Day -3

# *1.C program to implement single linked list with following operations:*

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data , node;

struct Node \*next;

};

void deleteStart (struct Node \*\*head)

{

struct Node \*temp = \*head;

if (\*head == NULL)

{

printf ("Linked List Empty, nothing to delete");

return;

}

\*head = (\*head)->next;

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

free (temp);

}

void insertStart (struct Node \*\*head, int data)

{

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

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

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

}

void display (struct Node \*node)

{

printf ("\nLinked List: ");

while (node != NULL)

{

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

node = node->next;

}

printf ("\n");

}

int main ()

{

struct Node \*head = NULL;

insertStart (&head, 24);

insertStart (&head, 78);

insertStart (&head, 92);

insertStart (&head, 40);

display (head);

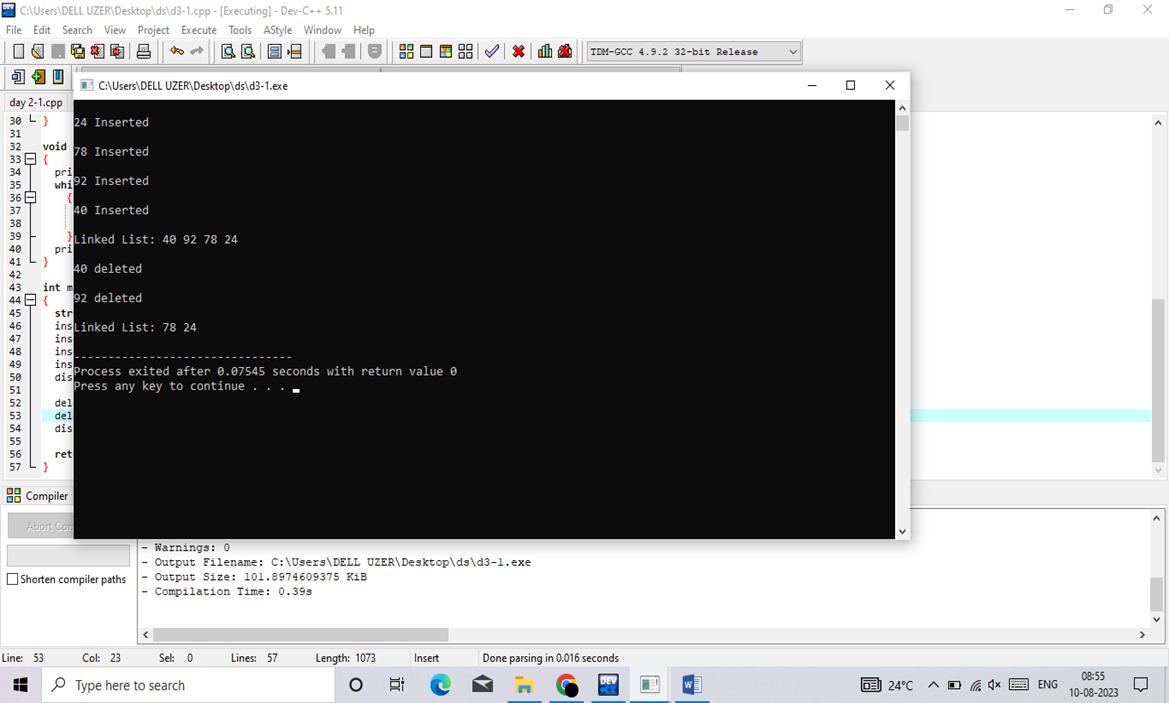
deleteStart (&head);

deleteStart (&head);

display (head);

  return 0;

}



# *2. C programme to implement stock data push element pop element display stack:*

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 5

struct stack

{

int stk[MAXSIZE];

int top;

};

typedef struct stack ST;

ST s;

void push ()

{

int num;

if (s.top == (MAXSIZE - 1))

{

printf ("Stack is Full\n");

return;

}

else

{

printf ("\nEnter element to be pushed : ");

scanf ("%d", &num);

s.top = s.top + 1;

s.stk[s.top] = num;

}

return;

}

int pop ()

{

int num;

if (s.top == - 1)

{

printf ("Stack is Empty\n");

return (s.top);

}

else

{

num = s.stk[s.top];

printf ("poped element is = %d\n", s.stk[s.top]);

s.top = s.top - 1;

}

return(num);

}

void display ()

{

int i;

if (s.top == -1)

{

printf ("Stack is empty\n");

return;

}

else

{

printf ("\nStatus of elements in stack : \n");

for (i = s.top; i >= 0; i--)

{

printf ("%d\n", s.stk[i]);

}

}

}

int main ()

{

int ch;

s.top = -1; printf ("\tSTACK OPERATIONS\n");

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

printf(" 1. PUSH\n");

printf(" 2. POP\n");

printf(" 3. DISPLAY\n");

printf(" 4. EXIT\n");

//printf("----------------------------\n");

while(1)

{

printf("\nChoose operation : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

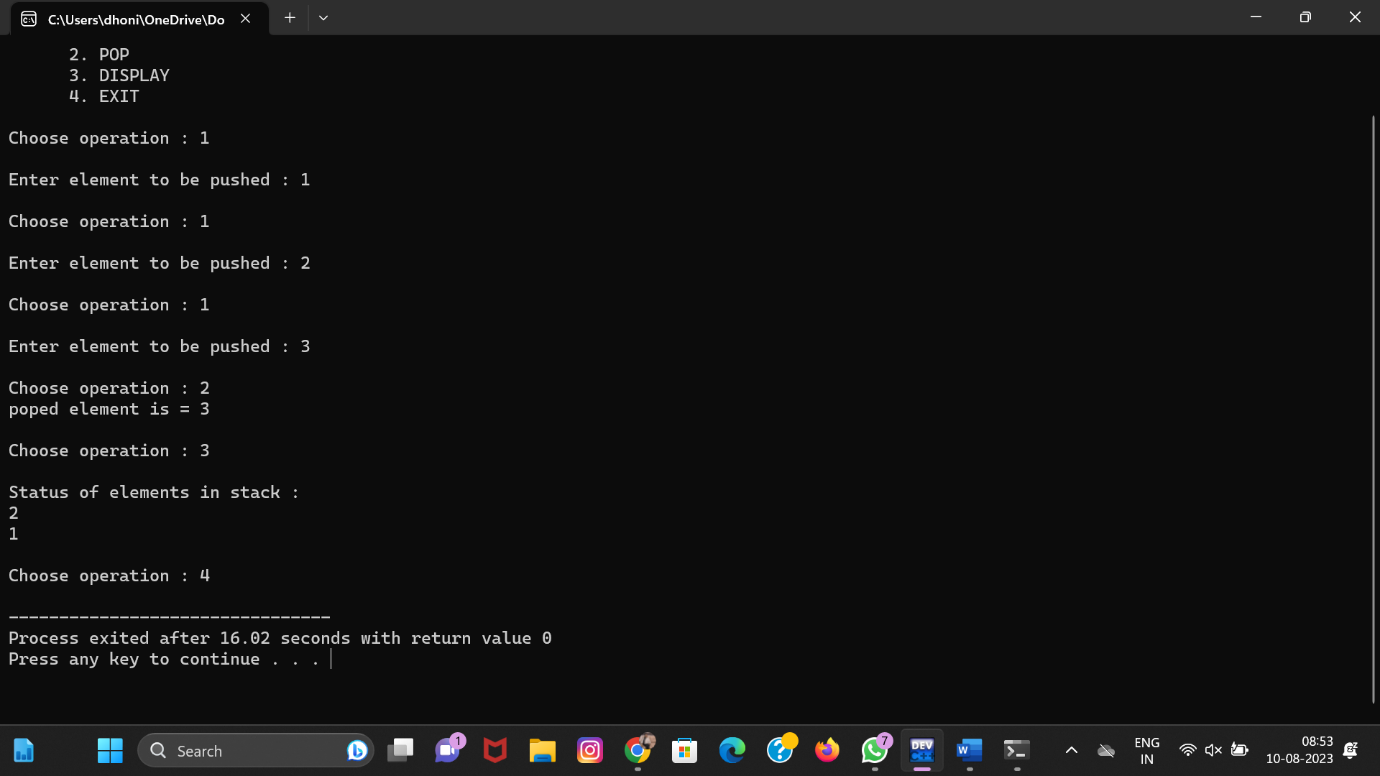
printf("Invalid operation \n");

}

}

return 0;

}



# *3.Queue data structure enqueue dequeue display:*

#include <stdio.h>

#define SIZE 5void enQueue(int);

void deQueue();

void display();int items[SIZE], front = -1, rear = -1;int main() {

deQueue();

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

enQueue(5);

enQueue(6);

display();

deQueue();

display();

return 0;

}

void enQueue(int value) {

if (rear == SIZE - 1)

printf("\nQueue is Full!!");

else {

if (front == -1)

front = 0;

rear++;

items[rear] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue() {

if (front == -1)

printf("\nQueue is Empty!!");

else {

printf("\nDeleted : %d", items[front]);

front++;

if (front > rear)

front = rear = -1;

}

}

void display() {

if (rear == -1)

printf("\nQueue is Empty!!!");

else {

int i;

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

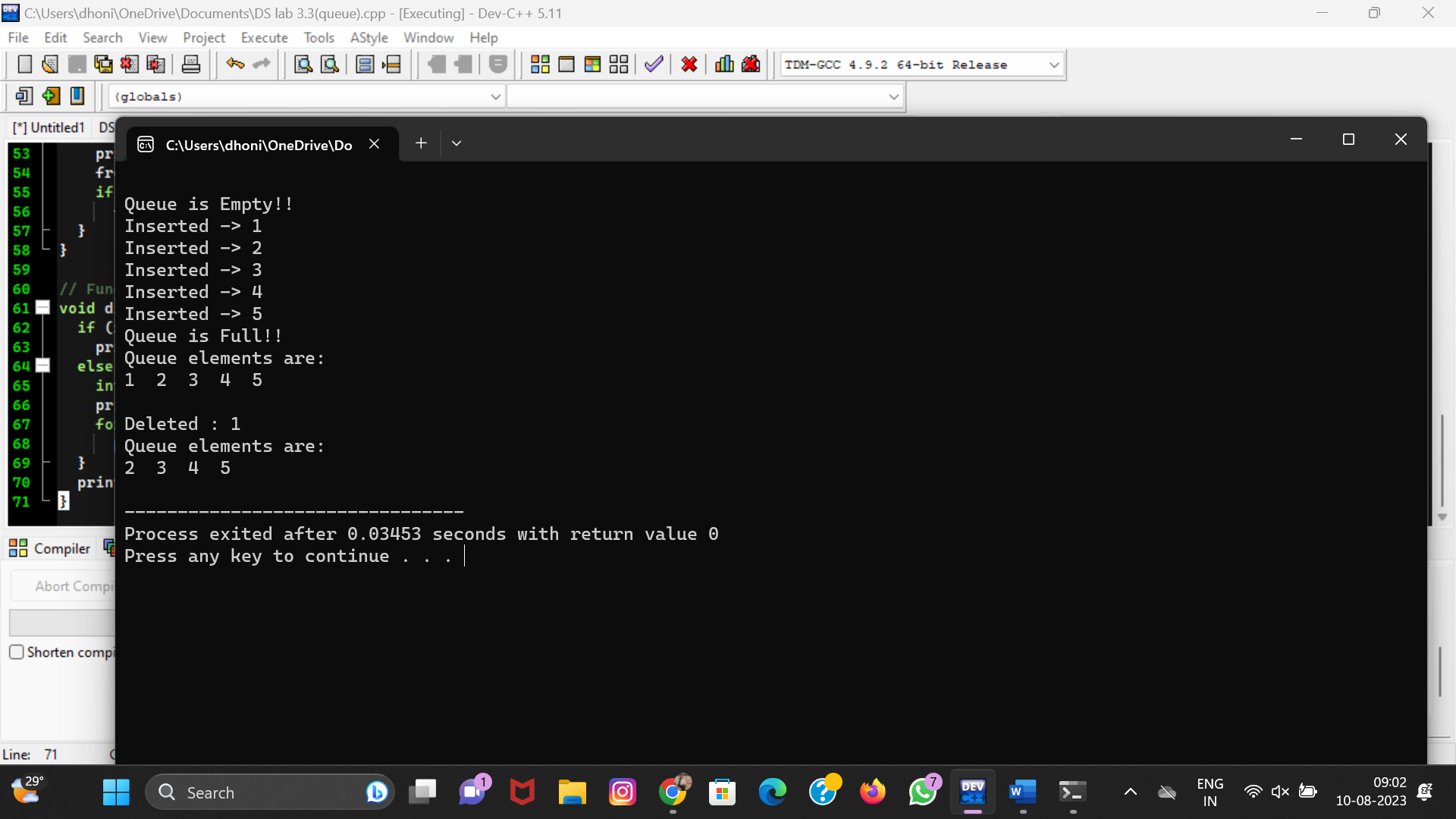
for (i = front; i <= rear; i++)

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

}

printf("\n");

}



# *4.C programme to convert infix expression into Postfix using stack:*

#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++;

}

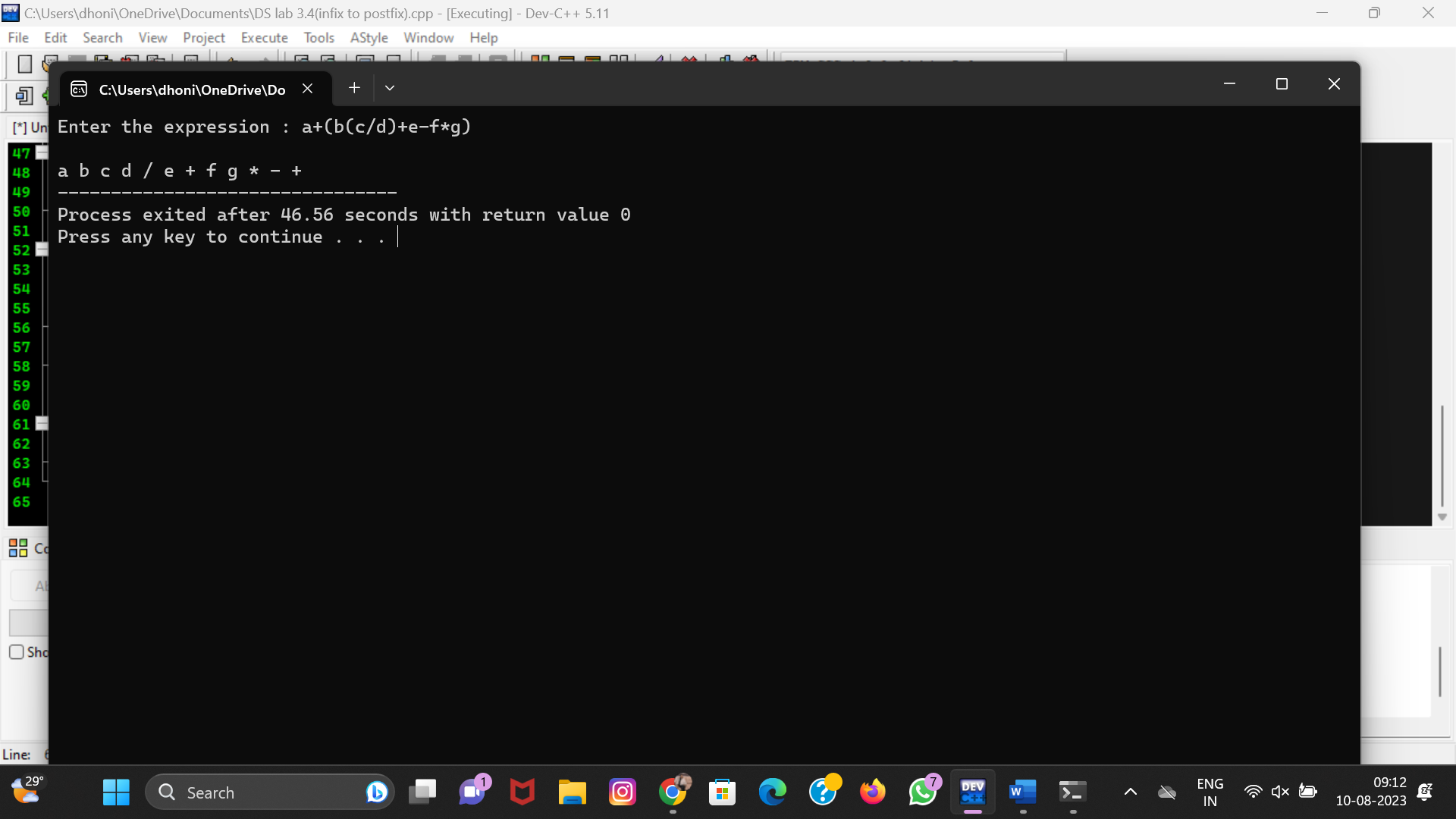
while(top != -1)

{

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

    }return 0;

}



# *5. C programme to evaluate a given expression using stack:*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 100

typedef struct {

int items[MAX\_SIZE];

int top;

}

Stack;

void push(Stack \*stack, int value) {

if (stack->top == MAX\_SIZE - 1) {

printf("Stack overflow\n");

exit(1);

}

stack->top++;

stack->items[stack->top] = value;

}

int pop(Stack \*stack) {

if (stack->top == -1) {

printf("Stack underflow\n");

exit(1);

}

int value = stack->items[stack->top];

stack->top--;

return value;

}

int evaluateExpression(char \*expression) {

Stack stack;

stack.top = -1;

for (int i = 0; i < strlen(expression); i++) {

if (expression[i] >= '0' && expression[i] <= '9') {

push(&stack, expression[i] - '0');

}

else

{

int operand2 = pop(&stack);

int operand1 = pop(&stack);

switch (expression[i]) {

case '+':

push(&stack, operand1 + operand2);

break;

case '-':

push(&stack, operand1 - operand2);

break;

case '\*':

push(&stack, operand1 \* operand2);

break;

case '/':

push(&stack, operand1 / operand2);

break;

}

}

}

return pop(&stack);

}

int main() {

char expression[MAX\_SIZE];

printf("Enter an arithmetic expression: ");

scanf("%s", expression);

int result = evaluateExpression(expression);

printf("Result: %d\n", result);

return 0;

}

