//QUESTION-1

//write a c program to reverse a string using stack

#include<stdio.h>

#include<string.h>

#define n 100

int top=-1,stack[n];

void push(char x){

if(top==n-1){

printf("stack overflow");

}else{

top++;

stack[top]=x;

}

}

void pop(){

printf("%c",stack[top--]);

}

int main(){

char str[]="Hello amaravati";

int i;

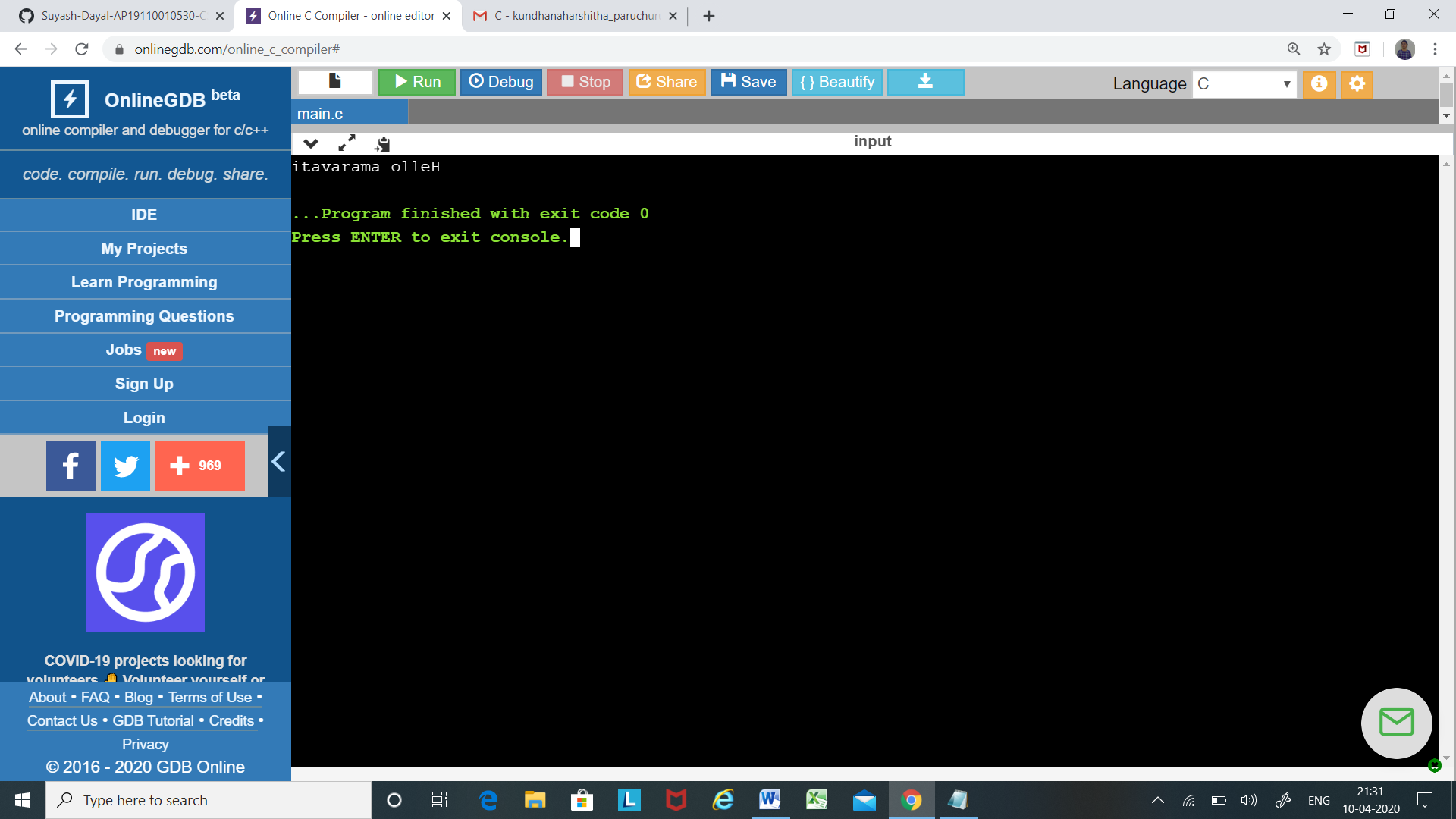
for(i=0;i<strlen(str);i++)

push(str[i]);

for(i=0;i<strlen(str);i++)

pop();

}



//QUESTION-2

//Write a program for Infix to Postfix Conversion Using Stack.

#include <stdio.h>

#include <ctype.h>

#define SIZE 50

char stack[SIZE];

int top=-1;

push(char elem){

stack[++top]=elem;

}

char pop(){

return(stack[top--]);

}

int pr(char symbol){

if(symbol == '^'){

return(3);

}else if(symbol == '\*' || symbol == '/'){

return(2);

}else if(symbol == '+' || symbol == '-'){

return(1);

}else{

return(0);

}

}

void main(){

char infix[50],postfix[50],ch,elem;

int i=0,k=0;

printf("Enter Infix Expression : ");

scanf("%s",infix);

push('#');

while( (ch=infix[i++]) != '\0'){

if( ch == '(') push(ch);

else

if(isalnum(ch)) postfix[k++]=ch;

else

if( ch == ')'){

while( stack[top] != '(')

postfix[k++]=pop();

elem=pop();

}else{

while( pr(stack[top]) >= pr(ch) )

postfix[k++]=pop();

push(ch);

}

}

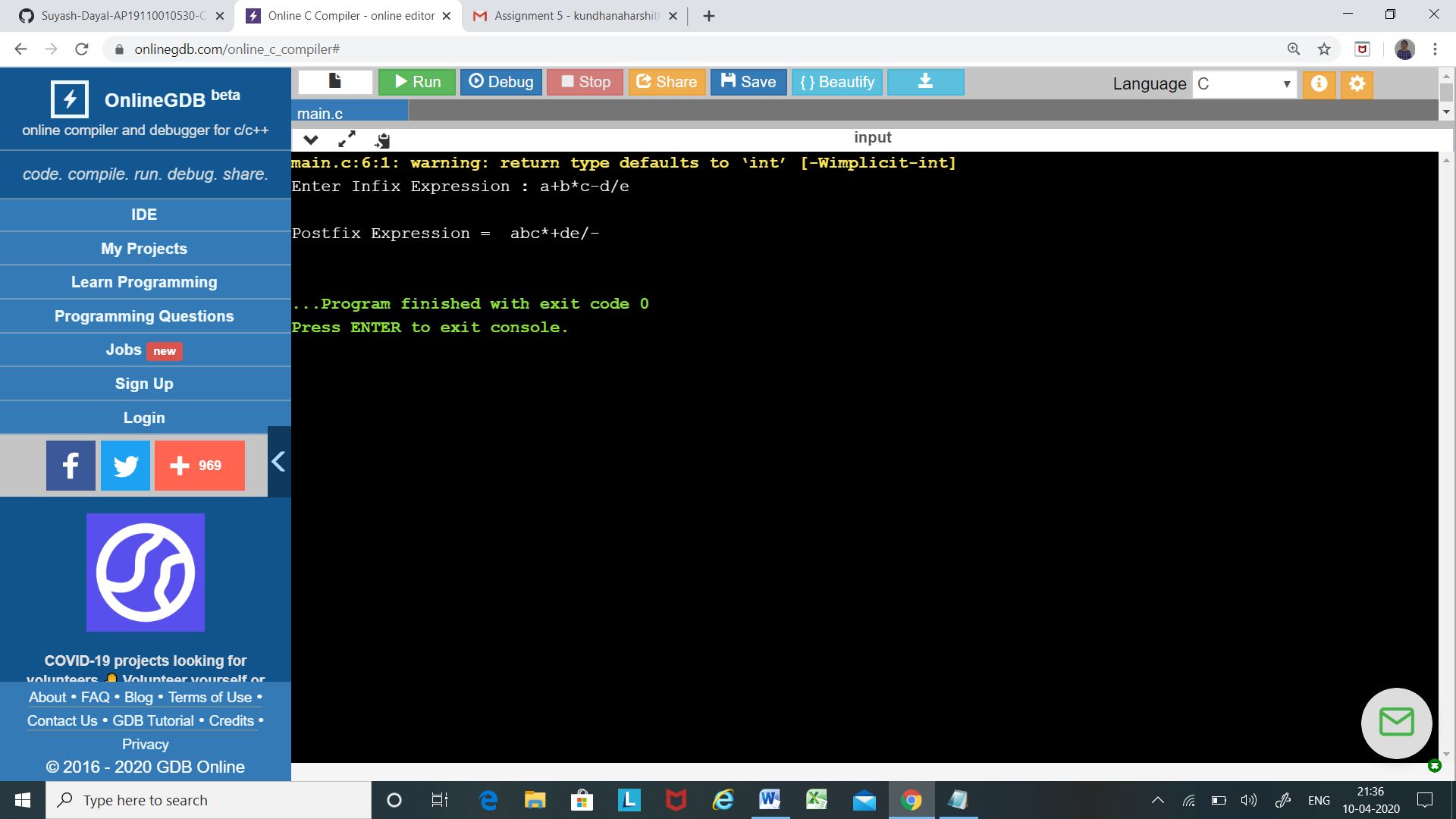
while( stack[top] != '#')

postfix[k++]=pop();

postfix[k]='\0';

printf("\nPostfix Expression = %s\n",postfix);

}



//QUESTION-3

/\* C program to implement queues using two stacks \*/

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*next;

};

void push(struct node\*\* top, int data);

int pop(struct node\*\* top);

struct queue{

struct node \*stack1;

struct node \*stack2;

};

void enqueue(struct queue \*q, int x){

push(&q->stack1, x);

}

void dequeue(struct queue \*q){

int x;

if (q->stack1 == NULL && q->stack2 == NULL) {

printf("queue is empty");

return;

}

if (q->stack2 == NULL) {

while (q->stack1 != NULL) {

x = pop(&q->stack1);

push(&q->stack2, x);

}

}

x = pop(&q->stack2);

printf("Deleted element is %d\n", x);

}

void push(struct node\*\* top, int data){

struct node\* newnode = (struct node\*) malloc(sizeof(struct node));

if (newnode == NULL) {

printf("Stack overflow \n");

return;

}

newnode->data = data;

newnode->next = (\*top);

(\*top) = newnode;

}

int pop(struct node\*\* top){

int buff;

struct node \*t;

if (\*top == NULL) {

printf("Stack underflow \n");

return 0;

}else {

t = \*top;

buff = t->data;

\*top = t->next;

free(t);

return buff;

}

}

void display(struct node \*top1,struct node \*top2){

while (top1 != NULL) {

printf("%d\n", top1->data);

top1 = top1->next;

}while (top2 != NULL) {

printf("%d\n", top2->data);

top2 = top2->next;

}

}

int main(){

struct queue \*q = (struct queue\*)malloc(sizeof(struct queue));

int f = 0, a;

char ch = 'y';

q->stack1 = NULL;

q->stack2 = NULL;

while (ch == 'y'||ch == 'Y') {

printf("enter ur choice\n1.add to queue 2.remove from queue 3.display 4.exit: ");

scanf("%d",&f);

switch(f){

case 1 : printf("enter the element to be added to queue: ");

scanf("%d", &a);

enqueue(q, a);

break;

case 2 : dequeue(q);

break;

case 3 : display(q->stack1, q->stack2);

break;

case 4 : exit(1);

break;

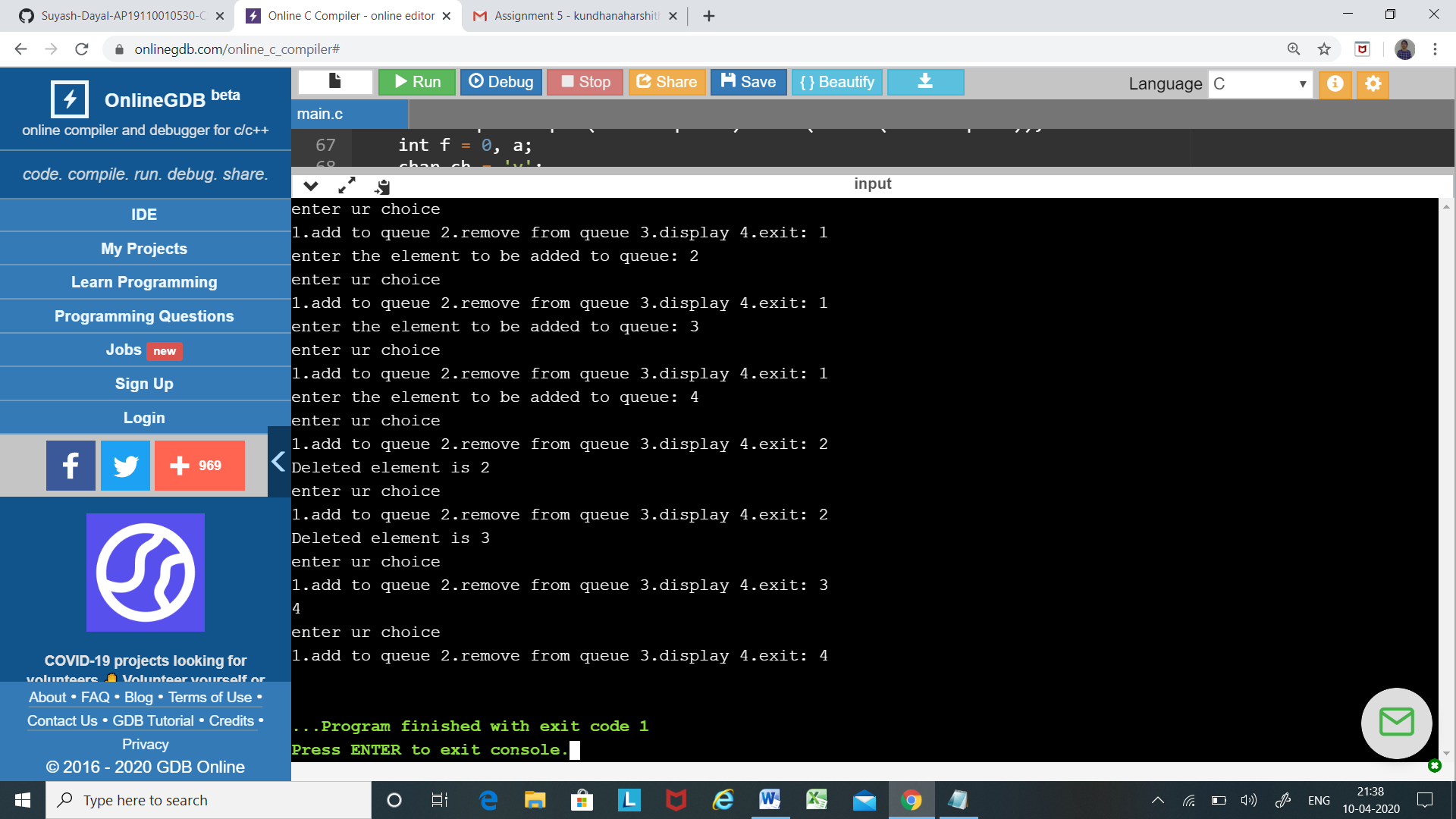
default : printf("invalid\n");

break;

}

}

}



//QUESTION-4

//Write a c program for insertion and deletion of BST.

#include<stdio.h>

#include<stdlib.h>

struct node{

int key;

struct node \*left,\*right;

};

struct node \*newNode(int item){

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

temp->key=item;

temp->left=temp->right=NULL;

return temp;

}

void inorder(struct node \*root){

if(root!=NULL){

inorder(root->left);

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

inorder(root->right);

}

}

struct node \*insert(struct node\* node,int key){

if(node==NULL){

return newNode(key);

}

if(key<node->key){

node->left=insert(node->left,key);

}

else if(key>node->key){

node->right=insert(node->right,key);

}

return node;

}

struct node \*minNode(struct node \*node){

struct node \*g=node;

while (g&&g->left!=NULL)

g=g->left;

return g;

}

struct node\*deleteNode(struct node\*root,int key){

if(root==NULL)

return root;

if(key<root->key){

root->left=deleteNode(root->left,key);

}else if(key>root->key){

root->right=deleteNode(root->right,key);

}else{

if(root->left==NULL){

struct node \*temp=root->right;

free(root);

return temp;

}

else if(root->right==NULL){

struct node \*temp=root->left;

free(root);

return temp;

}

struct node \*temp=minNode(root->right);

root->key=temp->key;

root->right=deleteNode(root->right,temp->key);

}

return root;

}

void main(){

struct node \*root=NULL;

root=insert(root,100);

insert(root,60);

insert(root,40);

insert(root,90);

insert(root,220);

insert(root,180);

insert(root,800);

inorder(root);

printf("\nDeleted number 90\n");

root=deleteNode(root,90);

printf("Inorder transversal of the modified tree is :\n");

inorder(root);

}

