Report on

# "Data Structures"

*Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering in the course of* ***Data Structures (19CS3PCDST)***

Submitted by

**Divyanshu**
**(1BM19CS052)**

Under the Guidance of
**Dr. Kayarvizhy N.**
Associate Professor
Department of CSE


Department of Computer Science and Engineering
BMS College of Engineering
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2020-2021

# B M S COLLEGE OF ENGINEERING

**P.O. Box No: 1908 Bull Temple Road Bangalore-560019**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ASSESSMENT**

Report on **Data Structures (19CS3PCDST),** "Advanced Algorithm assignment" has been successfully completed by **Divyanshu** at B.M.S College of Engineering in partial fulfillment of the requirements for the 3rd Semester, degree in Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during academic year 2020-2021.

**Dr. Kayarvizhy N.**
Associate Professor
Department of Computer science

**Final Marks Awarded**

| Obtained | Total |
|---|---|
| | |

# CONTENT

**LAB PROGRAM 1**

**Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow**

PROGRAM

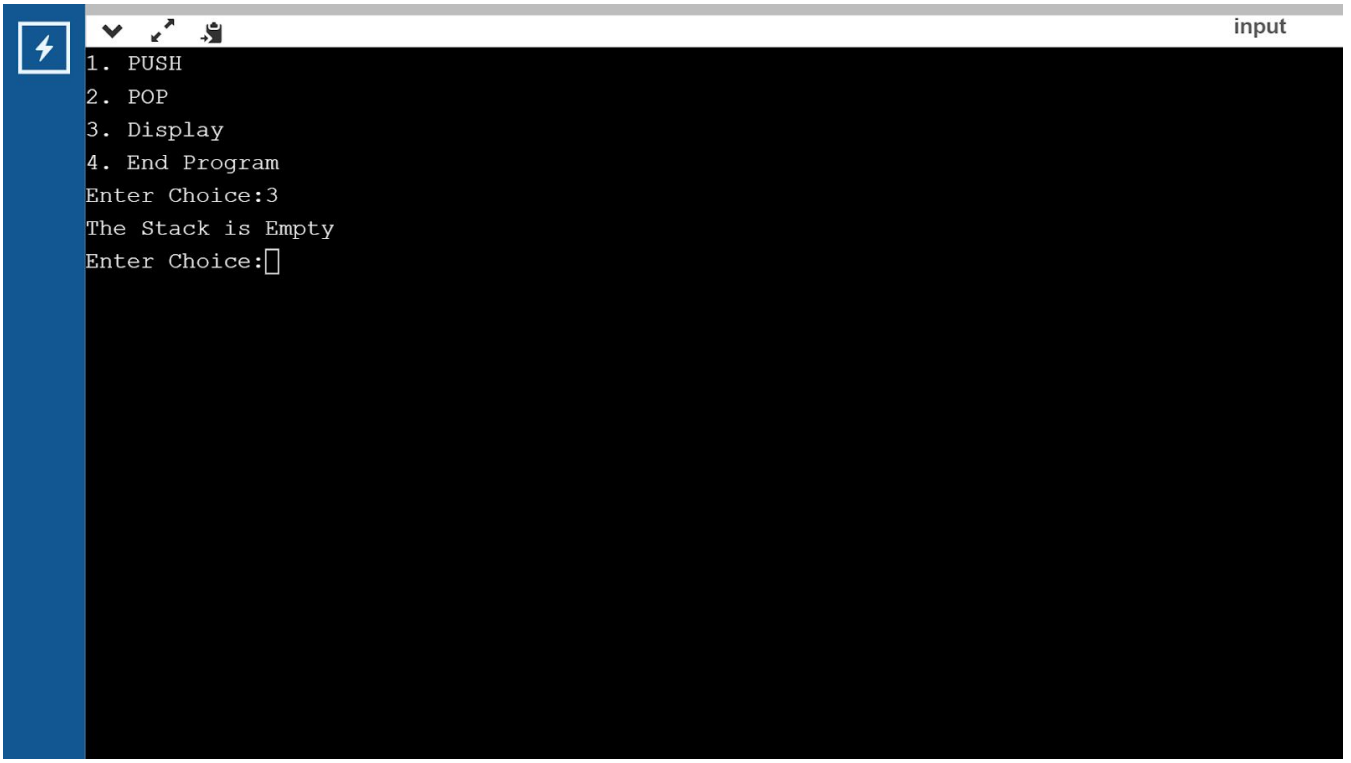
```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 5
int a[MAX],top=-1;
void push();
void pop();
void display();
int main()
{
   int ch;
   printf("1. PUSH\n");
   printf("2. POP\n");
   printf("3. Display\n");
   printf("4. End Program");
   while(1)
   {
      printf("\nEnter Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
```

```
            case 1:
            {
               push();
               break;
            }
            case 2:
            {
               pop();
               break;
            }
            case 3:
            {
               display();
               break;
            }
            case 4:
            {
               exit (0);
            }
            default:
            {
               printf("Wrong Choice");
            }
         }
      }
}
void push()
{
   int data;
   if (top==MAX-1)
   {
      printf("\nStack Overflow");
   }
   else
   {
      printf("Enter Element to be Pushed:");
      scanf("%d",&data);
      top++;
      a[top]=data;
   }
```

```
}
void pop()
{
   if(top==-1)
   {
      printf("Stack Underflow");
   }
   else
   {
      printf("Popped Element: %d",a[top]);
      top--;
   }
}
void display()
{
   int i;
   if(top>=0)
   {
      printf("Elements:");
      for(i=top;i>=0;i--)
      printf("\n%d",a[i]);
   }
   else
   {
      printf("The Stack is Empty");
   }
}
```

**OUTPUT**


input
1. PUSH
2. POP
3. Display
4. End Program
Enter Choice:1
Enter Element to be Pushed:5
Enter Choice:1
Enter Element to be Pushed:6
Enter Choice:1
Enter Element to be Pushed:7
Enter Choice:1
Enter Element to be Pushed:8
Enter Choice:1
Enter Element to be Pushed:9
Enter Choice:1
Stack Overflow
Enter Choice:


input
1. PUSH
2. POP
3. Display
4. End Program
Enter Choice:3
The Stack is Empty
Enter Choice:

## LAB PROGRAM 2

**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)**

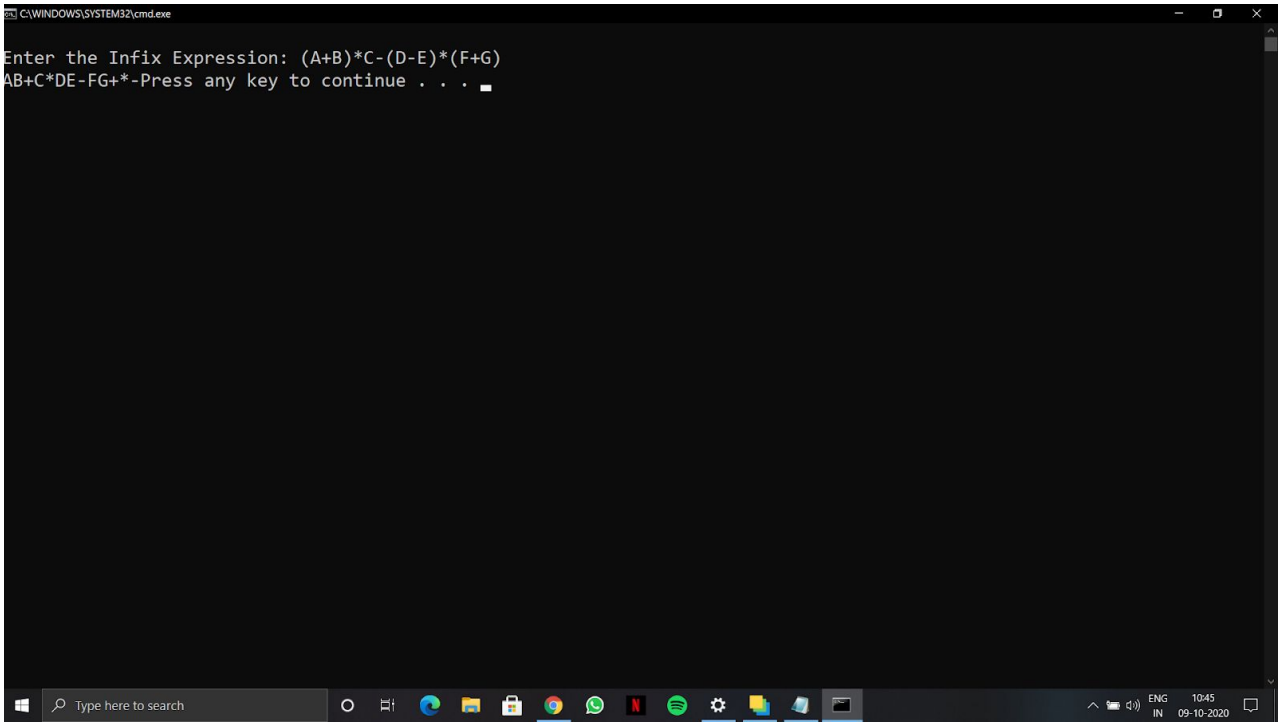
PROGRAM

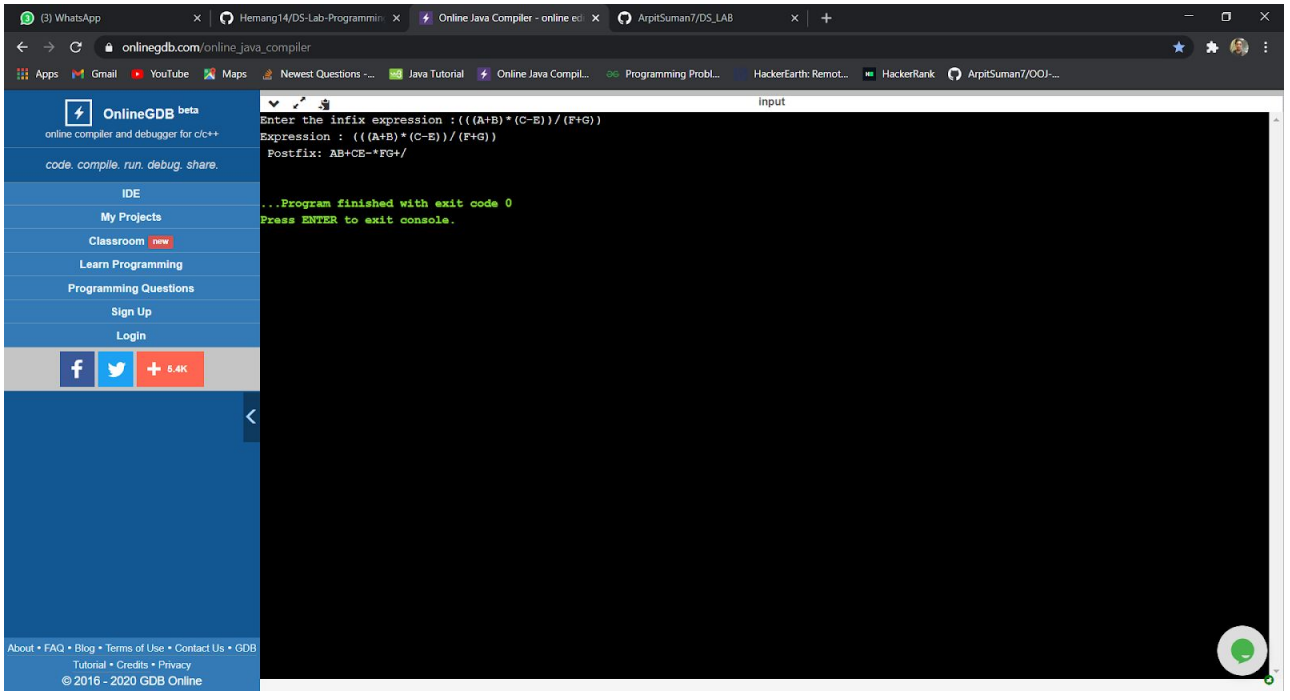
```
#include<stdio.h>
#include<ctype.h>
char st[100];
int top = -1;
void push(char x)
{
   if(top==99)
   {
      printf("\nSTACK OVERFLOW");
   }
   else
   {
   top++;
   st[top] = x;
   }
}
char pop()
{
   if(top == -1)
   {
      printf("\nSTACK UNDERFLOW");
      return -1;
   }
   else
      return st[top--];
}
```

```
int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}
int main()
{
   char exp[100];
   char x;
   printf("\nEnter the Infix Expression: ");
   scanf("%s",exp);
   int i=0;
   while(exp[i] != '\0')
   {
      if(isalnum(exp[i]))
         printf("%c",exp[i]);
      else if(exp[i] == '(')
         push(exp[i]);
      else if(exp[i] == ')')
      {
         while((x = pop()) != '(')
            printf("%c",x);
      }
      else
      {
         while(priority(st[top]) >= priority(exp[i]))
            printf("%c",pop());
         push(exp[i]);
      }
      i++;
   }
   while(top != -1)
   {
      printf("%c",pop());
   }
```

```
    return 0;
}
```

## OUTPUT



**LAB PROGRAM 3**

**WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions**

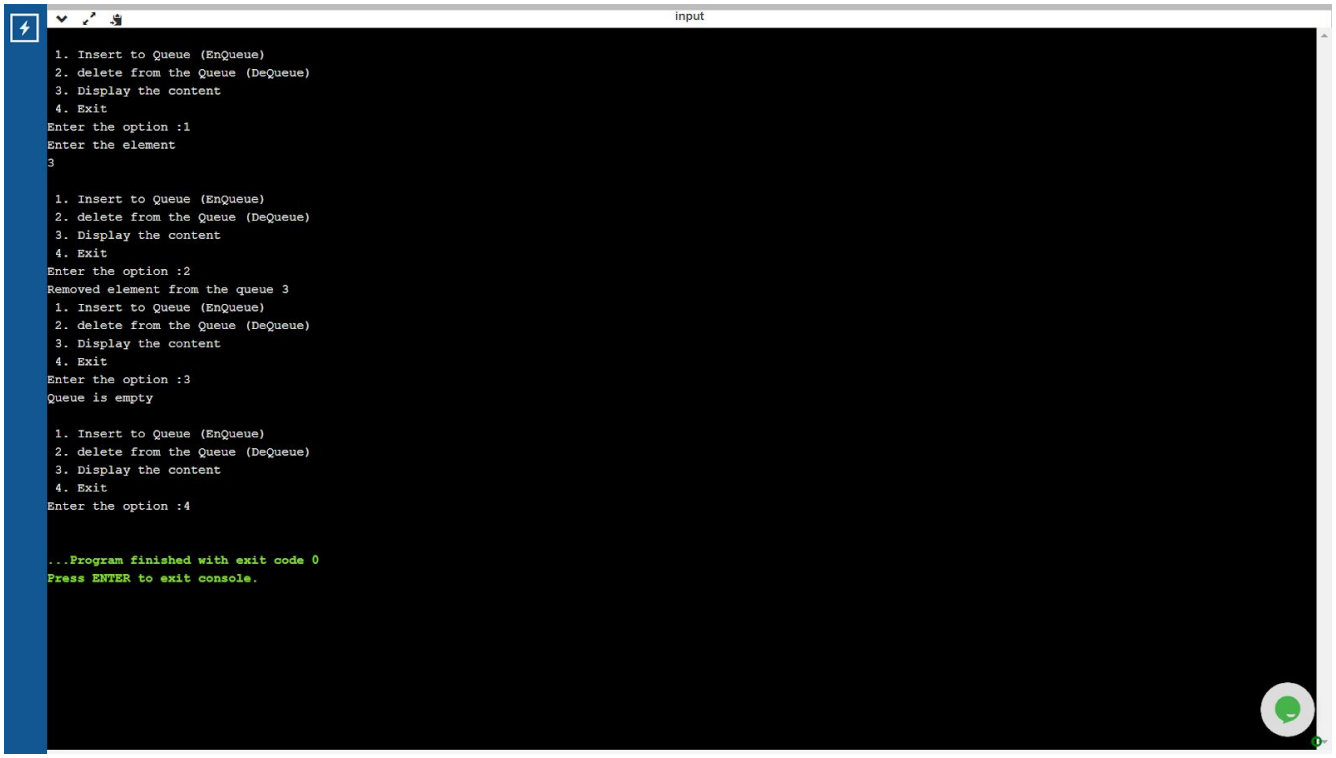
PROGRAM

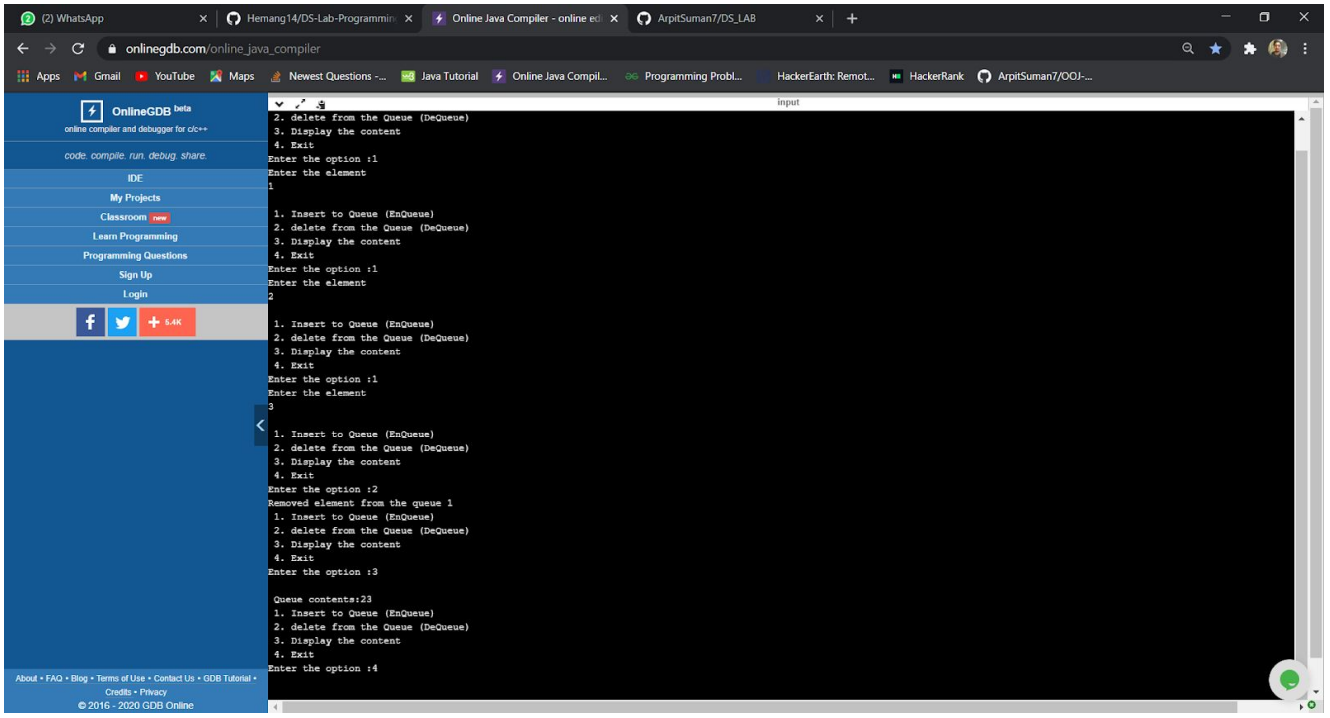
```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5

int front=0;
int rear=-1;

int queue[MAX];

void Enque(int);
int Deque();
void display();
int main(int argc, char **argv)
{
        int option;
   int item;
   do{
      printf("\n 1. Insert to Queue (EnQueue)");
      printf("\n 2. delete from the Queue (DeQueue)");
      printf("\n 3. Display the content ");
      printf("\n 4. Exit\n");
      printf("Enter the option :");
      scanf("%d",&option);
```

```
      switch(option)
      {
         case 1:  printf("Enter the element\n");
                  scanf("%d",&item);
                  Enque(item);
                  break;
         case 2: item=Deque();
                 if(item==-1)
                    printf("Queue is empty\n");
                 else
                 printf("Removed element from the queue %d",item);
                 break;
         case 3: display();
                 break;
         case 4: exit(0);
      }
   } while (option!=4);
           return 0;
}

void Enque(int ele)
{
   if (rear==MAX-1)
      printf("Queue is full\n");
   else
   {
    rear++;
    queue[rear]=ele;

   }
}
int Deque()
```

```
{
   int item;
   if(front == -1)
      return -1;
   else
   {
      item=queue[front];
      front++;
      if(front>rear)
      {
         front=-1;
         rear=-1;
      }
      return item;
   }

}

void display()
{
   int i;
   if(front==-1)
      printf("Queue is empty\n");
   else
   {
      printf("\n Queue contents:");
      for(i=front;i<=rear;i++)
         printf("%d", queue[i]);
   }
}
```

## OUTPUT


input
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
3
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 3
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue is empty
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :4
...Program finished with exit code 0
Press ENTER to exit console.


onlinegdb.com/online_java_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom
Learn Programming
Programming Questions
Sign Up
Login
input
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
1
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
2
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
3
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 1
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue contents:23
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :4
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online

**LAB PROGRAM 4**

**WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions**

PROGRAM

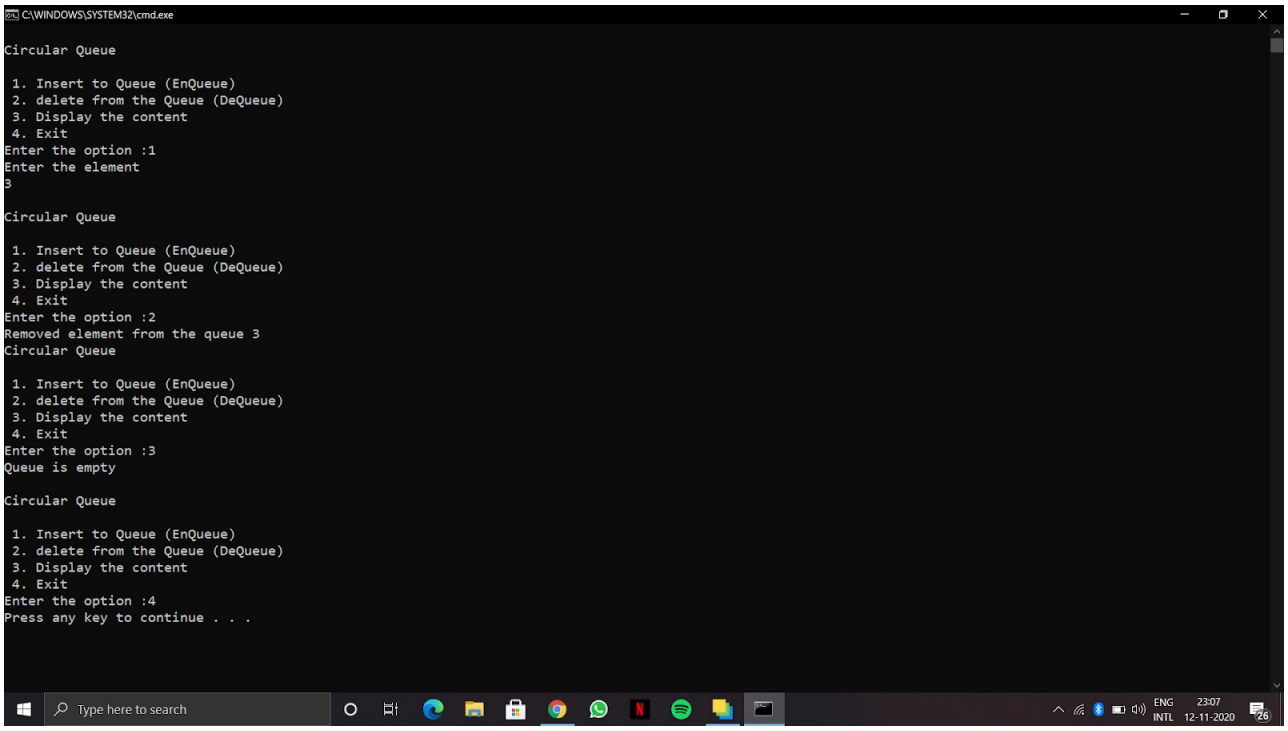
```
#include <stdio.h>
#include <stdlib.h>
#define MAX 3

int front=-1;
int rear=-1;

int queue[MAX];

void Enque(int);
void Deque();
void display();
int main(int argc, char **argv)
{
        int option;
   int item;
   do{
      printf("\nCircular Queue\n");
      printf("\n 1. Insert to Queue (EnQueue)");
      printf("\n 2. delete from the Queue (DeQueue)");
      printf("\n 3. Display the content ");
```

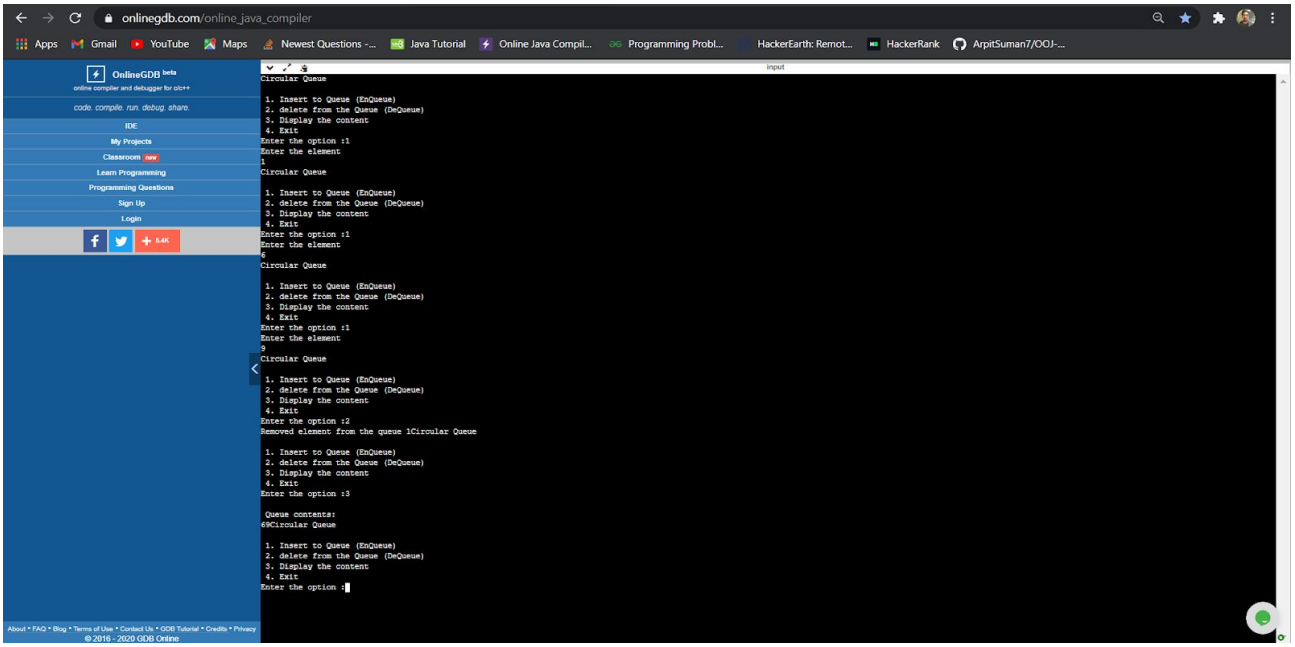
```
        printf("\n 4. Exit\n");
        printf("Enter the option :");
        scanf("%d",&option);
        switch(option)
        {
           case 1:  printf("Enter the element\n");
                    scanf("%d",&item);
                    Enque(item);
                    break;
           case 2: Deque();

                   break;
           case 3: display();
                   break;
           case 4: exit(0);
        }
    } while (option!=4);
           return 0;
}

void Enque(int ele)
{
    if(((front == 0 && rear == MAX - 1))|| (front == rear + 1) )
    {
       printf("Queue is full\n");return;

    }
    else
    {
      rear=(rear+1)%MAX;
```

```
    queue[rear]=ele;
    if(front ==-1)
       front=0;



  }
}
void Deque()
{
  int item;
  if((front == -1)&&(rear == -1))
  {

     printf("Queue is empty");
  }
  else
  {
     item=queue[front];
     printf("Removed element from the queue %d",item);
     if(front==rear)
     {
        front=-1;
        rear=-1;
     }
     else
     {
        front=(front+1)%MAX;
     }

  }
```

```
}

void display()
{
   int i;
   if((front==-1)&& (rear==-1))
   {

      printf("Queue is empty\n");return;

   }
   else
   {
      printf("\n Queue contents:\n");
      i=front;
      do
      {
        printf("%d",queue[i]);
        if(i==rear)
           break;
        i=(i+1)%MAX;
      }while (i!=front);

   }
}
```

## OUTPUT


C:\WINDOWS\SYSTEM32\cmd.exe
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
3
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 3
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue is empty
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :4
Press any key to continue . . .
Type here to search
ENG
INTL
23:07
12-11-2020




## LAB PROGRAM 5

**WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list**

PROGRAM

```
#include<stdio.h>


struct node
{
   int data;
   struct node *next;
};
```

```
struct node *head=NULL;

int length=0;

void insertend(int ele)
{
   struct node *newnode,*temp;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=ele;
   newnode->next=NULL;
   if(head==NULL)
   {
      head=newnode;
      length=1;
   }
   else
   {
      temp=(struct node*)malloc(sizeof(struct node));
      temp=head;
      while(temp->next!=NULL)
      {
         temp=temp->next;
      }
      temp->next=newnode;
      length++;
   }

}

void insertfront(int ele)
```

```
{
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   temp->data=ele;
   temp->next=head;
   head=temp;
   length++;
}

void insertrandom(int ele,int pos)
{
   if(pos==1)
      insertfront(ele);
   else if(pos>=length)
      insertend(ele);
   else
   {
      struct node *inst;
      inst=(struct node*)malloc(sizeof(struct node));
      struct node *temp;
      temp=(struct node*)malloc(sizeof(struct node));
      temp=head;
      for(int i=1;i<pos-1;i++)
      {
            temp=temp->next;
      }
      inst->data=ele;
      inst->next=temp->next;
      temp->next=inst;
      length++;
```

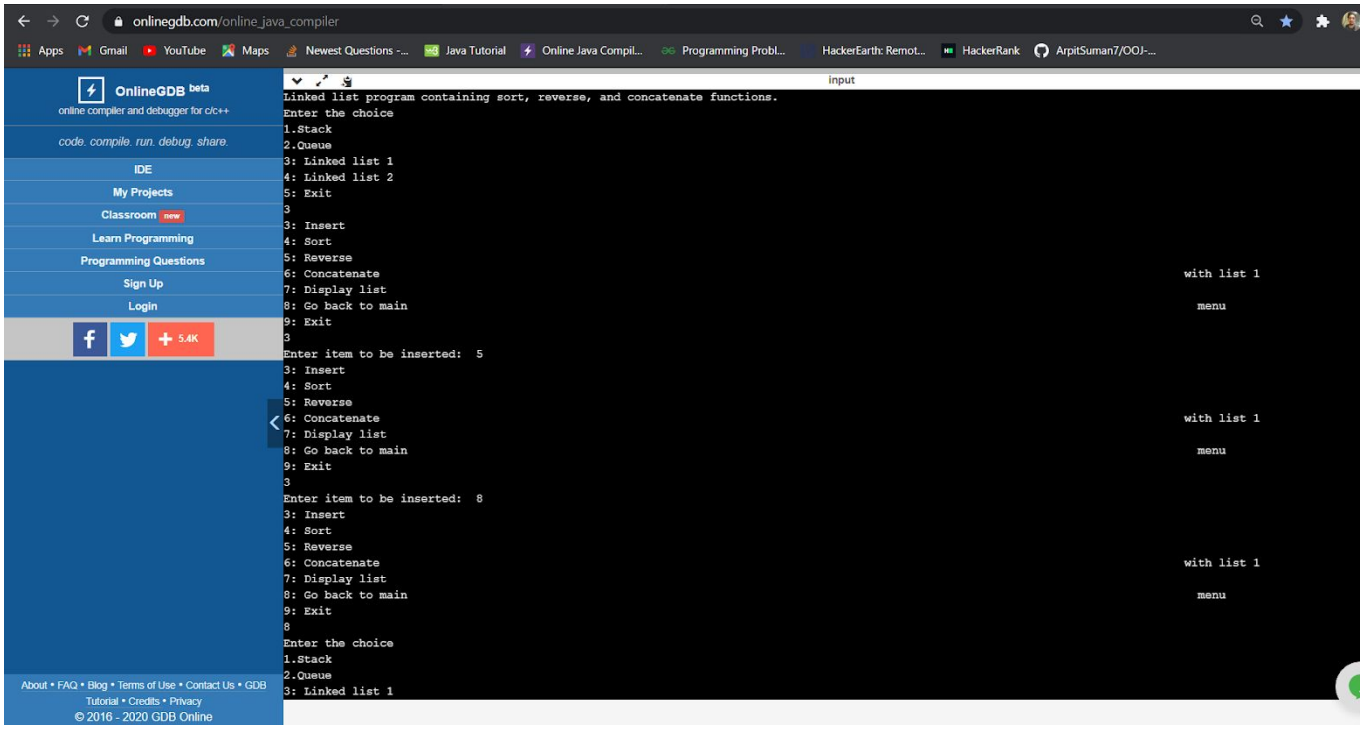
```
    }

}

void deleteele(int ele)
{
    struct node *temp,*del;
    temp=(struct node*)malloc(sizeof(struct node));
    del=(struct node*)malloc(sizeof(struct node));
    del=NULL;
    if(head->data==ele)
    {
       del=head;
       head=head->next;
       del->next=NULL;
    }
    else
    {
       temp=head;
       while(temp->next!=NULL)
       {
         if(temp->next->data==ele)
         {

            del=temp->next;
            temp->next=del->next;
            del->next=NULL;
            length--;
            break;
         }
         else
```

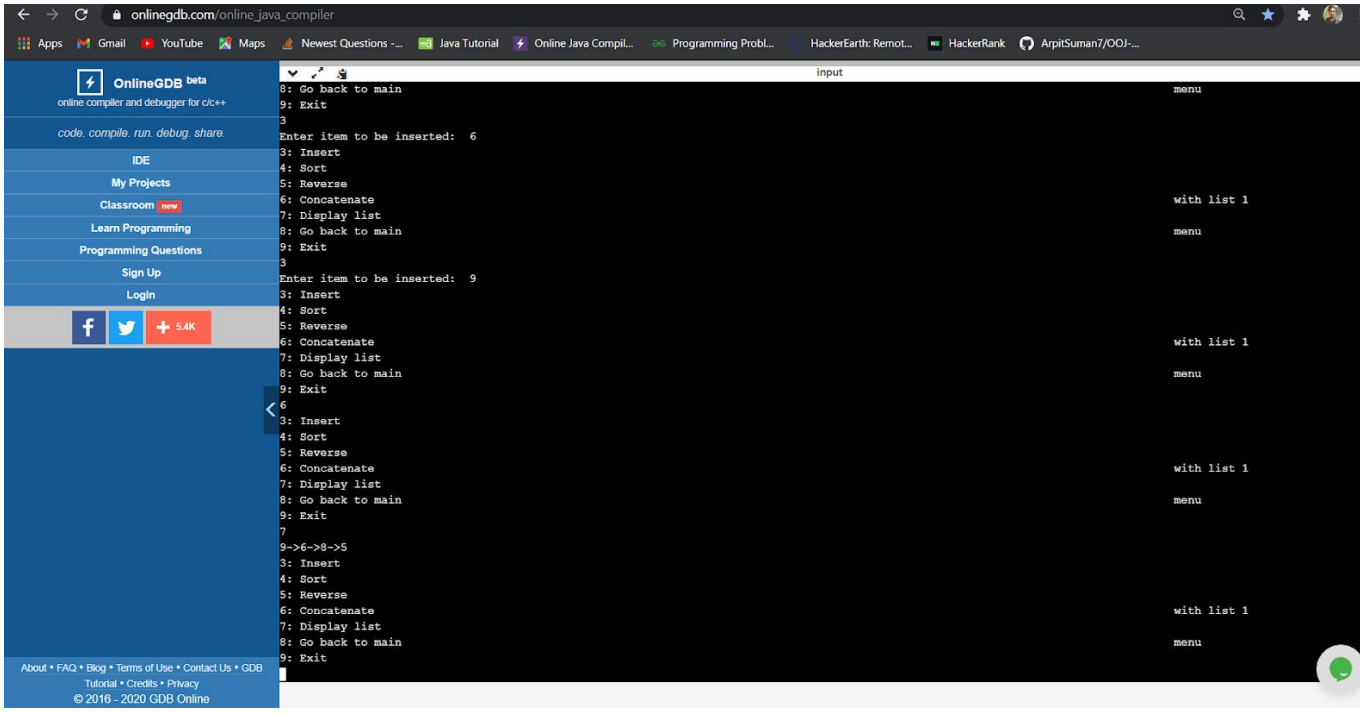
```
            {
               temp=temp->next;
            }

         }
      }
      if(del==NULL)
      {
         printf("\nElement not found.\n");
      }
}

void display()
{
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   temp=head;
   if(temp==NULL)
   {
      printf("\n List is empty \n");
   }
   else
   {
      printf("\nThe contents of the list are :\n");
      while(temp!=NULL)
      {
         printf("%d\n",temp->data);
         temp=temp->next;
      }
   }
```

```
}

int main()
{
   int choice,ele,pos;
   char ch;
   do
   {
   printf("\n1. Inset at end \n2.Insert at front \n3.Insert at random position \n4. Display \n5. Delete  \n6.exit");
   printf("\nEnter your choice : ");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1: printf("Enter the element to be inserted\n");
            scanf("%d",&ele);
            insertend(ele);
            break;
      case 2: printf("Enter the element to be inserted\n");
            scanf("%d",&ele);
            insertfront(ele);
            break;
      case 3: printf("Enter the element to be inserted\n");
            scanf("%d",&ele);
            printf("Enter the position \n");
            scanf("%d",&pos);
            insertrandom(ele,pos);
            break;
      case 4: display();
            break;
      case 5: printf("Enter the element to be deleted\n");
            scanf("%d",&ele);
```

```
            deleteele(ele);
            break;
    }
    }while(choice!=6);
    return 0;
}
```

# OUTPUT


onlinegdb.com/online_java_compiler
Apps Gmail YouTube Maps Newest Questions -... Java Tutorial Online Java Compil... Programming Probl... HackerEarth: Remot... HackerRank ArpitSuman7/OOJ-...
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
5.4K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
input
Linked list program containing sort, reverse, and concatenate functions.
Enter the choice
1.Stack
2.Queue
3: Linked list 1
4: Linked list 2
5: Exit
3
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
3
Enter item to be inserted: 5
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
3
Enter item to be inserted: 8
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
8
Enter the choice
1.Stack
2.Queue
3: Linked list 1


onlinegdb.com/online_java_compiler
Apps Gmail YouTube Maps Newest Questions -... Java Tutorial Online Java Compil... Programming Probl... HackerEarth: Remot... HackerRank ArpitSuman7/OOJ-...
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
5.4K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
input
8: Go back to main menu
9: Exit
3
Enter item to be inserted: 6
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
3
Enter item to be inserted: 9
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
6
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit
7
9->6->8->5
3: Insert
4: Sort
5: Reverse
6: Concatenate with list 1
7: Display list
8: Go back to main menu
9: Exit

**LAB PROGRAM 6**

**WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.**

PROGRAM

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

```
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first){
NODE temp;
if(first==NULL){
printf("List is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("Item deleted at front end is %d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE first,int item){
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first){
NODE cur,prev;
if(first==NULL){
printf("List is empty cannot delete\n");
```

```
return first;
}
if(first->link==NULL){
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL){
prev=cur;
cur=cur->link;
}
printf("Item deleted at rear end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE insert_pos(int item,int pos,NODE first){
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1){
return temp;
}
if(first==NULL){
printf("Invalid position\n");
return first;
}
if(pos==1){
temp->link=first;
```

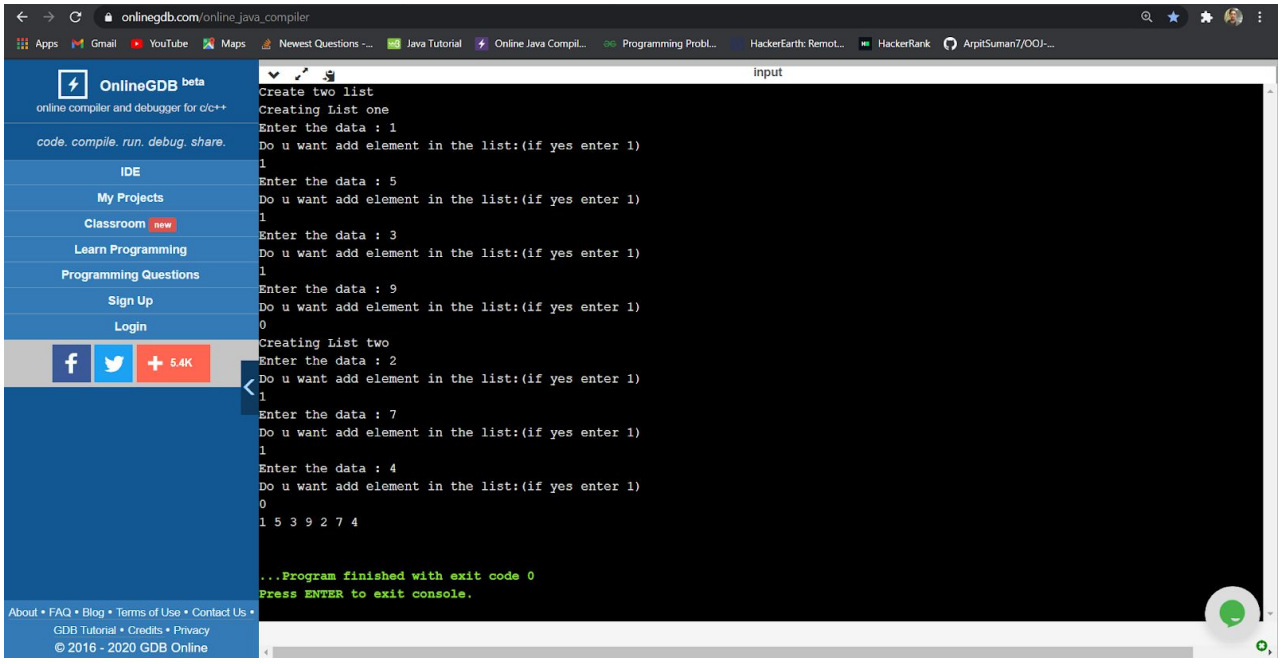
```
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos){
prev=cur;
cur=cur->link;
count++;
}
if(count==pos){
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid position\n");
return first;
}
NODE delete_pos(int pos,NODE first){
NODE cur;
NODE prev;
int count,flag=0;
if(first==NULL || pos<0){
printf("Invalid position\n");
return NULL;
}
if(pos==1){
cur=first;
first=first->link;
freenode(cur);
return first;
}
```

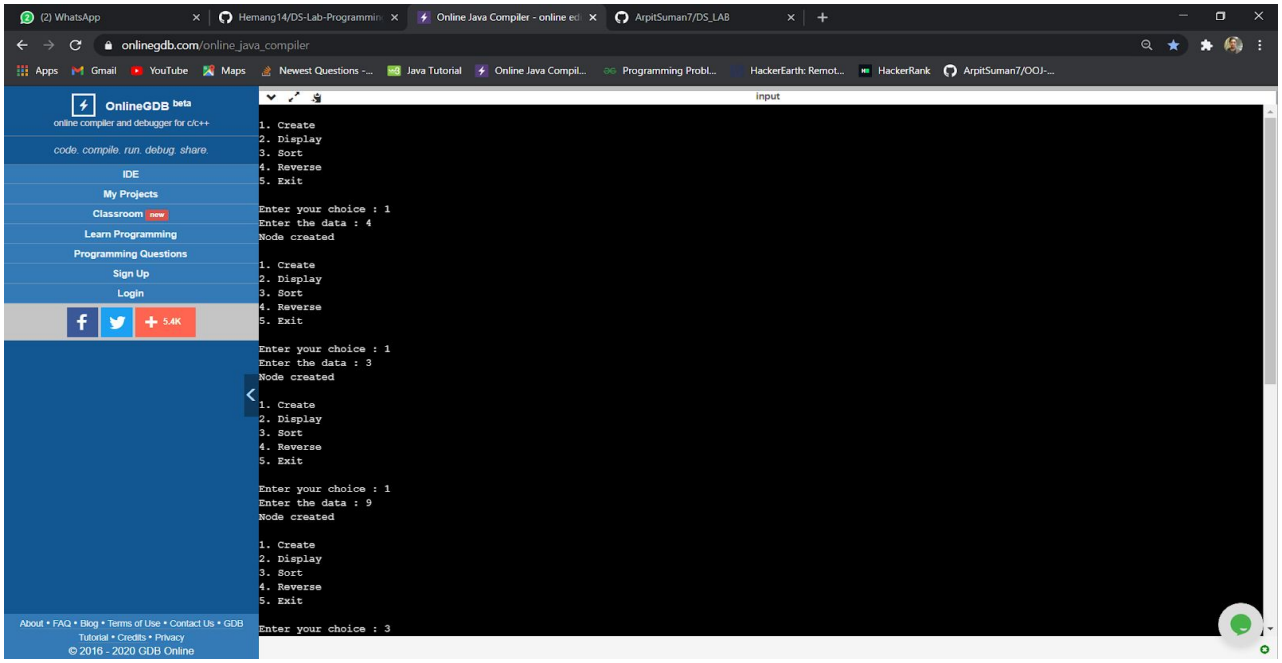
```
prev=NULL;
cur=first;
count=1;
while(cur!=NULL){
if(count==pos){
flag=1;
break;
}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0){
printf("Invalid position\n");
return first;
}
printf("Item deleted at given position is %d\n",cur->info);
prev->link=cur->link;
freenode(cur);
return first;
}
void display(NODE first){
NODE temp;
if(first==NULL)
printf("List empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link){
printf("%d\t",temp->info);
}
}
void main()
{
int item,choice,key,pos;
int count=0;
```

```
NODE first=NULL;
for(;;){
printf("\n1:Insert rear\n2:Delete rear\n3:Insert front\n4:Delete front\n5:Insert info position\n6:Delete info position\n7:Display list\n8:Exit\n");
printf("Enter the choice: ");
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_rear(first);
break;
case 3:printf("Enter the item at front end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 4:first=delete_front(first);
break;
case 5:printf("Enter the item to be inserted at given position\n");
scanf("%d",&item);
printf("Enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 6:printf("Enter the position\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;
case 7:display(first);
break;
default:exit(0);
break;
```

```
}
}
}
```

# OUTPUT


onlinegdb.com/online_java_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
5.4K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
input
Create two list
Creating List one
Enter the data : 1
Do u want add element in the list:(if yes enter 1)
1
Enter the data : 5
Do u want add element in the list:(if yes enter 1)
1
Enter the data : 3
Do u want add element in the list:(if yes enter 1)
1
Enter the data : 9
Do u want add element in the list:(if yes enter 1)
0
Creating List two
Enter the data : 2
Do u want add element in the list:(if yes enter 1)
1
Enter the data : 7
Do u want add element in the list:(if yes enter 1)
1
Enter the data : 4
Do u want add element in the list:(if yes enter 1)
0
1 5 3 9 2 7 4
...Program finished with exit code 0
Press ENTER to exit console.


onlinegdb.com/online_java_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
5.4K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
input
1. Create
2. Display
3. Sort
4. Reverse
5. Exit
Enter your choice : 1
Enter the data : 4
Node created
1. Create
2. Display
3. Sort
4. Reverse
5. Exit
Enter your choice : 1
Enter the data : 3
Node created
1. Create
2. Display
3. Sort
4. Reverse
5. Exit
Enter your choice : 1
Enter the data : 9
Node created
1. Create
2. Display
3. Sort
4. Reverse
5. Exit
Enter your choice : 3

**LAB PROGRAM 7**

**WAP Implement Single Link List with following operations a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists**

PROGRAM

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

struct node{
        int data;
        struct node*next;
};

void insertAtEnd(struct node**head,int d){
        struct node *temp,*n;
        if(*head == NULL){
                temp = (struct node*)malloc(sizeof(struct node));
                temp->data = d;
                temp->next = NULL;
                *head = temp;
        }
        else {
                temp = *head;
                //go to the last node
                while(temp->next!=NULL){
                        temp = temp->next;
                }
                //adding node at the end
                n = (struct node*)malloc(sizeof(struct node));
                n->data = d;
```

```
                n->next = NULL;
                temp->next = n;
        }
}
void reverse(struct node**head) {
        struct node *prev,*cur,*next1;
        cur = *head;
        prev= NULL;
      next1=NULL;
        if(*head == NULL) {
                printf("Empty LIST\n");
                return;
        }

        while(cur!=NULL) {
      next1=cur->next;
      cur->next=prev;
      prev=cur;
                cur=next1;
        }
        *head = prev;
}

void concat(struct node**head1,struct node**head2){
        if(*head1==NULL) {
                *head1 = *head2;
                return;
        }
        if(*head2==NULL) {
                *head2 = *head1;
                return;
```

```
        }
        struct node*temp = *head1;
        while(temp->next!=NULL) {
                temp = temp->next;
        }
        temp->next = *head2;
}

struct node* merger(struct node*a,struct node*b) {
        //base case
        if(a==NULL) {
                return b;
        }
        if(b==NULL) {
                return a;
        }

        struct node*c = NULL;
        //rec case
        if(a->data < b->data) {
                c = a;
                c->next = merger(a->next,b);
        }
        else{
                c = b;
                c->next = merger(a,b->next);
        }
        return c;
}

struct node* MidPoint(struct node*head){
```

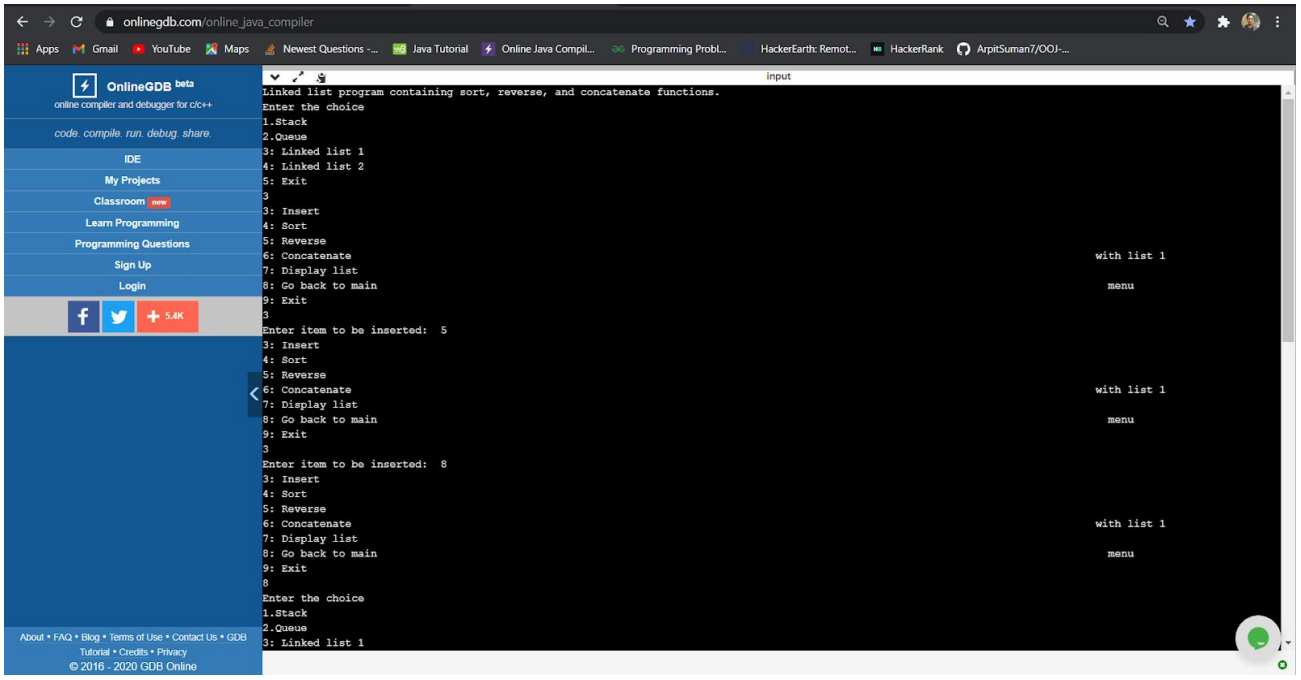
```
        if(head == NULL || head->next == NULL){
                return head;
        }

        struct node*fast = head->next;
        struct node*slow = head;

        while(fast != NULL && fast->next != NULL){
                fast = fast->next->next;
                slow = slow->next;
        }
        return slow;
}

struct node* MergeSort(struct node*head){
        if(head == NULL || head->next == NULL) {
                return head;
        }
        //rec case
        //1. Breaking into 2
        struct node* mid = MidPoint(head);
        struct node*a = head;
        struct node*b = mid->next;

        mid->next = NULL;

        //2. rec sort the two parts
        a = MergeSort(a);
        b = MergeSort(b);

        //3. Merging them
```

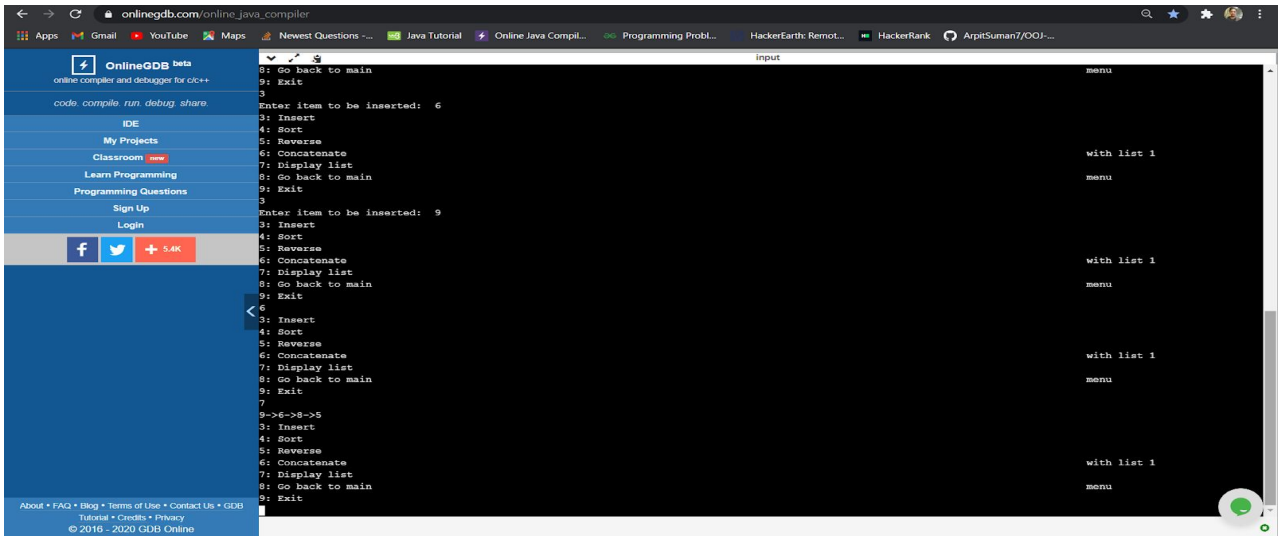
```
	struct node* c = merger(a,b);
	return c;
}

void display(struct node *head){
	while(head!=NULL){
		printf("%d-->",head->data);
		head = head->next;
	}
	printf("\n");
}


int main()
{
	struct node
*head1=NULL,*head2=NULL,*head3=NULL,*head4=NULL,*ans=NULL;
	int data,n;
	printf("----SORTING----\n");
	printf("Enter the list to be sorted(Enter -1 to stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head1,data);
		scanf("%d",&data);
	}
	printf("List before sorting: ");
	display(head1);
	ans = MergeSort(head1);
	printf("List after sorting: ");
	display(ans);

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

```
printf("Enter the list to be reversed(Enter -1 to stop): \n");
scanf("%d",&data);
while(data!=-1) {
        insertAtEnd(&head2,data);
        scanf("%d",&data);
}
printf("List before reversing: ");
display(head2);
reverse(&head2);
printf("List after reversing: ");
display(head2);

printf("\n----CONCATENATION----\n");
printf("Enter the first list(Enter -1 to stop): \n");
scanf("%d",&data);
while(data!=-1) {
        insertAtEnd(&head3,data);
        scanf("%d",&data);
}
printf("Enter the second list(Enter -1 to stop): \n");
scanf("%d",&data);
while(data!=-1) {
        insertAtEnd(&head4,data);
        scanf("%d",&data);
}

printf("First List: ");
display(head3);
printf("Second List: ");
display(head4);
concat(&head3,&head4);
```

```
        printf("Concatenated List: ");
        display(head3);
    return 0;
}
```

## OUTPUT



**LAB PROGRAM 8**

**WAP to implement Stack & Queues using Linked Representation**

PROGRAM

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

struct node{
        int data;
        struct node*next;
};
struct node*front;
struct node*rear;

void push(struct node**top,int d) {
        struct node*temp,n;

        temp = (struct node*)malloc(sizeof(struct node));

        if(temp == NULL) {
                printf("Stack is full\n");
        }

        temp->data = d;
        temp->next = *top;
        *top = temp;
        printf("%d is pushed\n",d);
}

void pop(struct node**top) {
```

```
	struct node*temp;

	if(*top==NULL) {
		printf("Stack Underflow\n");
		return;
	}

	temp = *top;
	printf("%d poped\n",temp->data);
	*top = (*top)->next;

	free(temp);
}

void display(struct node* top) {
	if(top == NULL){
		printf("No Elements Present in Stack\n");
		return;
	}

	while(top!=NULL) {
		printf("%d ",top->data);
		top = top->next;
	}
	printf("\n");
}

void insert(int d) {
	struct node*n;
	n = (struct node*)malloc(sizeof(struct node));
	if(n == NULL){
```

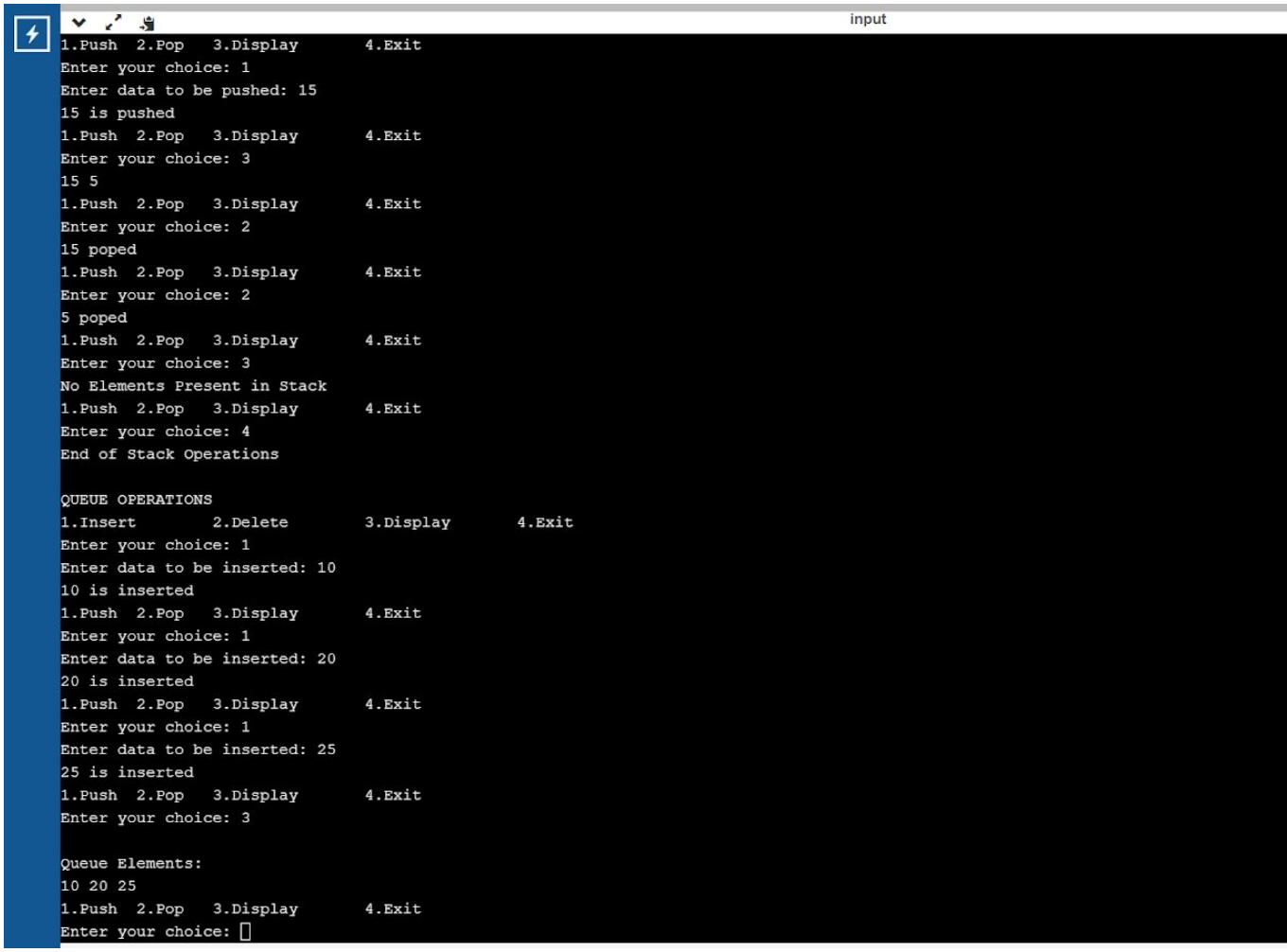
```
                printf("Queue Overflow\n");
                return;
        }
        n->data = d;
        if(front==NULL) {
                front = n;
                rear = n;
                front->next = NULL;
                rear->next = NULL;
        }
        else {
                rear->next = n;
                rear = n;
                rear->next = NULL;
        }
        printf("%d is inserted\n",d);
}

void delete() {
        struct node*temp;
        if(front == NULL) {
                printf("Queue Underflow\n");
                return;
        }
        temp = front;
        printf("%d deleted\n",temp->data);
        front = front->next;
        free(temp);
}

void display_queue() {
```

```
    struct node *temp;
    temp = front;
    if(front == NULL)
    {
       printf("\nEmpty queue\n");
    }
    else
    {   printf("\nQueue Elements: \n");
       while(temp != NULL)
       {
          printf("%d ",temp -> data);
          temp = temp -> next;
       }
       printf("\n");
    }
}

int main() {
          struct node*stack = NULL;
          printf("STACK OPERATIONS\n");
          printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
          int choice,item;
          printf("Enter your choice: ");
          scanf("%d",&choice);
          while(choice!=4) {
                    switch(choice) {
                              case 1: printf("Enter data to be pushed: ");
                                        scanf("%d",&item);
                                        push(&stack,item);
                                        break;
```

```
                case 2: pop(&stack);
                                break;

                case 3: display(stack);
                                break;
        }
        printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
}
printf("End of Stack Operations\n\n");

printf("QUEUE OPERATIONS\n");
printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
printf("Enter your choice: ");
scanf("%d",&choice);
while(choice!=4) {
        switch(choice) {
                case 1: printf("Enter data to be inserted: ");
                                scanf("%d",&item);
                                insert(item);
                                break;

                case 2: delete();
                                break;

                case 3: display_queue();
                                break;
        }
        printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
```

```
            printf("Enter your choice: ");
            scanf("%d",&choice);
        }
        printf("End Of Queue Operations\n");
        return 0;
}
```

## OUTPUT


input
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 15
15 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
15 5
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
15 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
5 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
No Elements Present in Stack
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 4
End of Stack Operations
QUEUE OPERATIONS
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 10
10 is inserted
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 20
20 is inserted
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 25
25 is inserted
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
Queue Elements:
10 20 25
1.Push 2.Pop 3.Display 4.Exit
Enter your choice:

**LAB PROGRAM 9**

**WAP Implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list**

PROGRAM

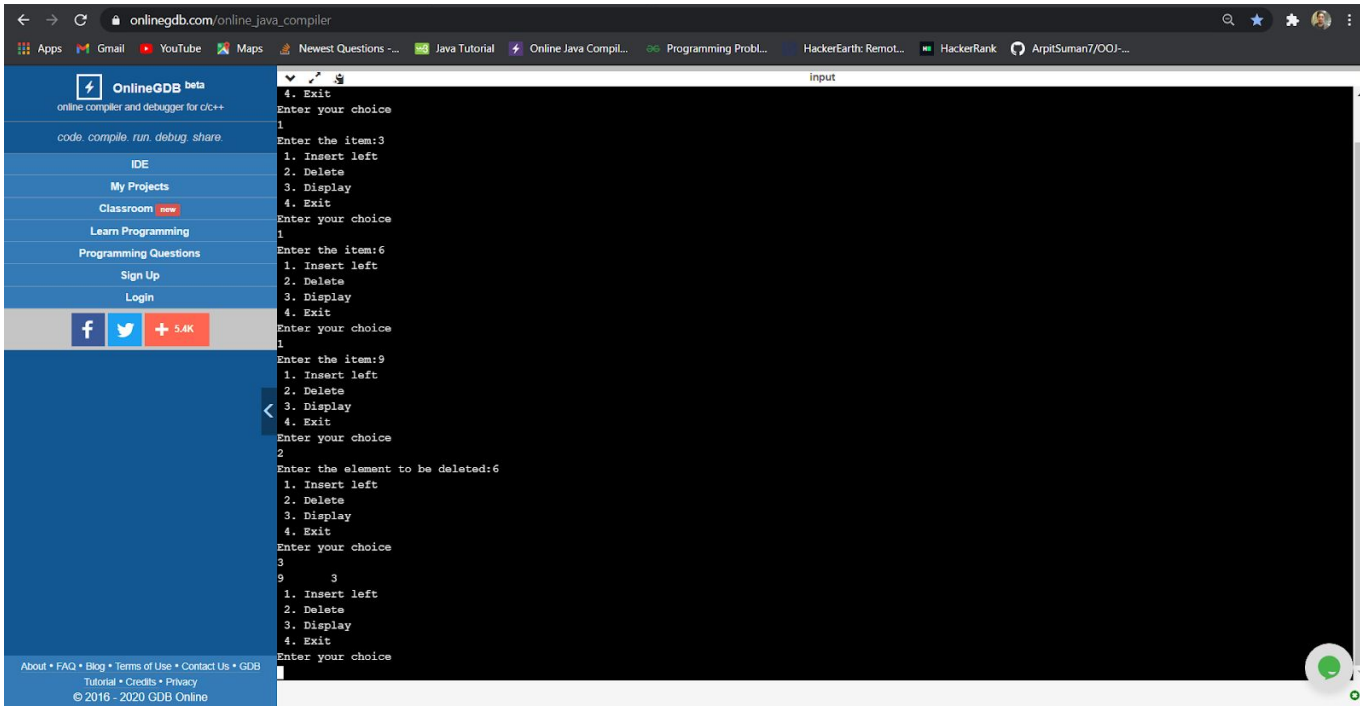
```
#include<stdio.h>
#include<stdlib.h>
void insert_left();
void del();
void display();
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head=NULL;
int main()
{
int choice;
while(choice!=4)
{
printf(" 1. Insert left \n");
printf(" 2. Delete \n");
printf(" 3. Display\n");
printf(" 4. Exit\n");
printf("Enter your choice\n");
scanf("%d",&choice);
if(choice==1)
            insert_left();
         else if(choice==2)
```

```
            del();
        else if(choice==3)
            display();
        else if(choice==4)
            break;
}
return 0;
}
void insert_left()
{
struct node *new_node;
new_node=(struct node*)malloc(sizeof(struct node));
printf("Enter the item:");
scanf("%d",&new_node->data);
new_node->next=NULL;
new_node->prev=NULL;
if(head==NULL)
{
head=new_node;
}
else
{
new_node->next=head;
head->prev=new_node;
head=new_node;
}
}
void del()
{
struct node *temp;
int ele;
```

```
   if(head==NULL)
   {
      printf("Empty List \n");
      return;
   }
printf("Enter the element to be deleted:");
scanf("%d",&ele);
temp=head;
while(temp->data!=ele)
{
temp=temp->next;
if(temp==NULL)
{
printf("Element is not in the list\n");
break;
}
}
if(temp==head)
{
head=head->next;
}
else if(temp->next==NULL)
{
temp=temp->prev;
temp->next=NULL;
}
else
{
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
}
```

```
}
void display()
{
struct node *temp;
temp=head;
while(temp!=NULL)
{
printf("%d\t",temp->data);
temp=temp->next;
}
printf("\n");
}
```

## OUTPUT



**LAB PROGRAM 10**

**Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree**

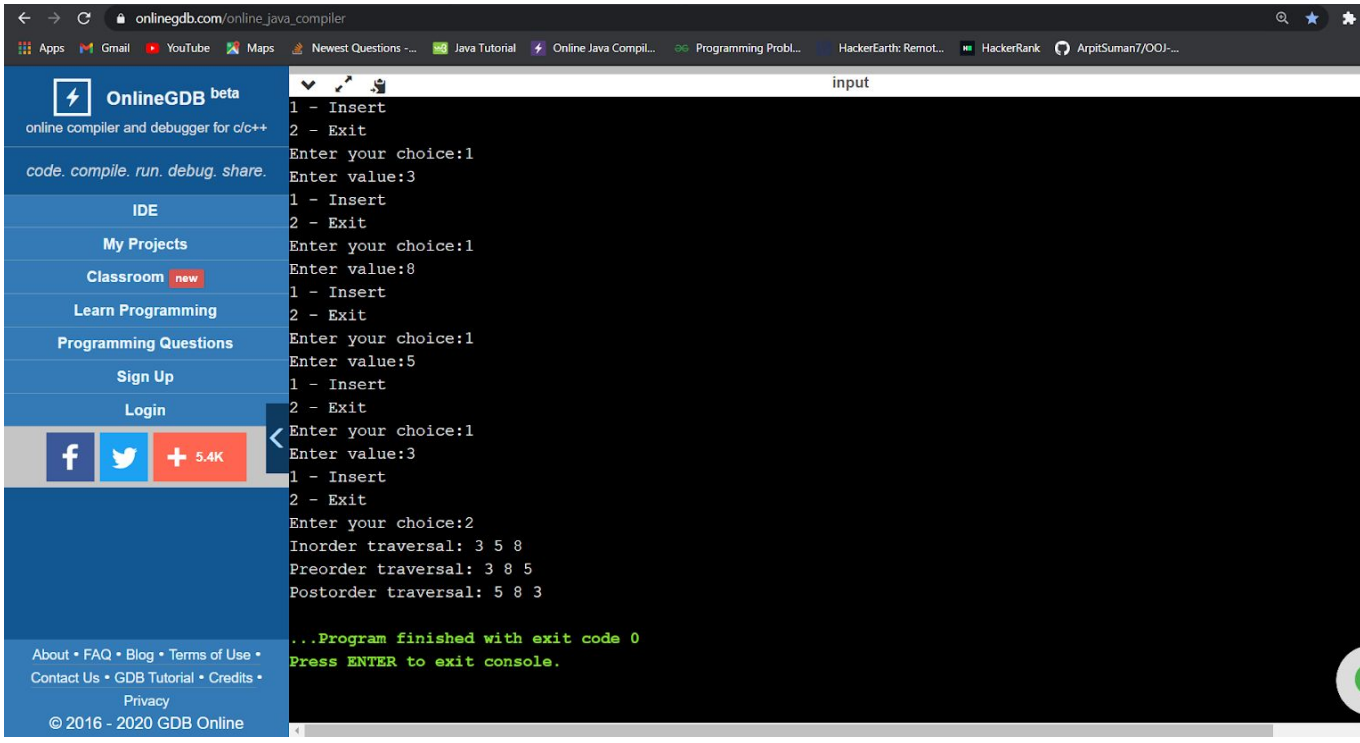
PROGRAM

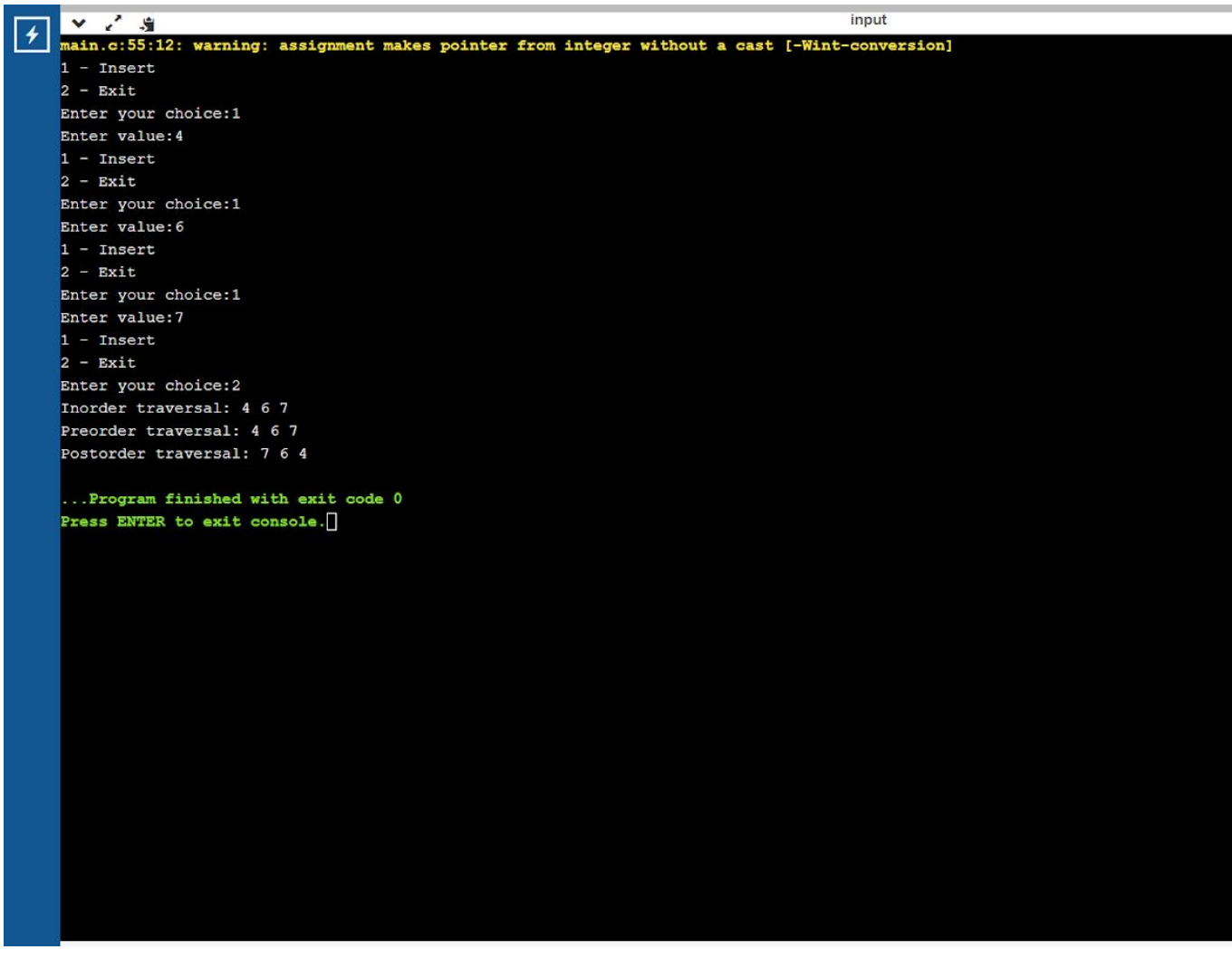
```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
  int data;
  struct Node *left, *right;
} node;
node *create(int data) {
  node *temp;
  temp = (node*)malloc(sizeof(node));
  temp->data = data;
  temp->left = temp->right = NULL;
  return temp;
}
void inorder(node *root) {
  if (root != NULL) {
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
  }
}
void preorder(node *root) {
  if (root != NULL) {
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
```

```
  }
}
void postorder(node *root) {
  if (root != NULL) {
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
  }
}
void insert(node *root, node *temp) {
  if(temp->data<root->data){
    if(root->left!=NULL)
        insert(root->left,temp);
    else
        root->left = temp;
  }
  if(temp->data>root->data)
  {
      if(root->right!=NULL)
        insert(root->right,temp);
      else
        root->right=temp;
  }
}
int main(void) {
  node *root = NULL,*temp;
  int choice = 0;
  while(choice != 2)
  {
      temp =
```

```
    printf("1 - Insert\n");
    printf("2 - Exit\n");
    printf("Enter your choice:");
    scanf("%d",&choice);
    if(choice==1)
    {
       int val;
       printf("Enter value:");
       scanf("%d",&val);
       temp = create(val);
       if(root==NULL)
         root=temp;
       else
         insert(root,temp);
    }
    else if(choice==2)
      break;
    else
      printf("Invalid choice\n");
  }
  printf("Inorder traversal: ");
  inorder(root);
  printf("\nPreorder traversal: ");
  preorder(root);
  printf("\nPostorder traversal: ");
  postorder(root);
}
```

# OUTPUT


onlinegdb.com/online_java_compiler
Apps
Gmail
YouTube
Maps
Newest Questions -...
Java Tutorial
Online Java Compil...
Programming Probl...
HackerEarth: Remot...
HackerRank
ArpitSuman7/OOJ-...
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
5.4K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
input
1 - Insert
2 - Exit
Enter your choice:1
Enter value:3
1 - Insert
2 - Exit
Enter your choice:1
Enter value:8
1 - Insert
2 - Exit
Enter your choice:1
Enter value:5
1 - Insert
2 - Exit
Enter your choice:1
Enter value:3
1 - Insert
2 - Exit
Enter your choice:2
Inorder traversal: 3 5 8
Preorder traversal: 3 8 5
Postorder traversal: 5 8 3
...Program finished with exit code 0
Press ENTER to exit console.


input
main.c:55:12: warning: assignment makes pointer from integer without a cast [-Wint-conversion]
1 - Insert
2 - Exit
Enter your choice:1
Enter value:4
1 - Insert
2 - Exit
Enter your choice:1
Enter value:6
1 - Insert
2 - Exit
Enter your choice:1
Enter value:7
1 - Insert
2 - Exit
Enter your choice:2
Inorder traversal: 4 6 7
Preorder traversal: 4 6 7
Postorder traversal: 7 6 4
...Program finished with exit code 0
Press ENTER to exit console.