Day 3- Data Structures

1.

#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));

if (newNode == NULL) {

printf("Memory allocation failed!\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

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

struct Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void display(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

printf("NULL\n");

}

void freeList(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

struct Node\* temp = current;

current = current->next;

free(temp);

}

}

int main() {

struct Node\* head = NULL;

int numNodes, data;

printf("Enter the number of nodes: ");

scanf("%d", &numNodes);

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

printf("Enter data for node %d: ", i + 1);

scanf("%d", &data);

insertAtEnd(&head, data);

}

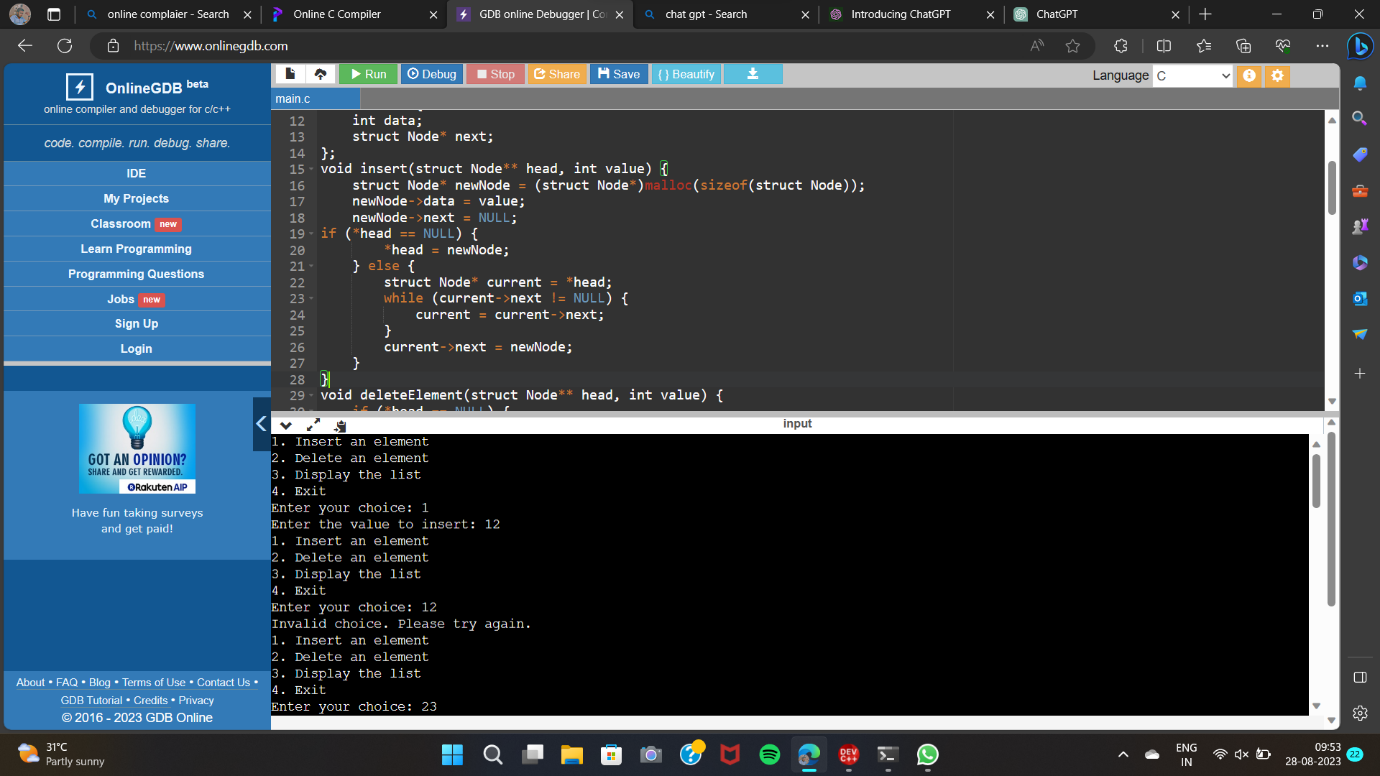
printf("Linked list: ");

display(head);

freeList(head);

    return 0;

}



2.

#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);

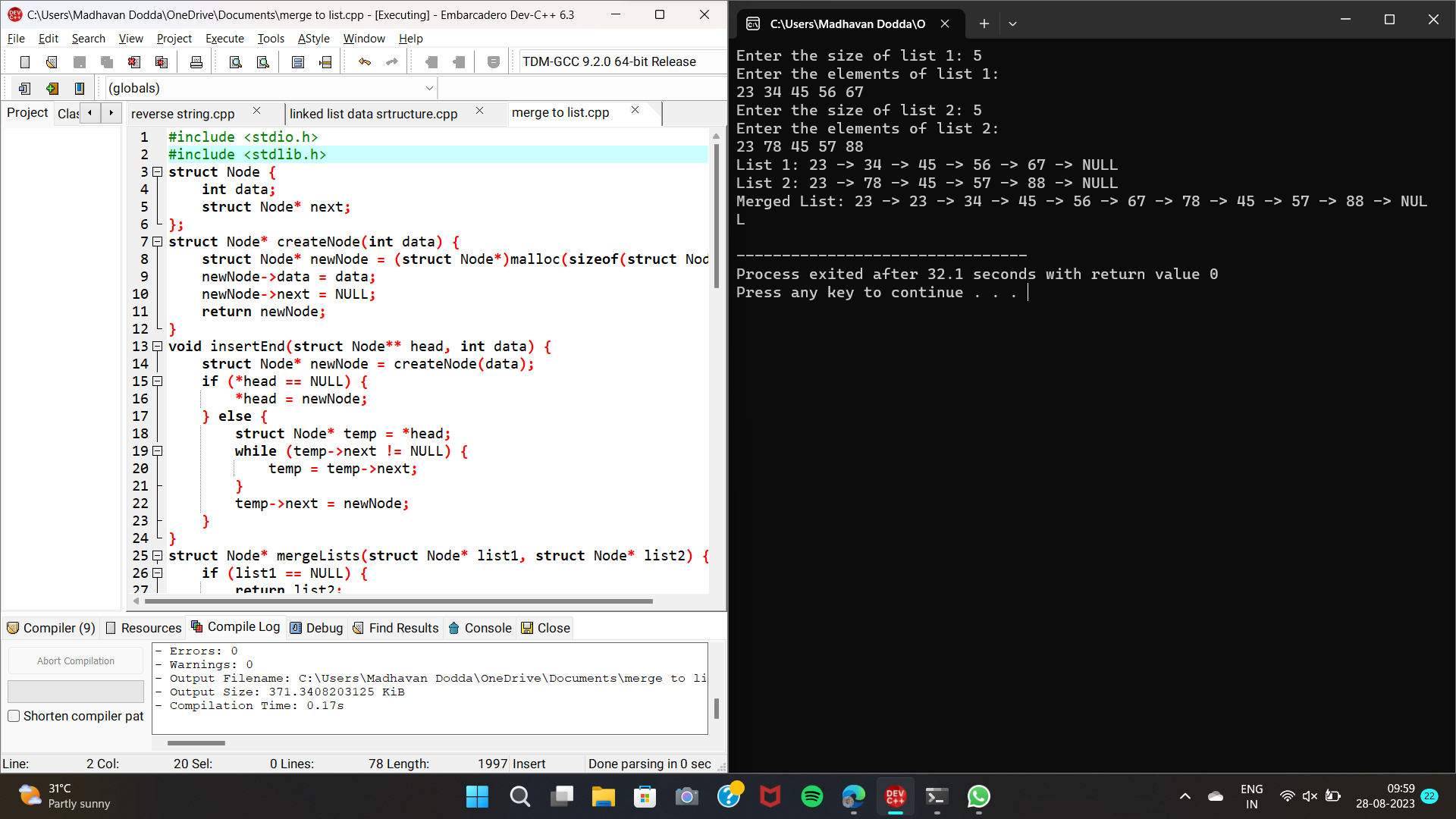
struct Node\* mergedList = mergeLists(list1, list2);

printf("Merged List: ");

displayList(mergedList);

return 0;

}



3. #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:");

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

{

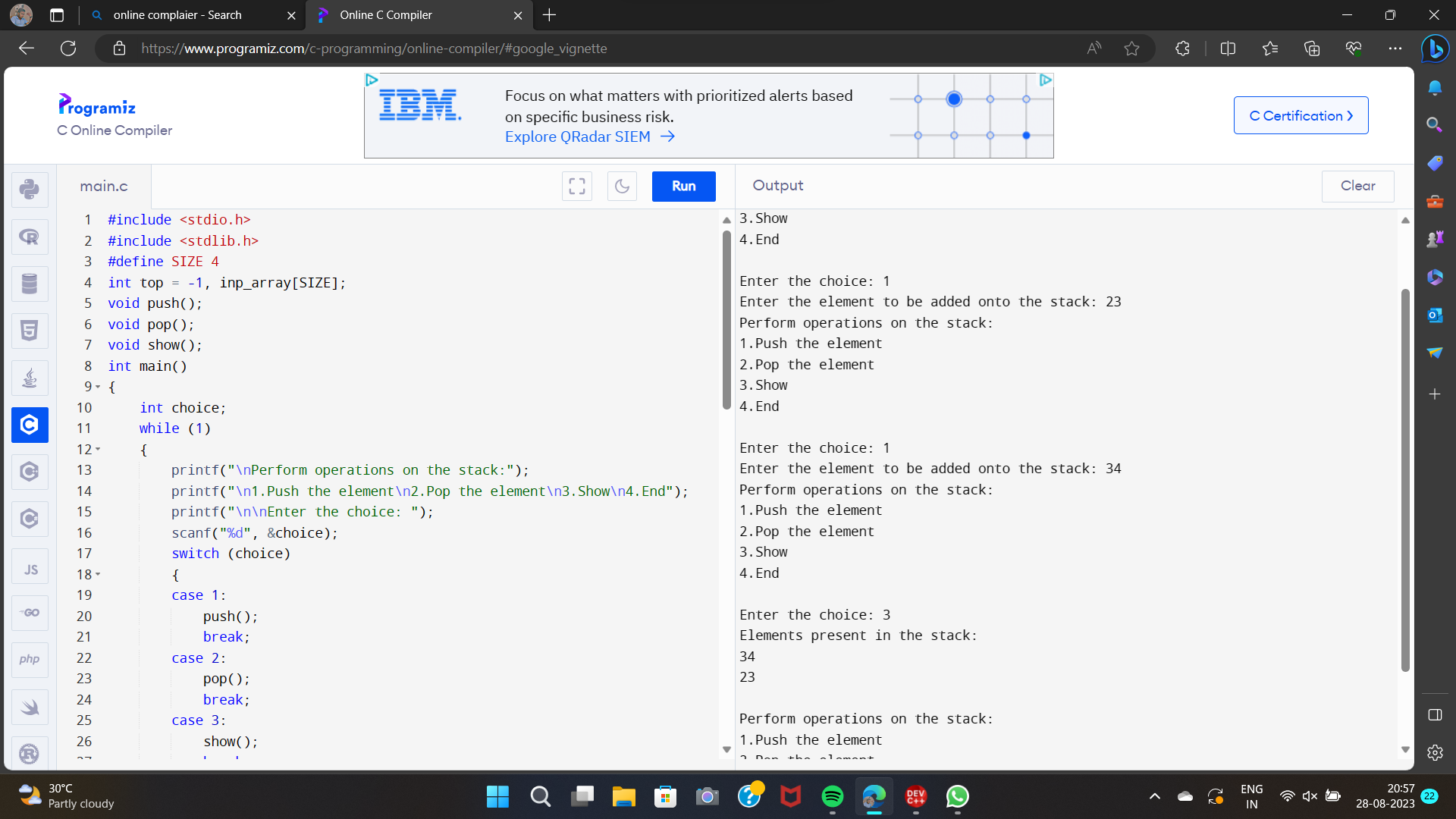
printf("\nElements present in the stack: \n");

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

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

}

}



4. #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.Insertion \n2.Deletion \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");

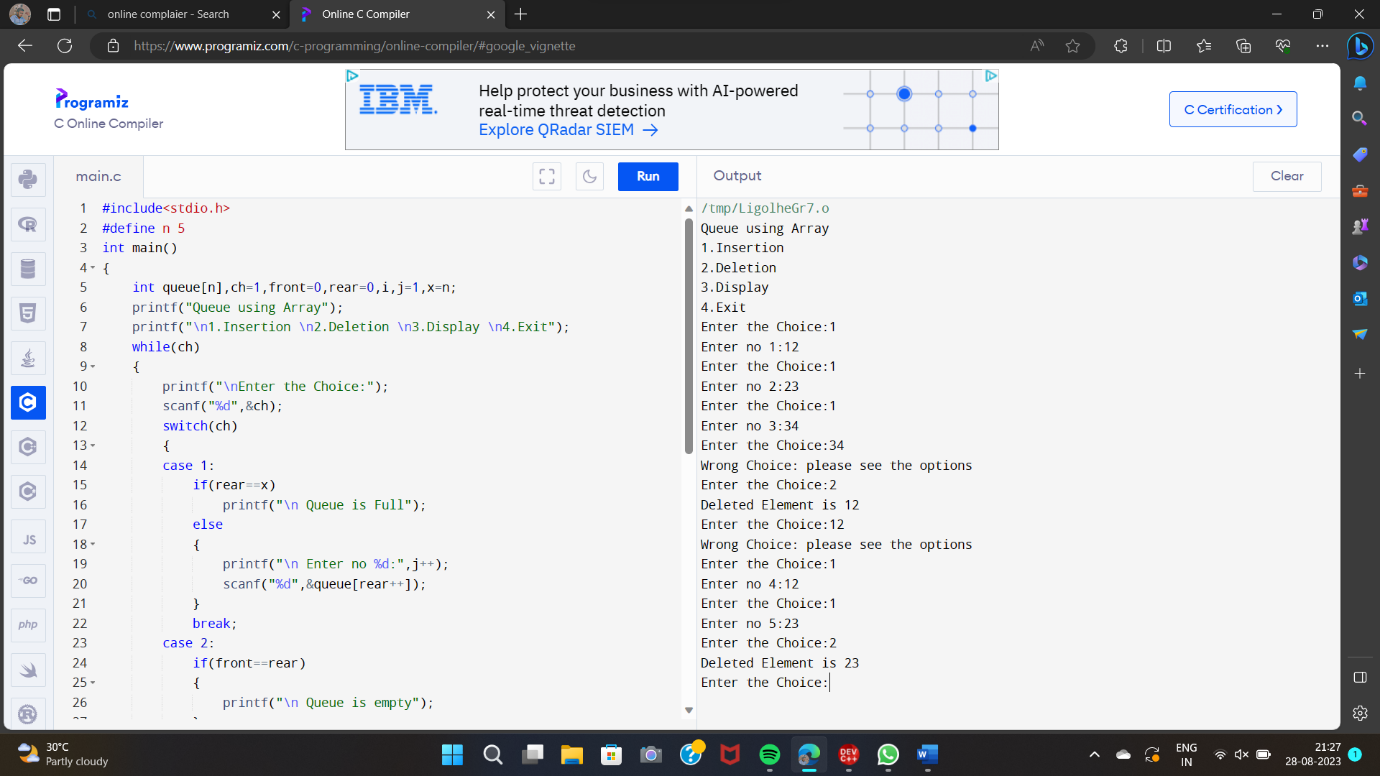
}

}

}

return 0;

}



5. #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++;

}

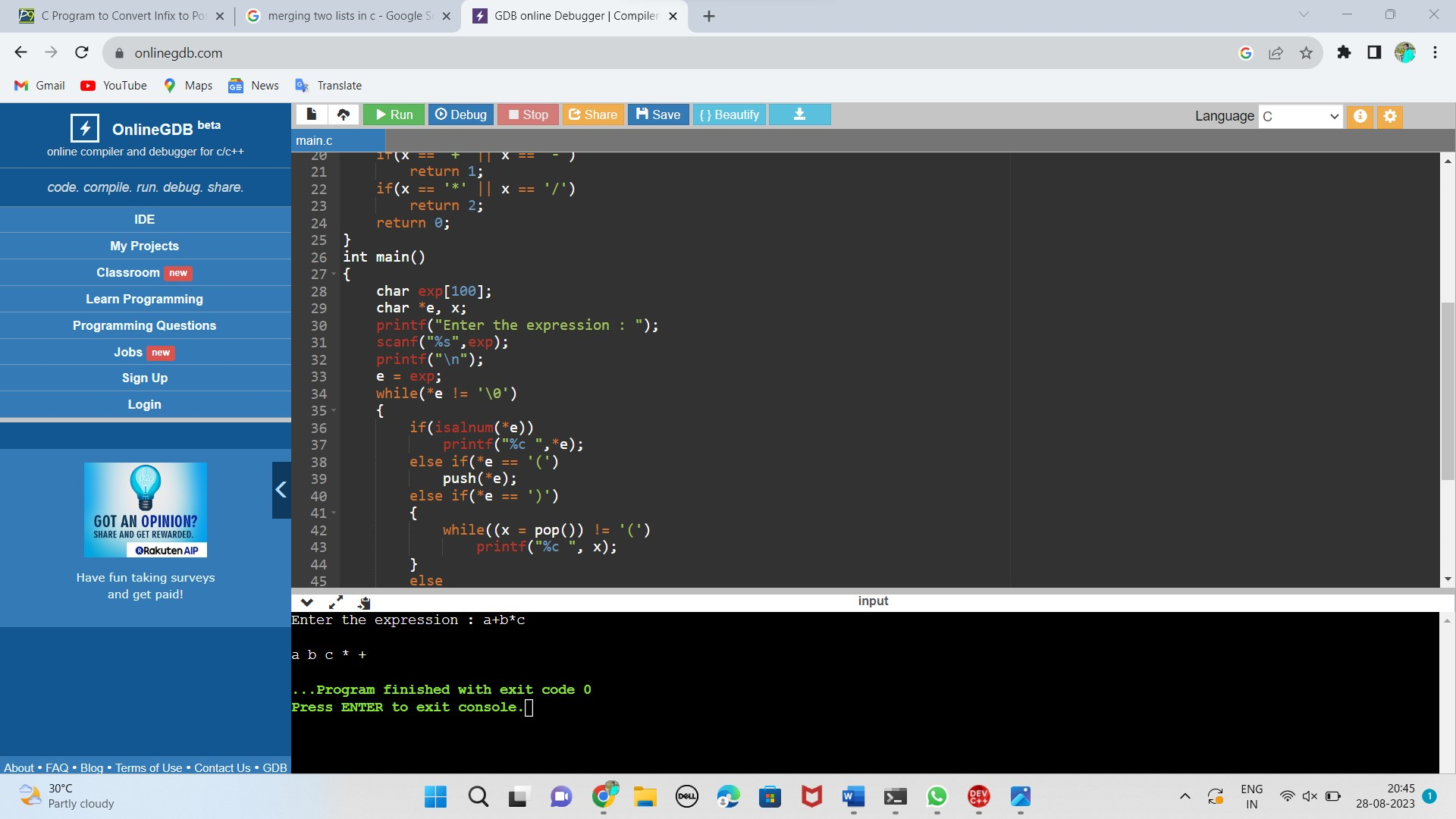
while(top != -1)

{

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

}return 0;

}



6. 6) write a C program to evaluate the postfix expression?

Code:

#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);

}

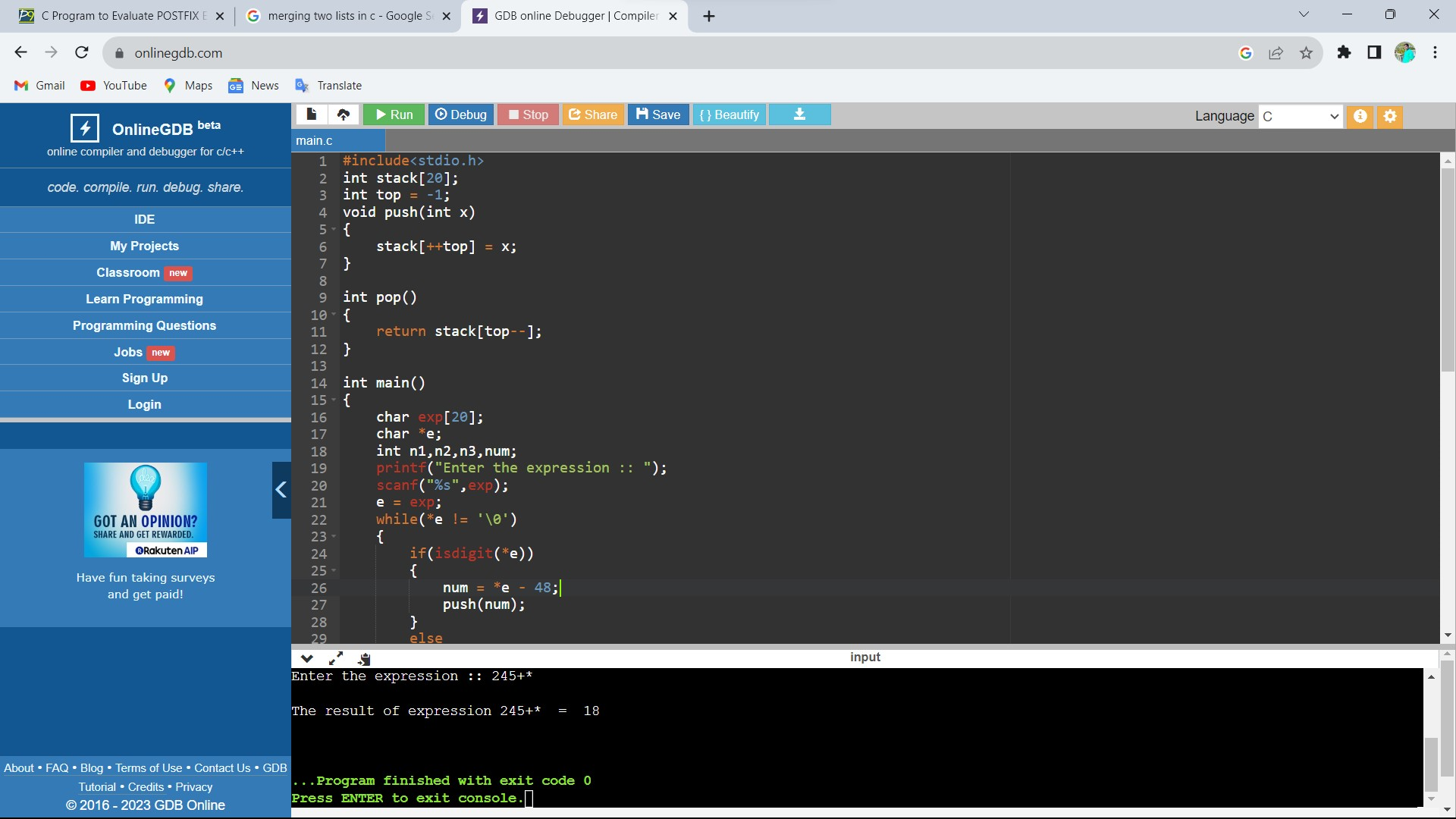
e++;

}

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

return 0;

}



7. 7)Write a C program to implement tree traversals?

Code:

#include <stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node\* left;

struct node\* right;

};

struct node\* create(int x){

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

newnode->data=x;

newnode->left=NULL;

newnode->right=NULL;

}

void inorder(struct node\* root){

if(root==NULL){

return;

}

inorder(root->left);

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

inorder(root->right);

}

void postorder(struct node\* root){

if(root==NULL){

return;

}

postorder(root->left);

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

postorder(root->right);

}

void preorder(struct node\* root){

if(root==NULL){

return;

}

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

preorder(root->left);

preorder(root->right);

}

int main(){

int a;

struct node\* root = create(1);

root->left=create(2);

root->right=create(3);

root->left->right=create(4);

root->left->left=create(5);

root->right->left=create(6);

root->right->right=create(7);

printf("enter the traversal type inorder->1 preorder->2 postorder->3:");

scanf("%d",&a);

switch(a){

case 1:inorder(root);

break;

case 2:preorder(root);

break;

case 3:postorder(root);

break;

}

return 0;

}

