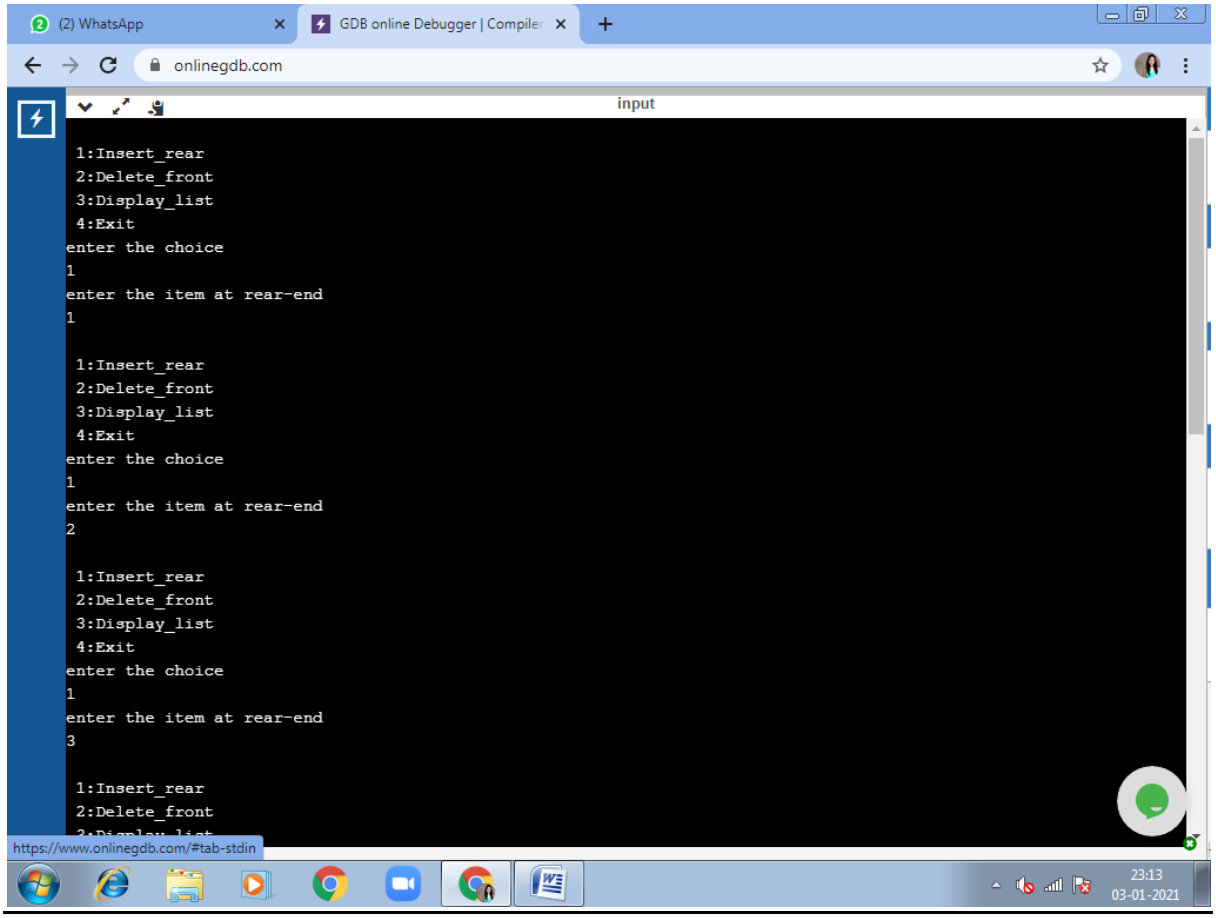


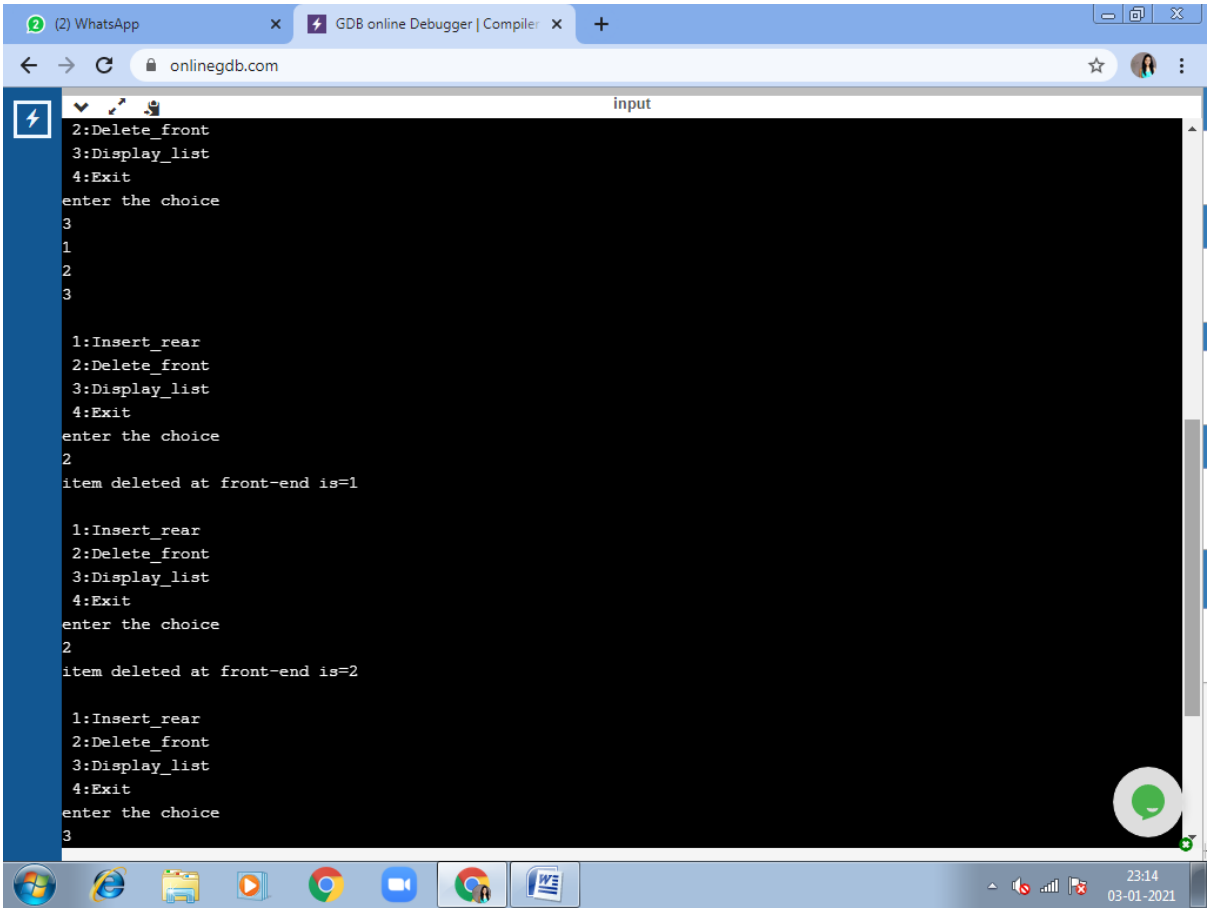


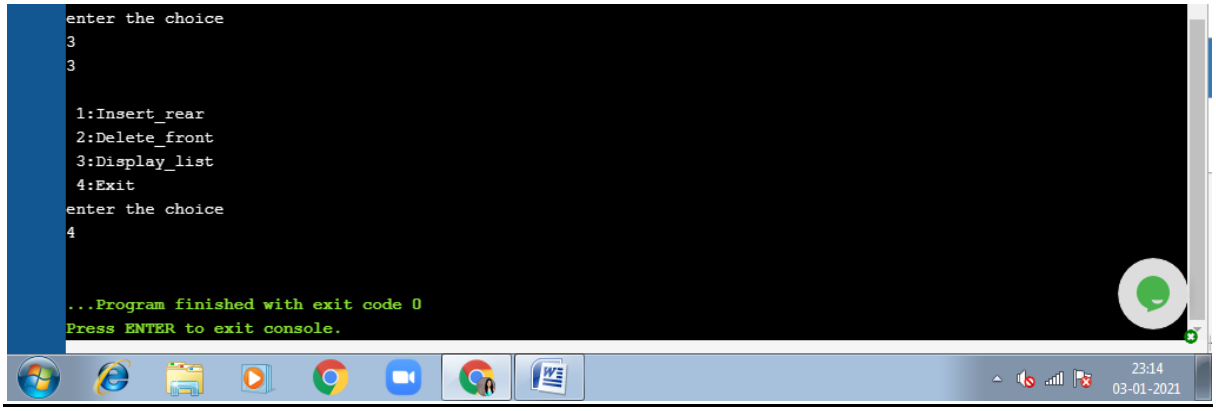
```
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}

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



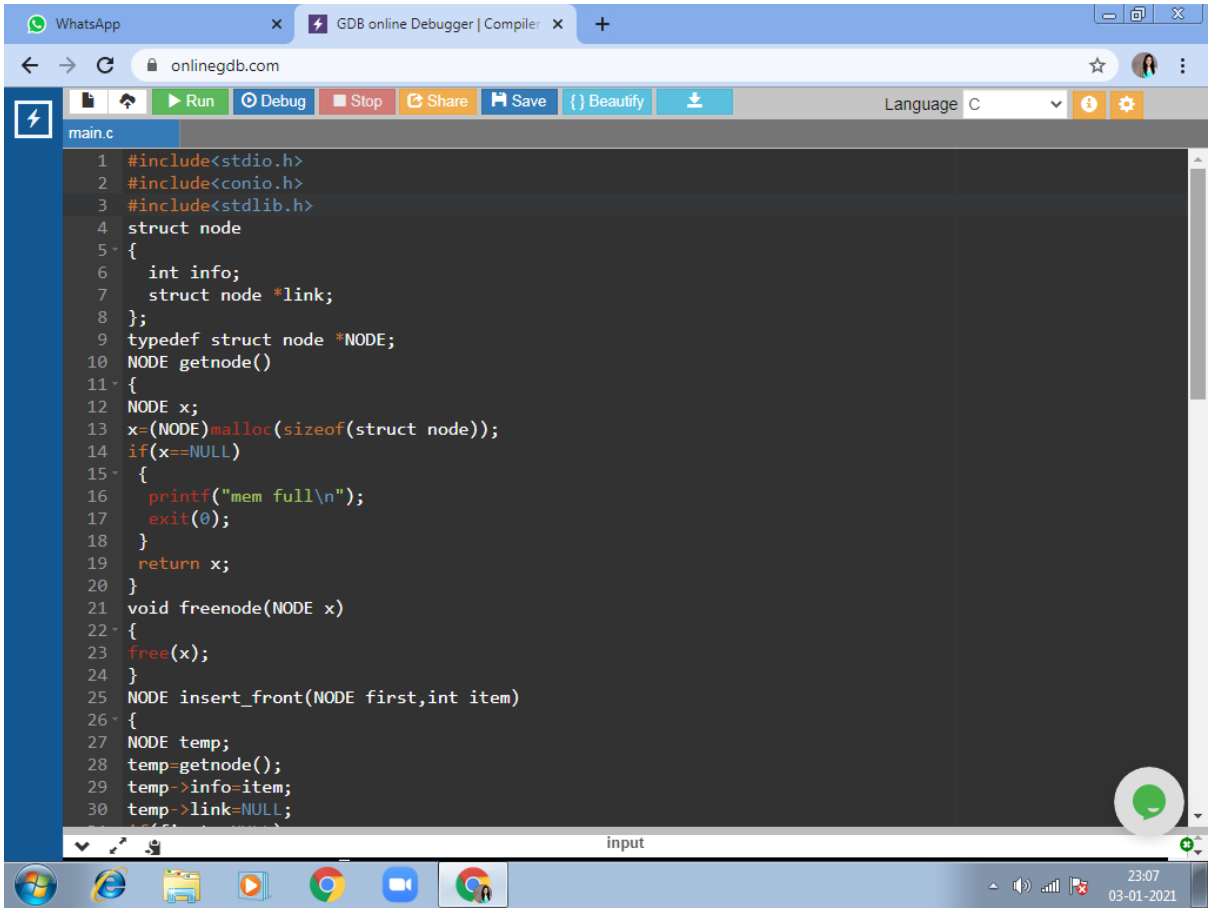
```
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item at rear-end\n");
     scanf("%d",&item);
     first=insert_rear(first,item);
     break;
  case 2:first=delete_front(first);
     break;
  case 3:display(first);
     break;
 default:exit(0);
     break;
 }
}
getch();
}
```

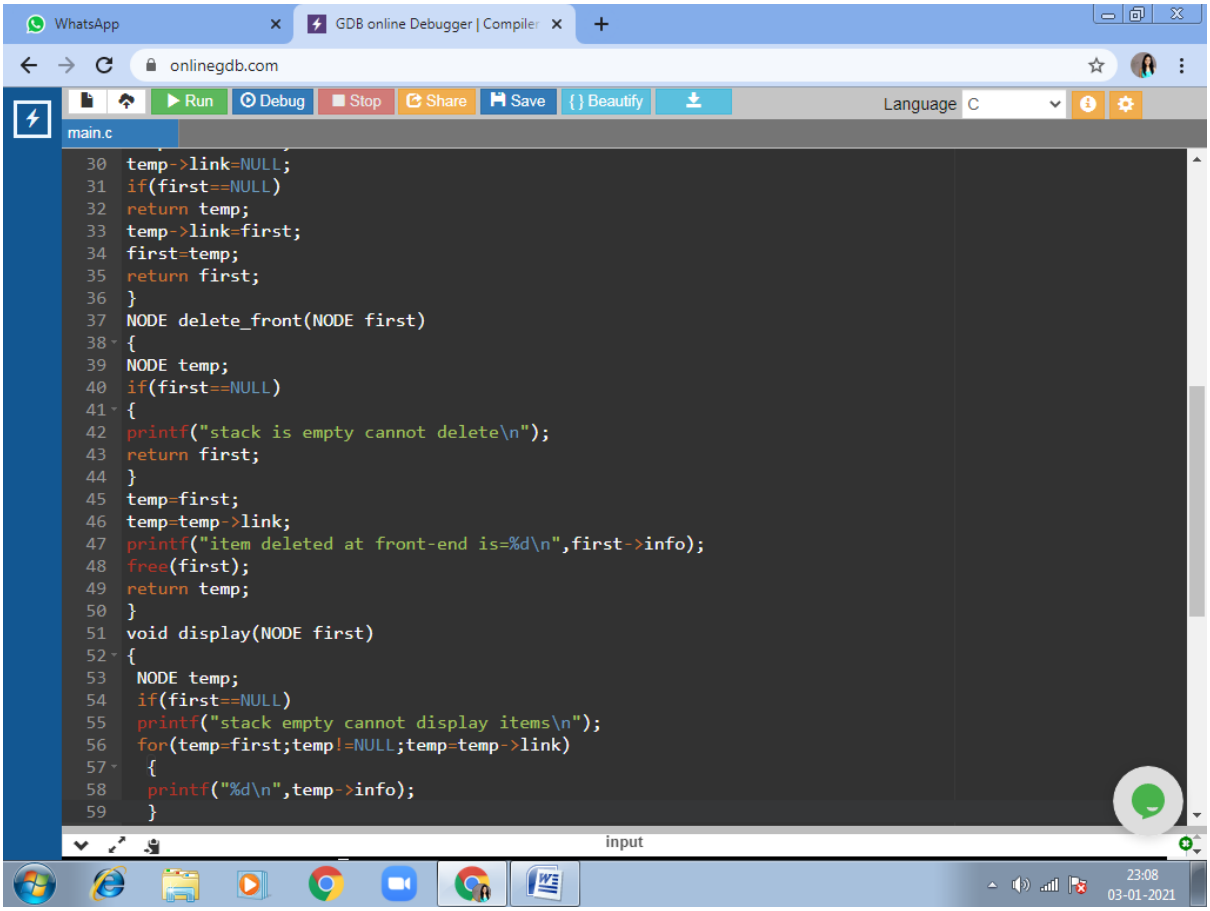

(2) WhatsApp
GDB online Debugger | Compiler
onlinegdb.com
input
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
1
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
2
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
3
1:Insert_rear
2:Delete_front
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

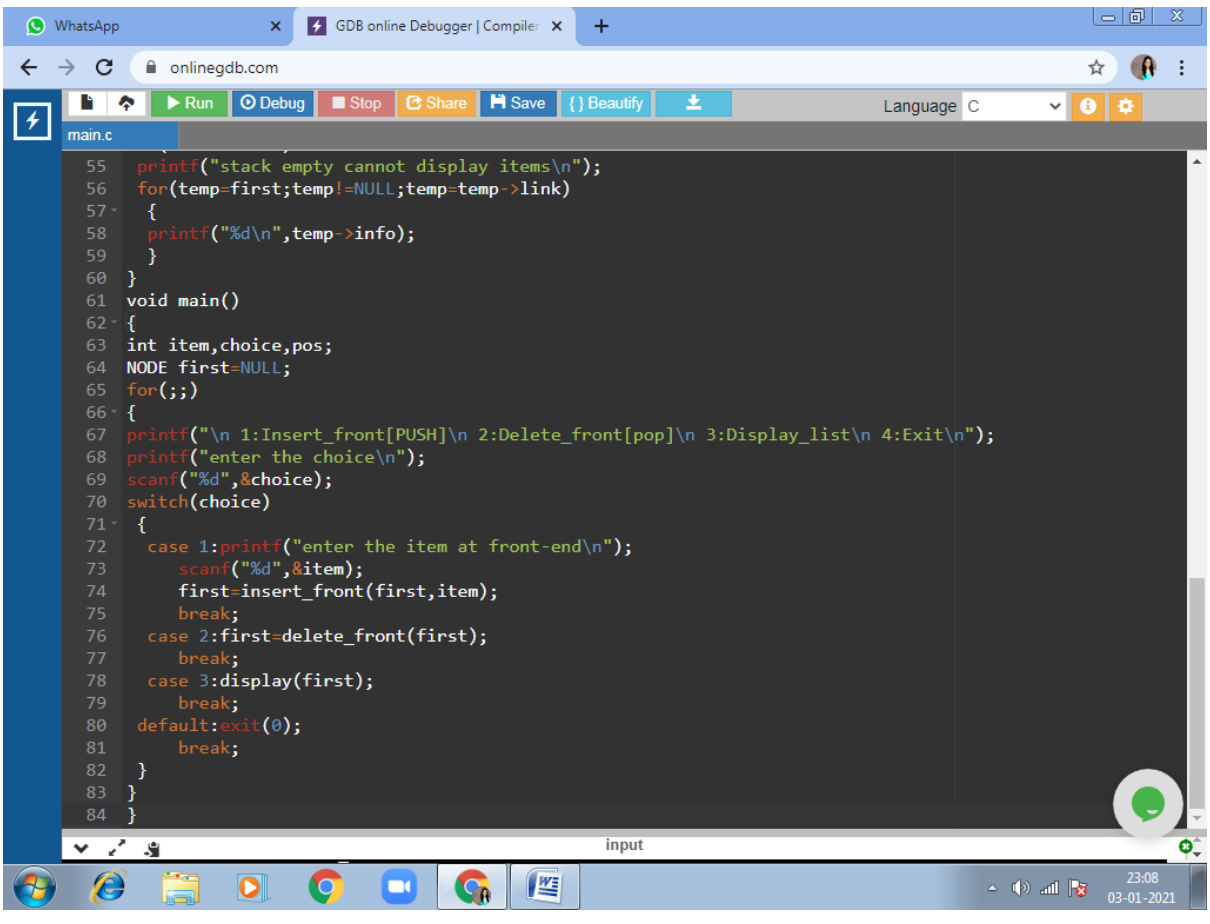



## Q2)wap to implement stacks using linked lists:

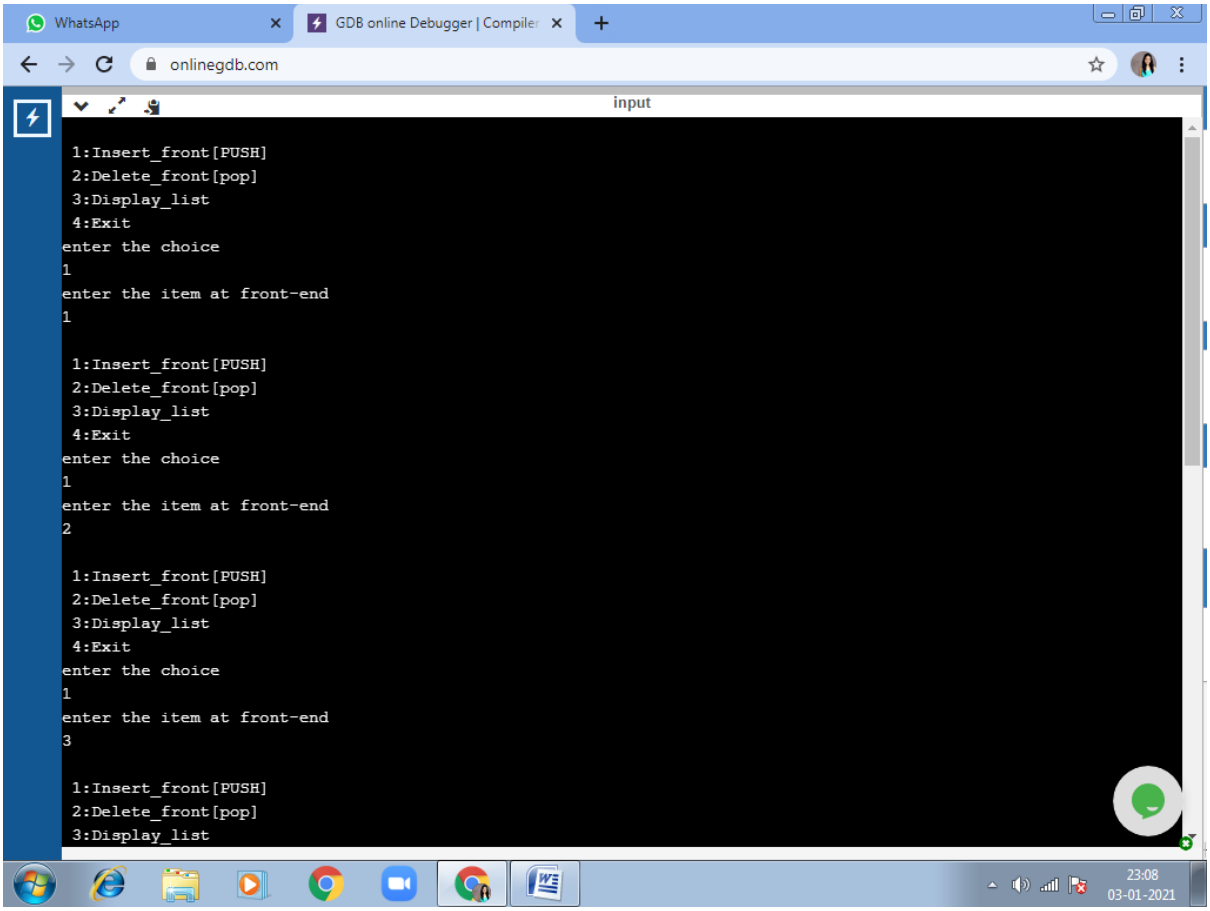


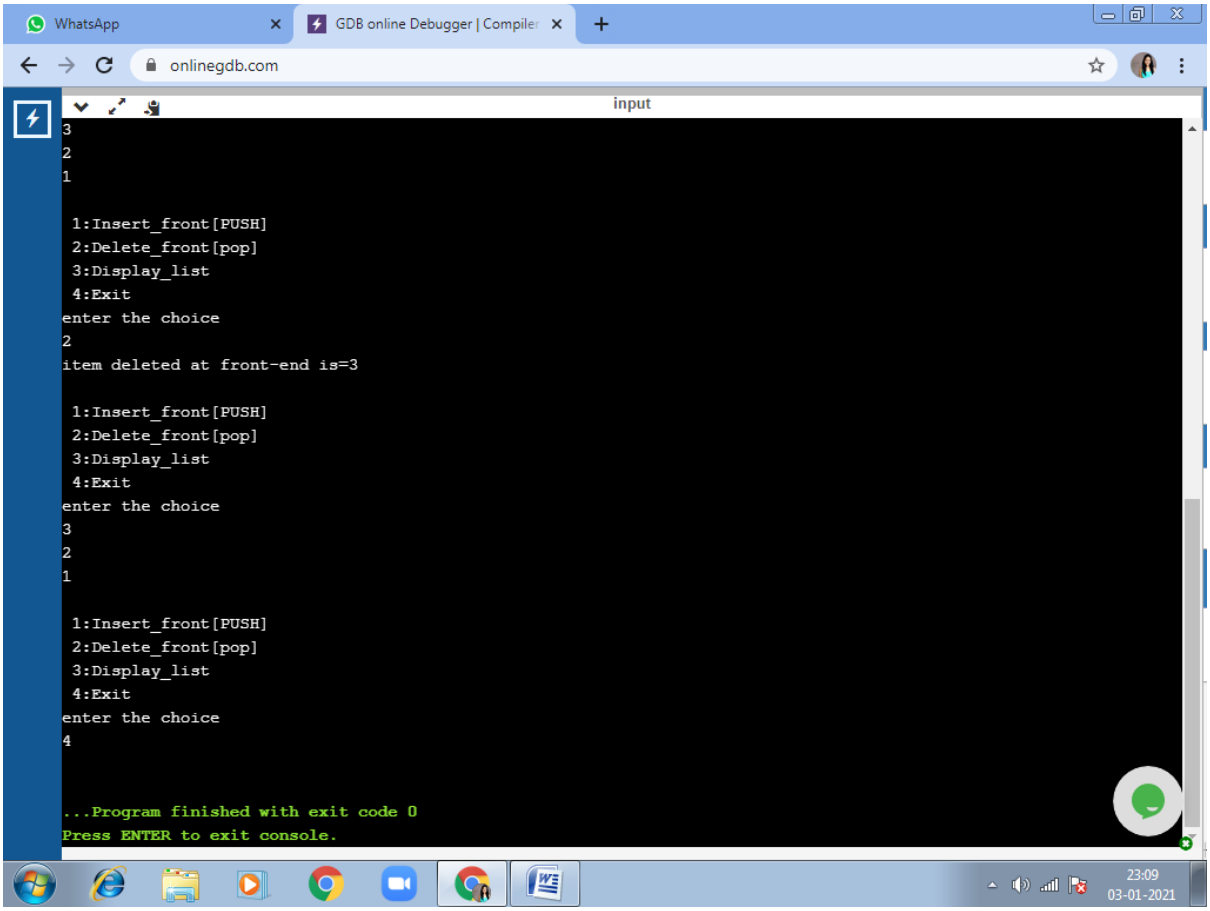


```
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("stack is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);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



```
 printf("stack empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front[PUSH]\n 2:Delete_front[pop]\n 3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item at front-end\n");
     scanf("%d",&item);
     first=insert_front(first,item);
     break;
  case 2:first=delete_front(first);
     break;
  case 3:display(first);
     break;
 default:exit(0);
     break;
 }
}
}
```


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:

```c
#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=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("iten deleted at rear-end is %d",cur->info);
free(cur);
```

```
printf( item deleted at rear-end is %d ",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE asc(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
            int temp;

if(first== NULL) {
       return 0;
       }
else {
      while(prev!= NULL) {

       cur = prev->link;

       while(cur!= NULL) {

        if(prev->info > cur->info) {
       temp = prev->info;
       prev->info = cur->info;
       cur->info = temp;
        }
        cur = cur->link;
       }
       prev= prev->link;
                }
           }
```

```
            return first;
        }
NODE des(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
            int temp;

if(first==NULL) {
    return 0;
     }
    else {
      while(prev!= NULL) {

       cur = prev->link;

        while(cur!= NULL) {

        if(prev->info < cur->info) {
       temp = prev->info;
       prev->info = cur->info;
       cur->info = temp;
           }
           cur = cur->link;
           }
           prev= prev->link;
            }
            }
            return first;
        }
int count(NODE first)
```

```
        }
int count(NODE first)
{
    NODE cur;
    cur=first;
    int c;
    if(first==NULL)
    {
        c=0;

    return c;
    }
    if(first->link==NULL)
    {
        c=1;
    return c;
    }
    while(cur!=NULL)
    {

        c++;
        cur=cur->link;

    }
    return c;
}

void search(NODE first,int ele)
{
    NODE cur;
    cur=first;
```

```
    cur=first;
    int flag=0;
    if(first==NULL)
    {
        printf("list empty can't search \n");
        return ;
    }
    while(cur!=NULL)
    {
        if(ele==cur->info)
        {
            flag=1;
            break;
        }
        cur=cur->link;
    }
    if(flag==1)
    printf("SEARCH SUCCESSFULL \n");
    else
    printf("SEARCH UNSUCCESSFULL \n");


}
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);
  }
}
void main()
{
int item,choice,choice2,j,key;
NODE first=NULL;


for(;;)
{
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"
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:
  printf("enter item to be inserted at front end \n");
  scanf("%d",&item);
  first=insert_front(first,item);
  break;
  case 2:
  delete_rear(first);
  break;
  case 3:
  j=count(first);
  printf("no of items in list: %d ",j);
  break;
  case 4:
  printf("press 1 for ascending order and 2 for descending order \n");
```

```
  case 4:
  printf("press 1 for ascending order and 2 for descending order \n");
  scanf("%d",&choice2);
  if(choice2==1)
  first=asc(first);
  if(choice2==2)
  first=des(first);
  break;
  case 5:
  printf("enter element to be serached for \n");
  scanf("%d",&key);
  search(first,key);
  break;
  case 6:
  display(first);
  break;
  default:exit(0);
     break;
 }
}
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 serached 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 serached 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

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
     int info;
     struct node *llink;
     struct node *rlink;
     };
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("mem full\n");
        exit(0);
        }
    return x;
    }
void freenode(NODE x)
{
    free(x);
}
NODE dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}
NODE ddelete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("dq empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
```

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

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

NODE insert_rightpos(int item,NODE head)
{
NODE temp,cur,next;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
```

```
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 next=cur->rlink;
 printf("enter towards right of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 cur->rlink=temp;
 temp->llink=cur;
 next->llink=temp;
 temp->rlink=next;
 return head;
}
NODE search(NODE head,int item)
{
 NODE temp,cur;
 int flag=0;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
```

```
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
    flag=1;
    break;
}
cur=cur->rlink;
}
 if(cur==head)
 printf("search unsuccessfull\n");
 if(flag==1)
 printf("search successfull\n");
}
NODE delete_all_key(int item,NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head)
    {
        printf("list empty\n");
        return head;
    }
    count=0;
    cur=head->rlink;
    while(cur!=head)
    {
        if(item!=cur->info)
        cur=cur->rlink;
        else
```

```
        else
        {
            count++;
            prev=cur->llink;
            next=cur->rlink;
            prev->rlink=next;
            next->llink=prev;
            freenode(cur);
            cur=next;
        }
    }
    if(count==0)
    printf("not found\n");
    else{
    printf("found at %d positions and are deleted",count);
      return head;
    }
}

void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of dq\n");
temp=head->rlink;
while(temp!=head)
{
```

```
while(temp!=head)
{
printf("%d",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
//clrscr();
for(;;)
{
    printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of k
    printf("enter the choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("enter the item at front end\n");
            scanf("%d",&item);
            last=dinsert_front(item,head);
            break;
        case 2: printf("enter the item at rear end\n");
            scanf("%d",&item);
            last=dinsert_rear(item,head);
            break;
        case 3:last=ddelete_front(head);
```

```
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     }
268 getch();
269 }
```

```
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 occurances
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 occurances
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 occurances
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 occurances
```

```
8:delete repeating occurances
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 occurances
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 occurances
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 occurances
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 occurances
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 occurances
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 occurances
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
```

```
7:search
8:delete repeating occurances
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 occurances
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

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

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

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

```
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

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

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

```
void inorder(NODE root)
{
if(root!=NULL)
 {

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

```
  case 5:inorder(root);
         break;
  case 6:printf("enter the item\n");
         scanf("%d",&item);
         root=delete(root,item);
         break;
  default:exit(0);
          break;
       }
     }
 }
```

```
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

```c
#include<stdio.h>
#include<conio.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;
```

\

```c
int i;
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;
```

```
prev->rlink=temp;
}
return(root);
}
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)
{
display(root->rlink,i+1);
for (j=1;j<=i;j++)
printf("   ");
printf("%d\n",root->info);
display(root->llink,i+1);
}
}

void main()
{
NODE root=NULL;
int choice,i,item;
//clrscr();
for(;;)
{
printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.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:exit(0);
}
}
}
```

```
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.
```