Day 3: DATA STRUCTURES PROGRAMS

1.write a c program to implement linked list data structure.

Program:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insert(struct Node\*\* head, int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void deleteElement(struct Node\*\* head, int value) {

if (\*head == NULL) {

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

return;

}

struct Node\* current = \*head;

struct Node\* previous = NULL;

while (current != NULL && current->data != value) {

previous = current;

current = current->next;

}

if (current == NULL) {

printf("Element not found in the list.\n");

return;

}

if (previous == NULL) {

\*head = current->next;

} else {

previous->next = current->next;

}

free(current);

printf("Element deleted successfully.\n");

}

void display(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main() {

struct Node\* head = NULL;

int choice, value;

while (1) {

printf("1. Insert an element\n");

printf("2. Delete an element\n");

printf("3. Display the list\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to insert: ");

scanf("%d", &value);

insert(&head, value);

break;

case 2:

printf("Enter the value to delete: ");

scanf("%d", &value);

deleteElement(&head, value);

break;

case 3:

display(head);

break;

case 4:

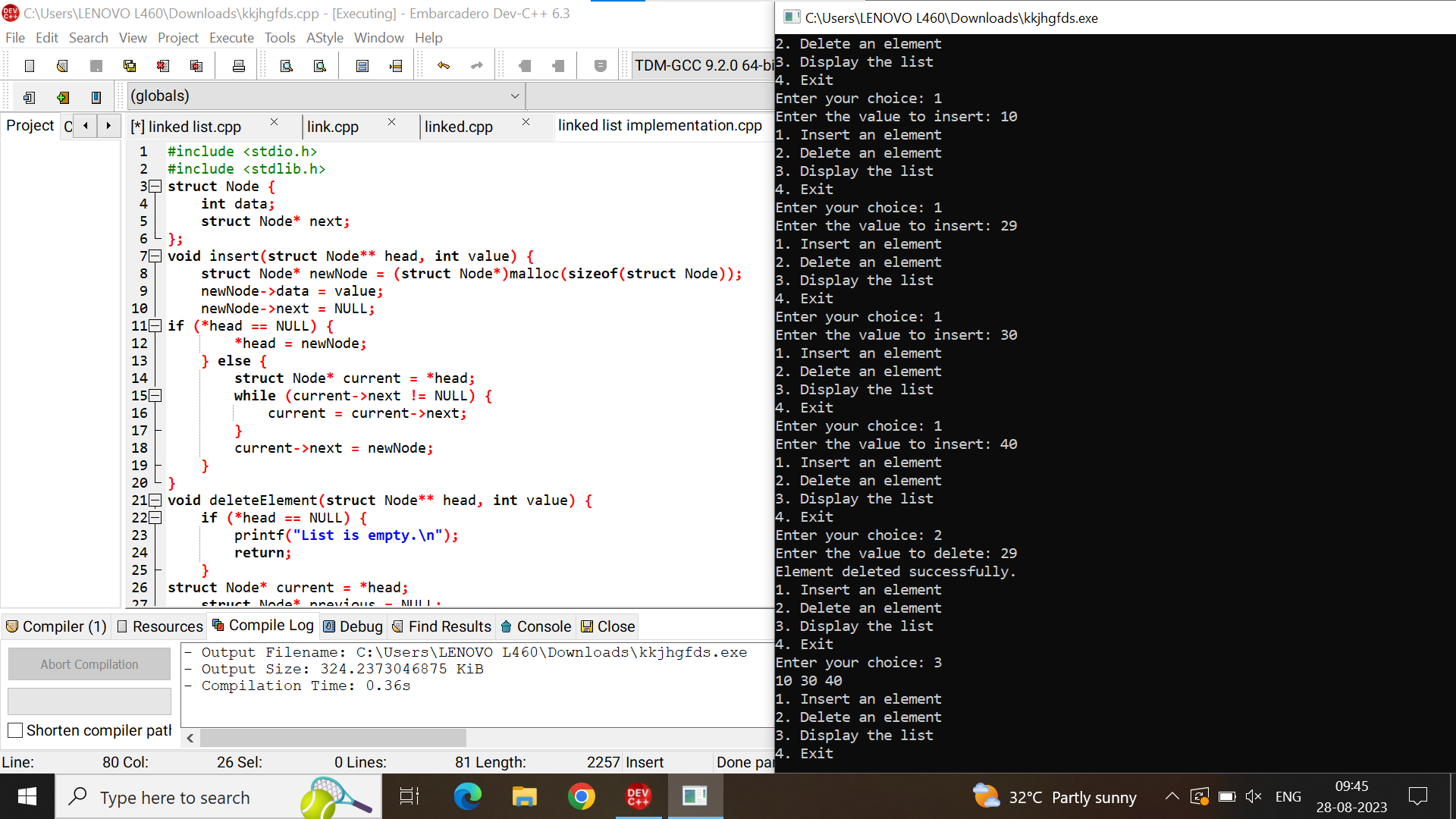
exit(0);

default:

printf("Invalid choice. Please try again.\n");}}

return 0;

}



Sample input: enter the choice:1

Enter the value to insert:10

29

30

40

Enter the choice:2

Enter the value to delete:29

Output: display the list : 10

30

40

2.write a c program to merge two list.

Program:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertEnd(struct Node\*\* head, int data) {

struct Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

struct Node\* mergeLists(struct Node\* list1, struct Node\* list2) {

if (list1 == NULL) {

return list2;

}

if (list2 == NULL) {

return list1;

}

if (list1->data < list2->data) {

list1->next = mergeLists(list1->next, list2);

return list1;

} else {

list2->next = mergeLists(list1, list2->next);

return list2;

}

}

void displayList(struct Node\* head) {

struct Node\* temp = head;

while (temp != NULL) {

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

int main() {

struct Node\* list1 = NULL;

struct Node\* list2 = NULL;

int n, m;

printf("Enter the size of list 1: ");

scanf("%d", &n);

printf("Enter the elements of list 1:\n");

for (int i = 0; i < n; i++) {

int num;

scanf("%d", &num);

insertEnd(&list1, num);

}

printf("Enter the size of list 2: ");

scanf("%d", &m);

printf("Enter the elements of list 2:\n");

for (int i = 0; i < m; i++) {

int num;

scanf("%d", &num);

insertEnd(&list2, num);

}

printf("List 1: ");

displayList(list1);

printf("List 2: ");

displayList(list2);

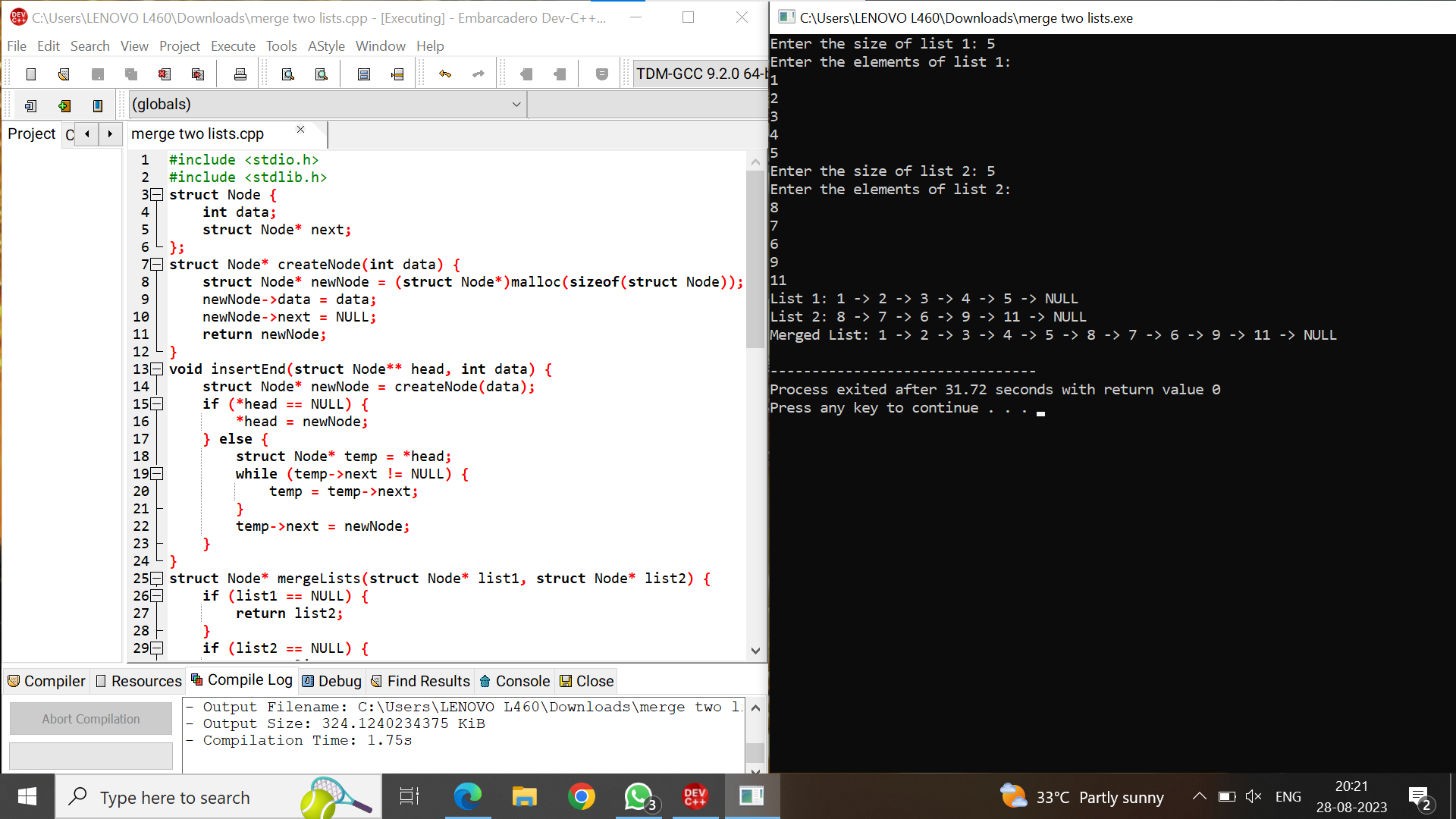
struct Node\* mergedList = mergeLists(list1, list2);

printf("Merged List: ");

displayList(mergedList);

return 0;

}

Sample input: size of list1:5 = 1 2 3 4 5 , list2: 5 = 8 7 6 9 11

Output: 1=> 2=> 3=> 4=> 5=> 8=> 7=> 6=> 9=> 11=> null.

3.write a c program to implement stack operstions.

Program:

#include <stdio.h>

#include <stdlib.h>

#define SIZE 4

int top = -1, inp\_array[SIZE];

void push();

void pop();

void show();

int main()

{

int choice;

while (1)

{

printf("\nPerform operations on the stack:");0`

printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");

printf("\n\nEnter the choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

show();

break;

case 4:

exit(0);

default:

printf("\nInvalid choice!!");

}

}

}

void push()

{

int x;

if (top == SIZE - 1)

{

printf("\nOverflow!!");

}

else

{

printf("\nEnter the element to be added onto the stack: ");

scanf("%d", &x);

top = top + 1;

inp\_array[top] = x;

}

}

void pop()

{

if (top == -1)

{

printf("\nUnderflow!!");

}

else

{

printf("\nPopped element: %d", inp\_array[top]);

top = top - 1;

}

}

void show()

{

if (top == -1)

{

printf("\nUnderflow!!");

}

else

{

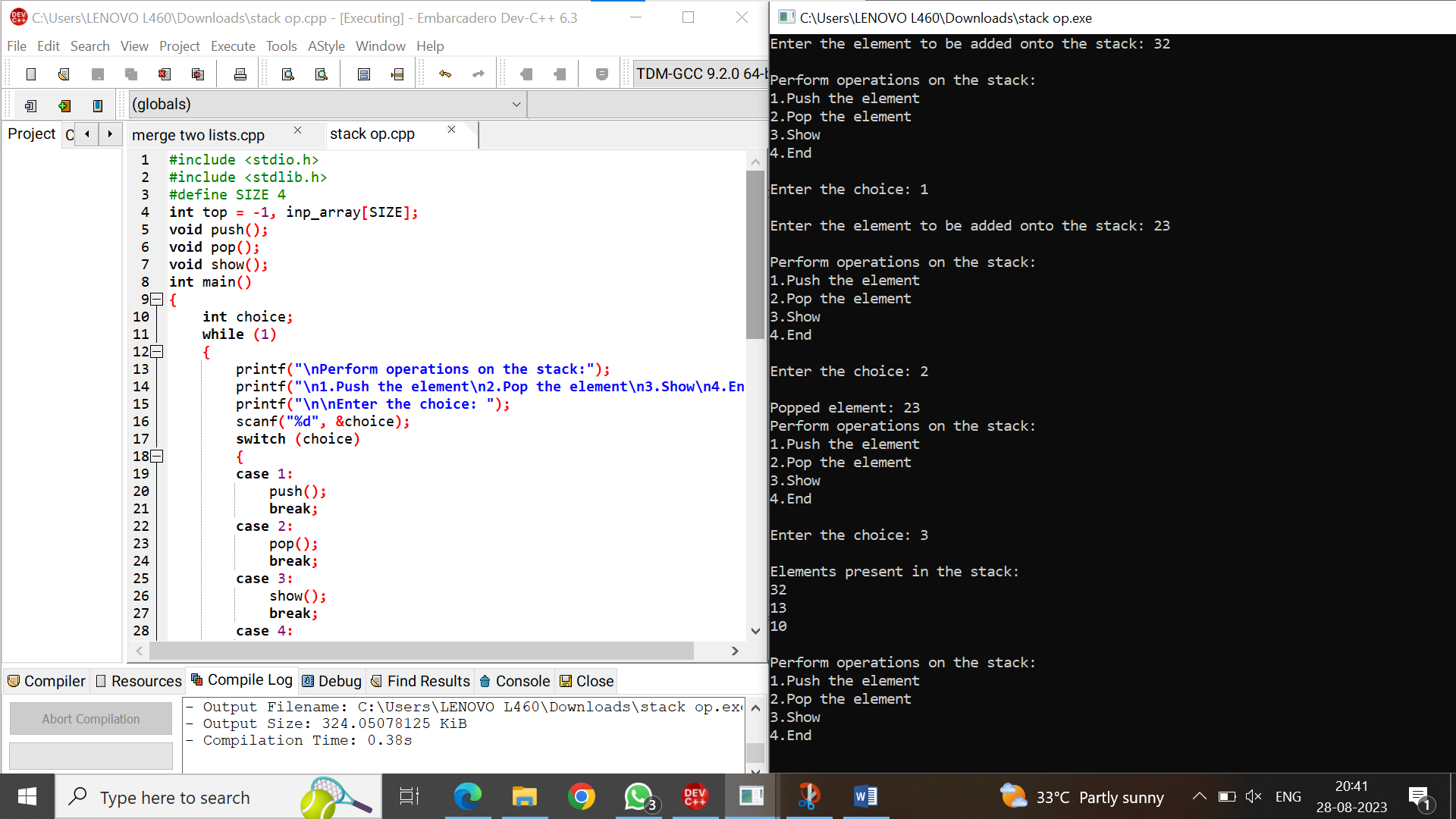
printf("\nElements present in the stack: \n");

for (int i = top; i >= 0; --i)

printf("%d\n", inp\_array[i]);

}

}



Sample input: push elements = 10,13,20,32,43 pop element 43

Output: elements present in the stack :32 , 20 , 13, 10.

4.write a c program to implement queue data structure.

Program:

#include<stdio.h>

#define n 5

int main()

{

int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;

printf("Queue using Array");

printf("\n1.enqueue \n2.Dequeue \n3.Display \n4.Exit");

while(ch)

{

printf("\nEnter the Choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

if(rear==x)

printf("\n Queue is Full");

else

{

printf("\n Enter no %d:",j++);

scanf("%d",&queue[rear++]);

}

break;

case 2:

if(front==rear)

{

printf("\n Queue is empty");

}

else

{

printf("\n Deleted Element is %d",queue[front++]);

x++;

}

break;

case 3:

printf("\nQueue Elements are:\n ");

if(front==rear)

printf("\n Queue is Empty");

else

{

for(i=front; i<rear; i++)

{

printf("%d",queue[i]);

printf("\n");

}

break;

case 4:

exit(0);

default:

printf("Wrong Choice: please see the options");

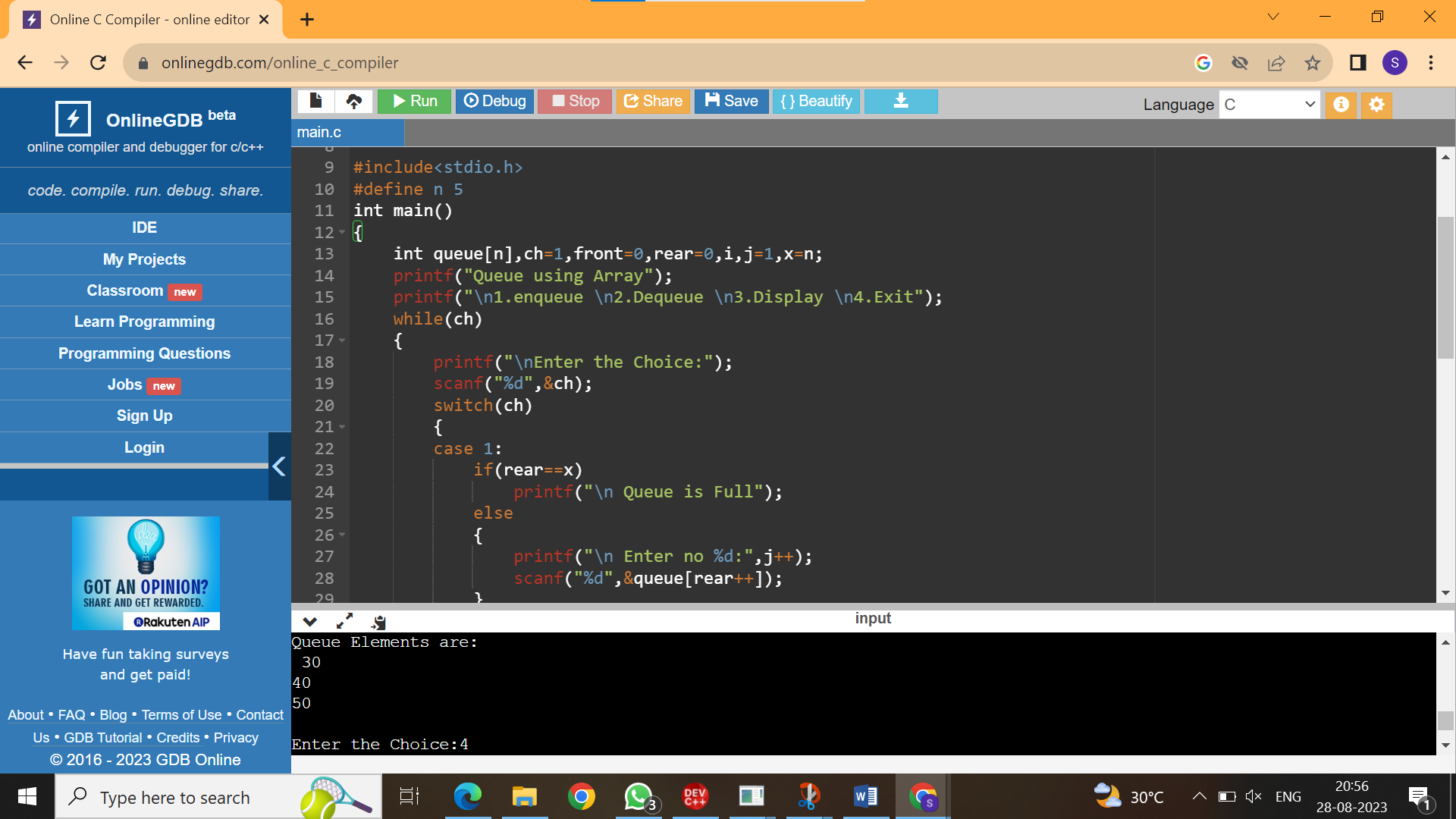
}

}

}

return 0;

}



Sample input:

Enqueue elements: 10 20 30 40 50

Dequeue elements: 10 20

Output: display the elements: 30

40

50

5.write a c program to convert infix to postfix using stack.

Program:

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[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 \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

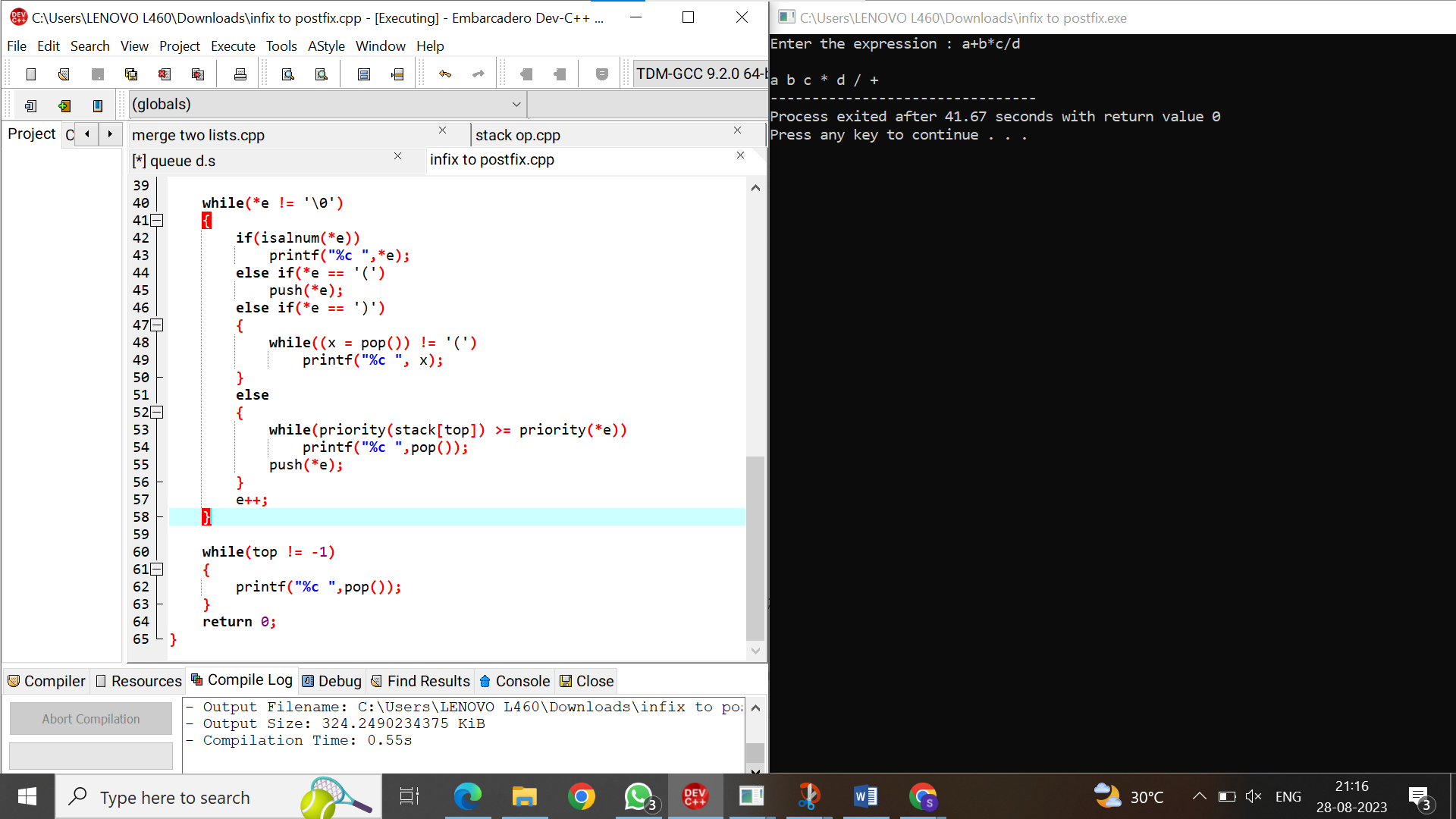
{

printf("%c ",pop());

}

return 0;

}



Sample input:

Expression : a+b\*c/d

Output:

abc\*d/+

6.write a c program to evaluate the postfix expression.

Program: #include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

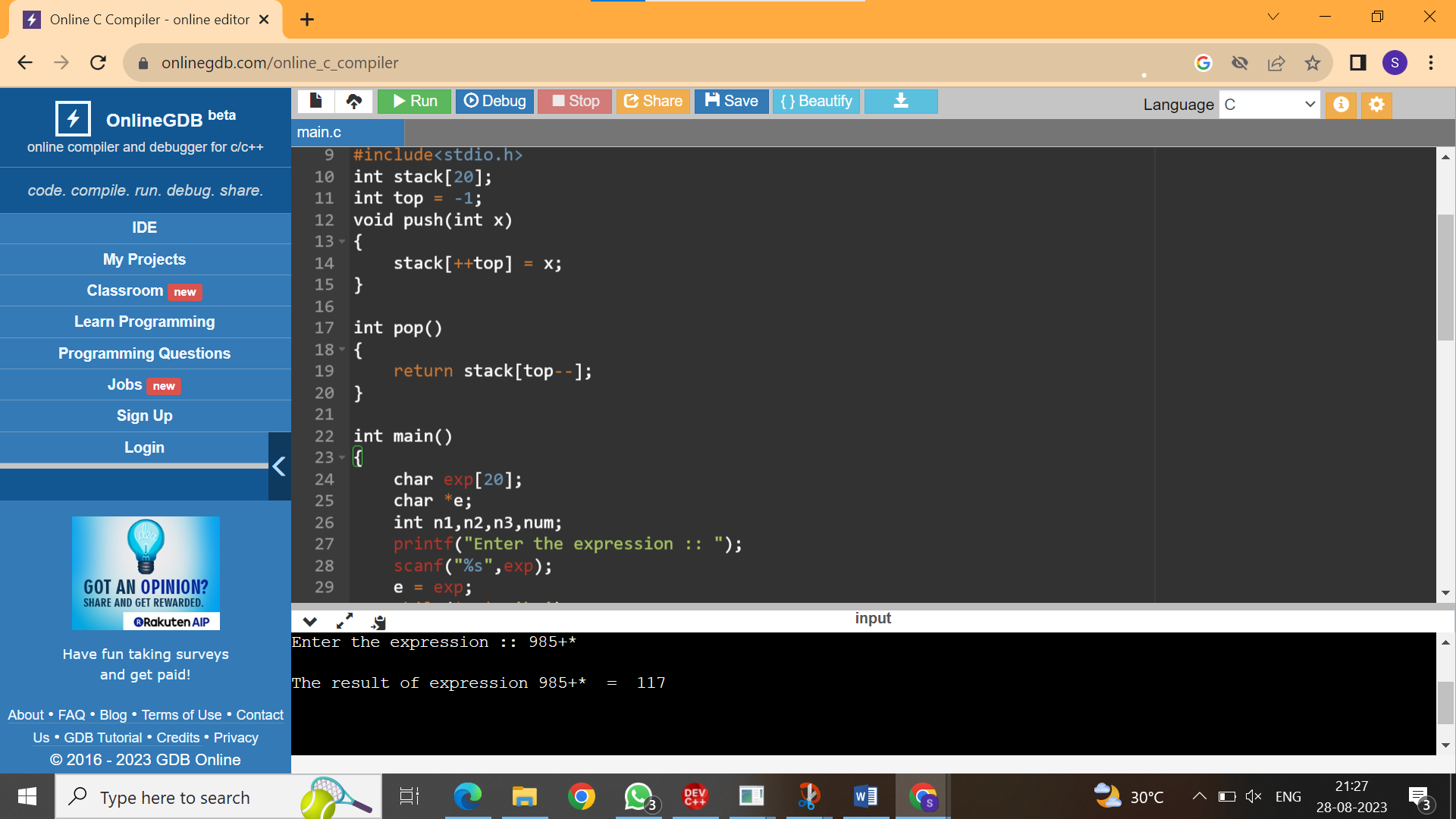
e++;

}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

return 0;

}



Sample input: 985+\*

Output: 117

7.write a c program to implement tree traversals.

Program:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

void inorder(struct Node\* root) {

if (root == NULL)

return;

inorder(root->left);

printf("%d ", root->data);

inorder(root->right);

}

void preorder(struct Node\* root) {

if (root == NULL)

return;

printf("%d ", root->data);

preorder(root->left);

preorder(root->right);

}

void postorder(struct Node\* root) {

if (root == NULL)

return;

postorder(root->left);

postorder(root->right);

printf("%d ", root->data);

}

int main() {

struct Node\* root = createNode(1);

root->left = createNode(2);

root->right = createNode(3);

root->left->left = createNode(4);

root->left->right = createNode(5);

printf("Inorder traversal: ");

inorder(root);

printf("\n");

printf("Preorder traversal: ");

preorder(root);

printf("\n");

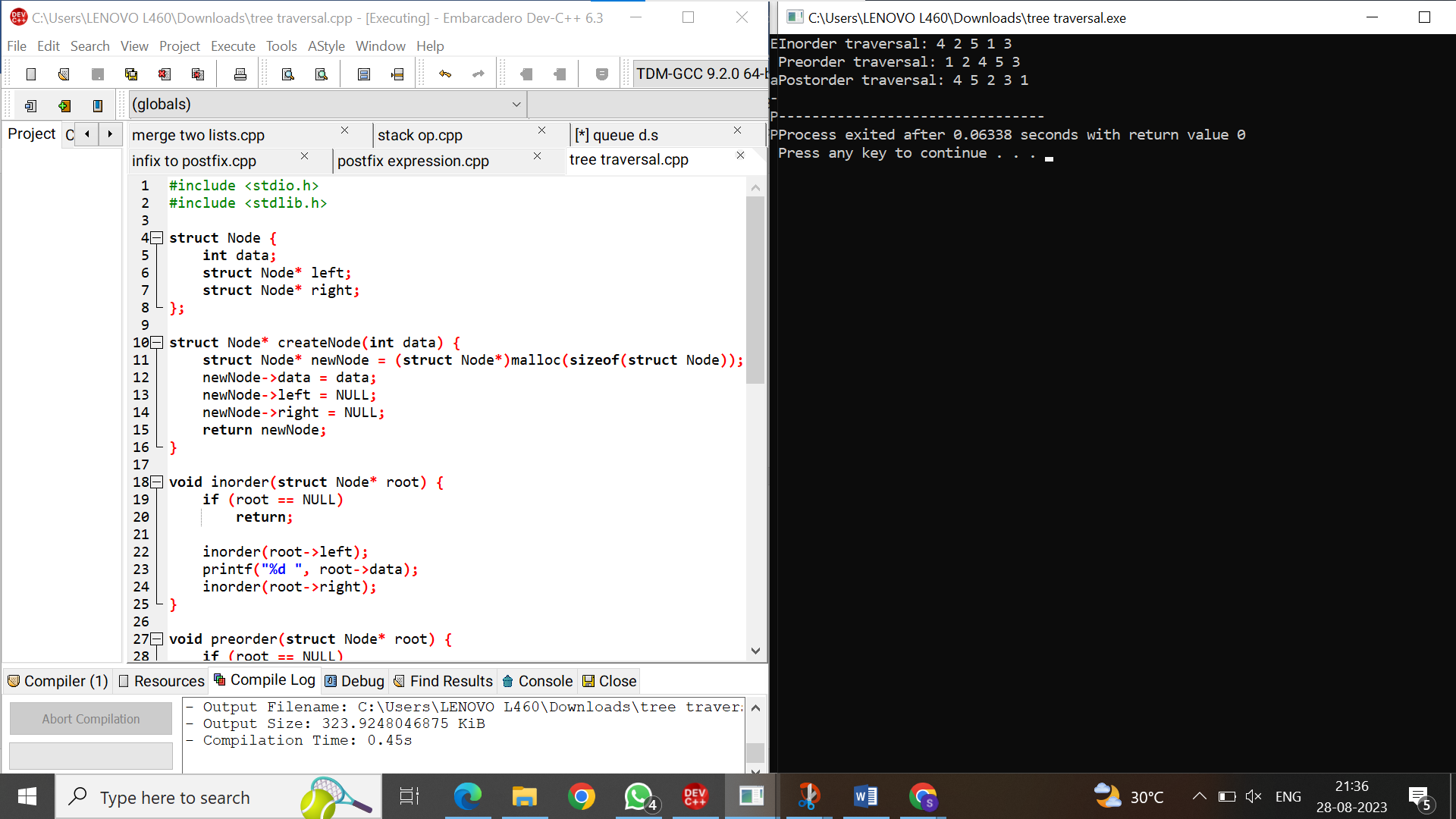
printf("Postorder traversal: ");

postorder(root);

printf("\n");

return 0;

}



Sample input:

Inorder traversal

Preorder traversal

Postorder traversal

Output:

Inorder: 4 2 5 1 3

Preorder: 1 2 4 5 3

Postorder: 4 5 2 3 1