**Practical 3 – Advanced Data Structures Lab**

**1. Write a java program to demonstrate the working of Stack using array.**

**Program:**

**StackDemo.java**

import java.util.Scanner;

class MyStack{

int maxSize;

int top;

int[] stackArray;

// constructor to initialize the stack

public MyStack(int size) {

maxSize = size;

stackArray = new int[maxSize];

top = -1;

}

//push operation

public void push(int value) {

if (isFull()) {

System.out.println("Stack is full! can not push more element!");

}

else {

stackArray[++top] = value;

System.out.println(value+" is push in stack");

}

}

//pop operation

public int pop() {

if(isEmpty()) {

System.out.println("Stack is empty! Can not pop element!");

return -1;

}

else {

return stackArray[top--];

}

}

// Peek operation

public int peek() {

if (isEmpty()) {

System.out.println("Stack is empty!");

return -1;

}

else {

return stackArray[top];

}

}

// check if the stack is full

public boolean isFull() {

return (top == maxSize-1);

}

// Check if the stack is empty

public boolean isEmpty() {

return top==-1;

}

void displayAll() {

if(isEmpty()) {

System.out.println("\nStack is empty! No elements to display! \n");

}

else {

System.out.println("Elements in the stack are: ");

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

System.out.println(stackArray[i]);

}

System.out.println("");

}

}

}

public class StackDemo {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

MyStack st = new MyStack(5);

int val;

int choice;

do {

System.out.println("\n1.Push");

System.out.println("\n2.Pop");

System.out.println("\n3.Peek");

System.out.println("\n4.Display");

System.out.println("\n5.Exit");

System.out.println("Enter your Choise:");

choice = sc.nextInt();

switch (choice) {

case 1:

System.out.println("\nEnter a value to be pushed :");

val = sc.nextInt();

st.push(val);

break;

case 2:

val = st.pop();

System.out.println("\nPopped element is : "+val);

break;

case 3:

System.out.println("\nPeeked element is "+st.peek());

break;

case 4:

st.displayAll();

break;

case 5:

System.out.println("Exiting the program.");

break;

default:

System.out.println("\nWrong Choice!\n");

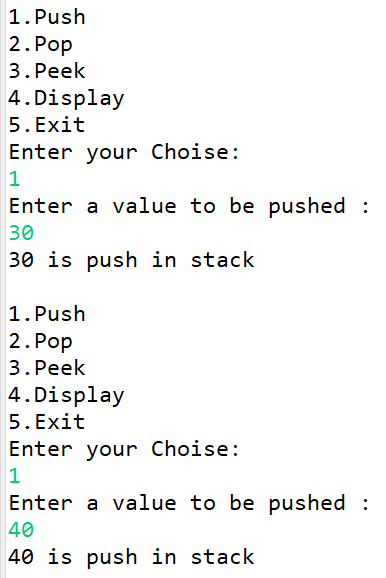
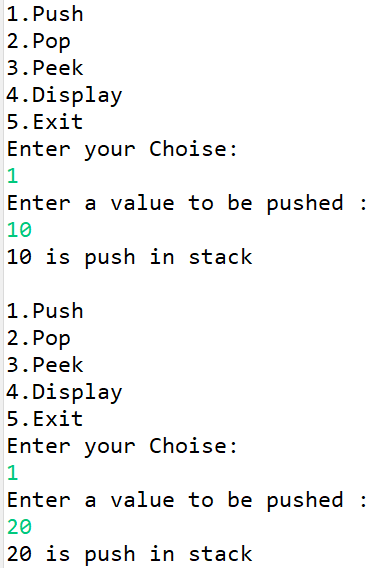
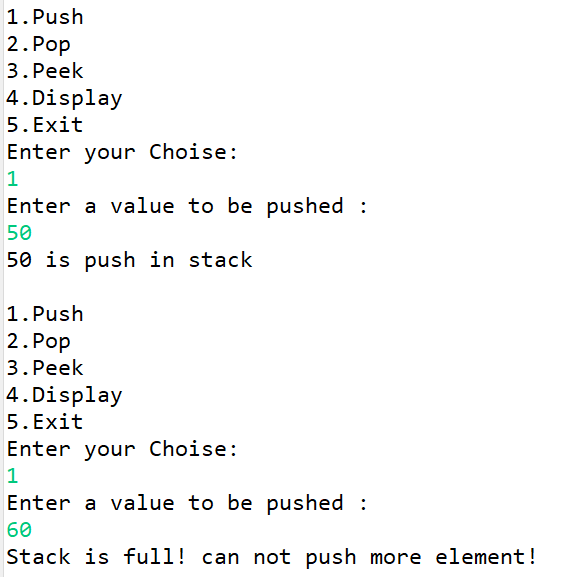
}

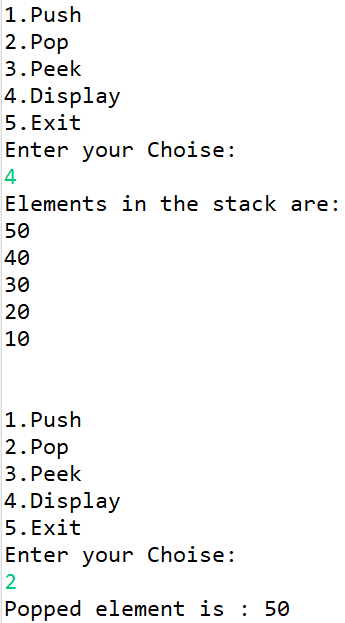
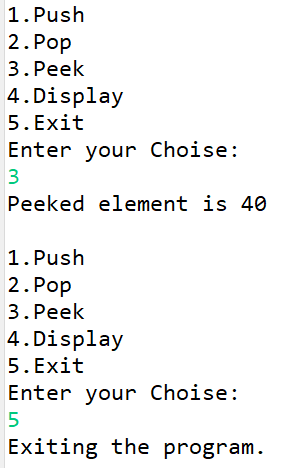
}while(choice!=5);

}

}

**Output:**

**2. Write a java program to demonstrate the working of Queue using array.**

**Program:**

import java.util.Scanner;

class MyQueue{

int queueSize;

int arr[];

int q\_front;

int q\_rear;

public MyQueue(int size) {

queueSize = size;

arr = new int[queueSize];

q\_front = 0;

q\_rear = -1;

}

public void enqueue(int value) {

if(isFull()) {

System.out.println("Queue is full! Cannot enqueue.");

}

else {

q\_rear = q\_rear + 1;

arr[q\_rear] = value;

}

}

public int dequeue() {

if(isEmpty()) {

System.out.println("Queue is empty! Cannot dequeue.");

return -1;

}

else {

int q\_element = arr[q\_front];

q\_front = q\_front + 1;

return q\_element;

}

}

public boolean isEmpty() {

return (q\_rear == -1 || q\_front>q\_rear);

}

public boolean isFull() {

return (q\_rear == queueSize - 1);

}

public int size() {

return q\_rear - q\_front + 1;

}

public void displayAllElement() {

if(isEmpty()) {

System.out.println("\nQueue is empty! No element in the queue.");

}

else {

System.out.println("Elements in the queue are: ");

for(int i = q\_front; i<=q\_rear;i++) {

System.out.println(arr[i]+" ");

}

System.out.println(" ");

}

}

}

public class QueueDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of the queue: ");

int size = sc.nextInt();

MyQueue queue = new MyQueue(size);

int choice, val;

do {

System.out.println("\n1. Enqueue");

System.out.println("2. Dequeue");

System.out.println("3. Check if Queue is Full");

System.out.println("4. Check if queue is Empty");

System.out.println("5. Display All Elements");

System.out.println("6. Display Queue Size");

System.out.println("7. Exit");

System.out.println("\nEnter your choice: ");

choice = sc.nextInt();

switch(choice) {

case 1:

if(!queue.isFull()) {

System.out.println("Enter value to be enqueued: ");

val = sc.nextInt();

queue.enqueue(val);

}

else {

System.out.println("Queue is full! cannot enqueue.");

}

break;

case 2:

val = queue.dequeue();

if(val!=-1) {

System.out.println("Dequeued element is: "+val);

}

break;

case 3:

if(queue.isFull()) {

System.out.println("Queue is full.");

}

else {

System.out.println("Queue is not full.");

}

break;

case 4:

if(queue.isEmpty()) {

System.out.println("Queue is empty.");

}

else {

System.out.println("Queue is not empty.");

}

break;

case 5:

queue.displayAllElement();

break;

case 6:

System.out.println("Current size of the queue: "+queue.size());

break;

case 7:

System.out.println("Exiting the program.");

break;

default:

System.out.println("Wrong choice! Please try again.");

}

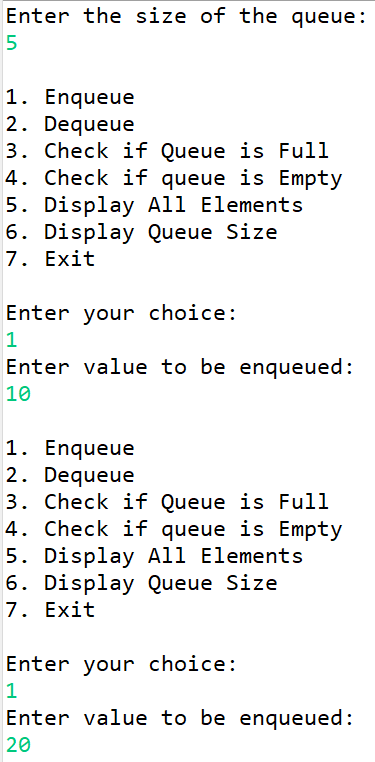
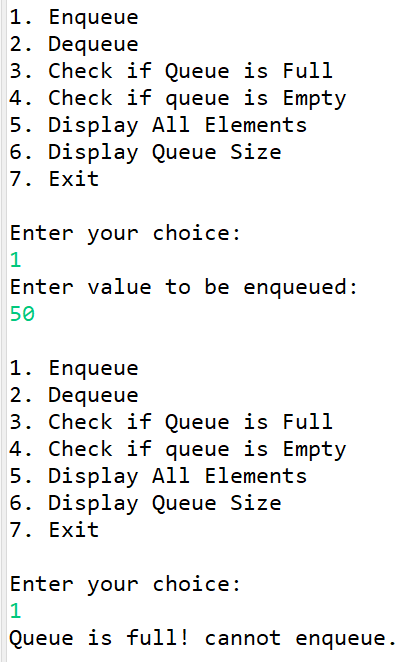
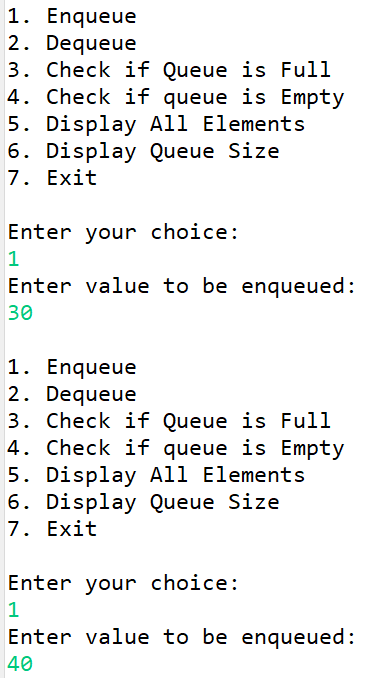
}while(choice!=7);

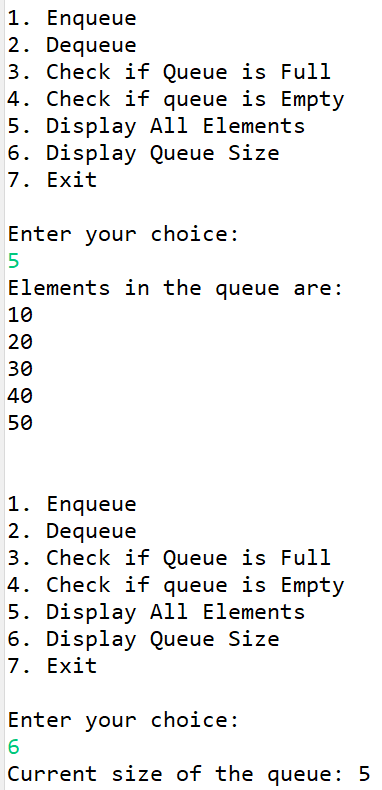
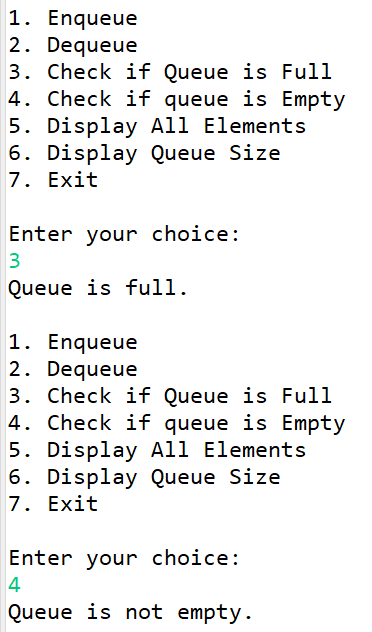
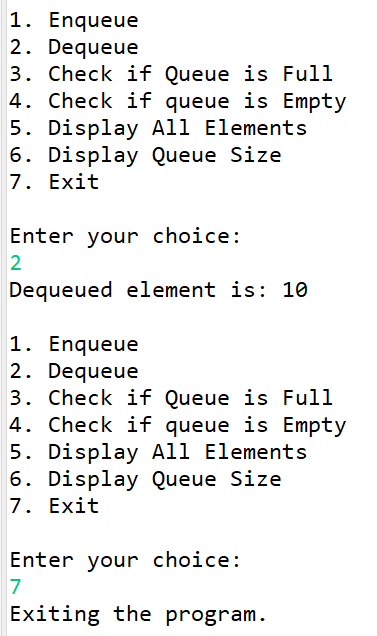
sc.close();

}

}

**Output:**

**3. Write a java program to demonstrate the working of Circular Queue using array.**

**Program:**

**CircularQueueDemo.java**

import java.util.Scanner;

class CircularQueue{

int QUEUESIZE;

int arr[];

int q\_front;

int q\_rear;

public CircularQueue(int size) {

QUEUESIZE = size;

arr = new int[QUEUESIZE];

q\_front = -1;

q\_rear = -1;

}

public void enqueue(int val) {

// check for empty queue for adding first element

if(isEmpty()) {

q\_rear=0;

q\_front=0;

}

else {

q\_rear = (q\_rear+1)%QUEUESIZE;

}

arr[q\_rear] = val;

}

public int dequeue() {

int q\_element;

if(q\_front==q\_rear) {

// the only element in the queue

q\_element = arr[q\_front];

q\_rear = -1; // resetting rear for empty queue

q\_front = -1; // reseting front for empty queue

}

else {

// more than one element existing

q\_element=arr[q\_front];

q\_front = (q\_front+1)%QUEUESIZE;

}

return q\_element;

}

public boolean isEmpty() {

if(q\_rear==-1) {

return true;

}

else {

return false;

}

}

public boolean isFull() {

if((q\_rear+1)%QUEUESIZE==q\_front) {

return true;

}

else {

return false;

}

}

public int size() {

if(q\_rear>q\_front) {

return q\_rear-q\_front+1;

}

else {

return QUEUESIZE-q\_rear-q\_front+1;

}

}

public void displayAllElements() {

if(q\_rear == -1) {

System.out.println("No elements to display!");

return;

}

System.out.println("Elements in the queue are: ");

for(int i=q\_front; i!=q\_rear;i=(i+1)%QUEUESIZE) {

System.out.println(arr[i]+ " ");

}

System.out.println(arr[q\_rear]+" ");

System.out.println("");

}

}

public class CircularQueueDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of the queue: ");

int size = sc.nextInt();

CircularQueue queue = new CircularQueue(size);

int choice, val;

do {

System.out.println("\n1. Enqueue");

System.out.println("2. Dequeue");

System.out.println("3. Check if Queue is Full");

System.out.println("4. Check if queue is Empty");

System.out.println("5. Display All Elements");

System.out.println("6. Display Queue Size");

System.out.println("7. Exit");

System.out.println("Enter your choice: ");

choice = sc.nextInt();

switch(choice) {

case 1:

if(!queue.isFull()) {

System.out.println("Enter value to be enqueued: ");

val = sc.nextInt();

queue.enqueue(val);

}

else {

System.out.println("Queue is full! cannot enqueue.");

}

break;

case 2:

val = queue.dequeue();

if(val!=-1) {

System.out.println("Dequeued element is: "+val);

}

break;

case 3:

if(queue.isFull()) {

System.out.println("Queue is full.");

}

else {

System.out.println("Queue is not full.");

}

break;

case 4:

if(queue.isEmpty()) {

System.out.println("Queue is empty.");

}

else {

System.out.println("Queue is not empty.");

}

break;

case 5:

queue.displayAllElements();

break;

case 6:

System.out.println("Current size of the queue: "+queue.size());

break;

case 7:

System.out.println("Exiting the program.");

break;

default:

System.out.println("Wrong choice! Please try again.");

}

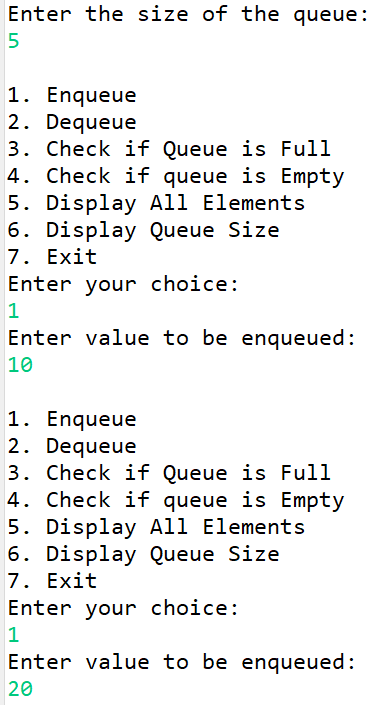
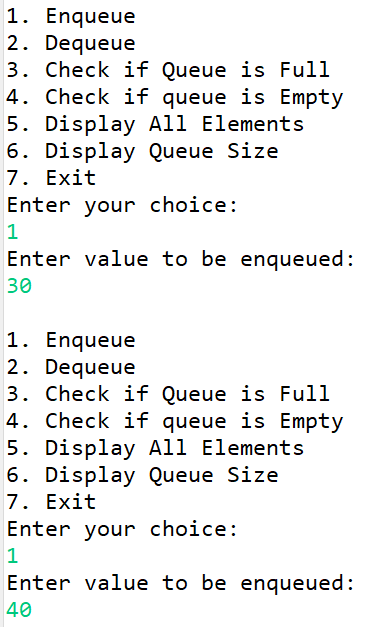
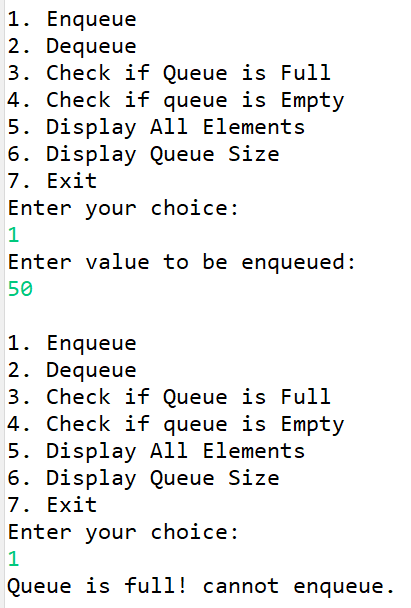
}while(choice!=7);

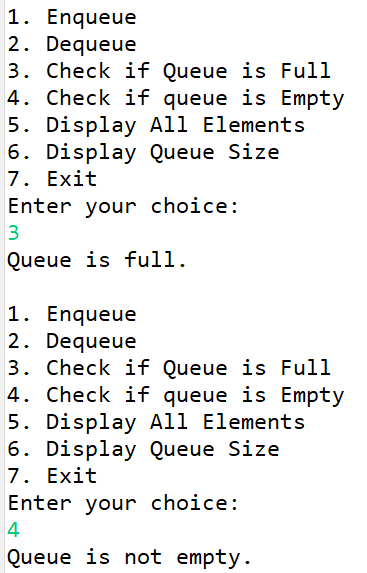
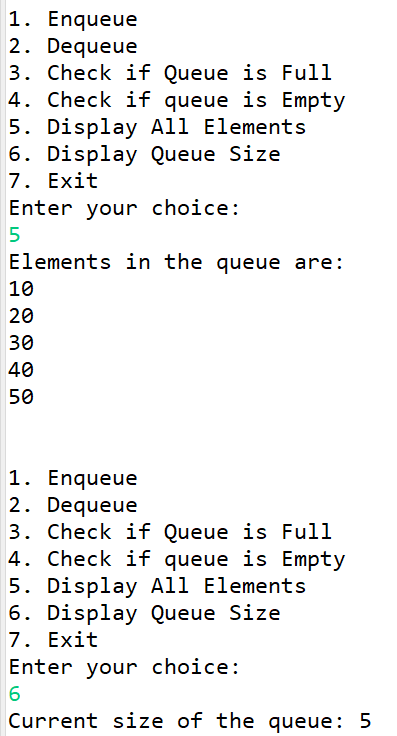
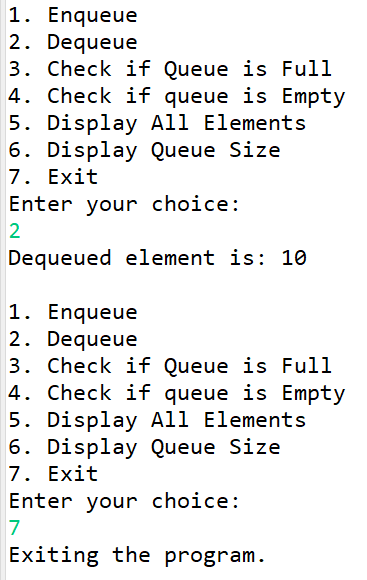
sc.close();

}

}

**Output:**

**4. Write a java program to demonstrate the application of stack for evaluating postfix expression.**

**Program:**

package mypack;

import java.util.Scanner;

class MyStack{

int maxSize;

int top;

int[]stackArray;

//constructor to initialize the stack

public MyStack(int size)

{

maxSize = size;

stackArray = new int[maxSize];

top = -1;

}

//push operation

public void push(int value) {

if(isFull()) {

System.out.println("stack is full!");

}

else {

stackArray[++top] = value;

}

}

//pop operation

public int pop() {

if(isEmpty()) {

System.out.println("stack is empty can not pop element!");

return -1;

}

else {

return stackArray[top--];

}

}

//peek operation

public int peek() {

if(isEmpty()) {

System.out.println("stack is empty");

return -1;

}

else {

return stackArray[top];

}

}

//check if the stack is full

public boolean isFull() {

return (top == maxSize-1);

}

//check if stack is empty

public boolean isEmpty() {

return top == -1;

}

//display

void displayAll() {

if(isEmpty()) {

System.out.println("\n stack is empty! no elements to display!");

}

else {

System.out.println("Elements in stack are:");

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

System.out.println(stackArray);

}

System.out.println(" ");

}

}

}

public class PostfixEval{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

MyStack st=new MyStack(5);

System.out.println("please enter postfix expression : ");

String exp= sc.nextLine();

for(int i=0; i<exp.length();i++) {

char c = exp.charAt(i);

if(Character.isDigit(c))

st.push(c-'0');

else {

int val1=st.pop();

int val2 = st.pop();

System.out.println(val2+" "+c+" "+val1);

switch(c) {

case'+':

st.push(val2+val1);

break;

case'-':

st.push(val2-val1);

break;

case'/':

st.push(val2/val1);

break;

case'\*':

st.push(val2\*val1);

break;

case'^':

int power=(int)Math.pow(val2, val1);

st.push(power);

break;

default:

System.out.println("Invalid operator found!");

}

}

}

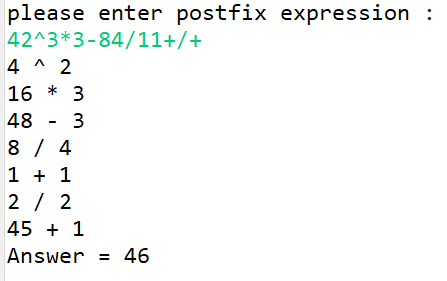
System.out.println("Answer = "+st.pop());

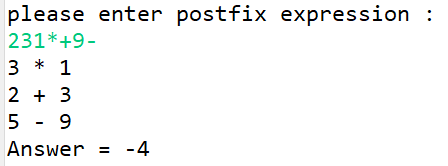
sc.close();

}

}

**Output:**





**5. Write a java program to demonstrate the application of stack for parenthesis balancing.**

**Program:**

import java.util.Scanner;

class MyStack4{

int maxSize;

int top;

char[] stackArray;

// constructor to initialize the stack

public MyStack4(int size) {

maxSize = size;

stackArray = new char[maxSize];

top = -1;

}

//push operation

public void push(char value) {

if (isFull()) {

System.out.println("Stack is full! can not push more element!");

}

else {

stackArray[++top] = value;

System.out.println(value+" is push in stack");

}

}

//pop operation

public char pop() {

if(isEmpty()) {

System.out.println("Stack is empty! Can not pop element!");

return 'F';

}

else {

return stackArray[top--];

}

}

// Peek operation

public char peek() {

if (isEmpty()) {

System.out.println("Stack is empty!");

return 'F';

}

else {

return stackArray[top];

}

}

// check if the stack is full

public boolean isFull() {

return (top == maxSize-1);

}

// Check if the stack is empty

public boolean isEmpty() {

return top==-1;

}

}

public class ParenBal {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

MyStack4 st = new MyStack4(10);

System.out.println("Please enter postfix expression : ");

String exp = sc.nextLine();

int isValid=1;

char c;

//Scan all characters one by one

for(int i=0;i<exp.length();i++) {

c = exp.charAt(i);

char stChar;

if(c=='{' || c=='[' || c=='(') {

st.push(c);

}

else {

if(st.isEmpty()) {

isValid = 0;

break;

}

else {

stChar = st.pop();

if((stChar != '(' && c==')') || (stChar != '{' && c=='}') || (stChar != '[' && c==']')) {

isValid = 0;

break;

}

}

}

}

if(isValid==1 && st.isEmpty()) {

System.out.println("\nParenthesis are balanced!");

}

else {

System.out.println("\nParentheses are not balanced!");

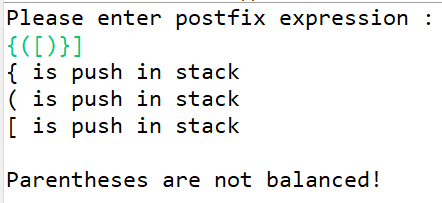
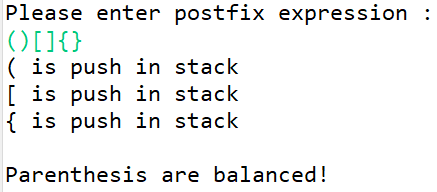
}

sc.close();

}

}

**Output:**



**6. Write a java program to demonstrate the application of stack for conversion of Infix into postfix expression.**

**Program:**

import java.util.Scanner;

class MyStack1{

int maxSize;

int top;

char[] stackArray;

// constructor to initialize the stack

public MyStack1(int size) {

maxSize = size;

stackArray = new char[maxSize];

top = -1;

}

//push operation

public void push(char value) {

if (isFull()) {

System.out.println("Stack is full! can not push more element!");

}

else {

stackArray[++top] = value;

System.out.println(value+" is push in stack");

}

}

//pop operation

public char pop() {

if(isEmpty()) {

System.out.println("Stack is empty! Can not pop element!");

return 'F';

}

else {

return stackArray[top--];

}

}

// Peek operation

public char peek() {

if (isEmpty()) {

System.out.println("Stack is empty!");

return 'F';

}

else {

return stackArray[top];

}

}

// check if the stack is full

public boolean isFull() {

return (top == maxSize-1);

}

// Check if the stack is empty

public boolean isEmpty() {

return top==-1;

}

}

public class InfixToPostfixDemo {

// function to return precedence of operators

int prec(char c) {

if(c=='^')

return 3;

else if (c=='/' || c=='\*')

return 2;

else if(c=='+'|| c=='-')

return 1;

else

return -1;

}

// function to perform infix to postfix conversion

void infixToPostfix(String s, MyStack1 st) {

StringBuilder result = new StringBuilder();

for (int i=0;i<s.length();i++) {

char c = s.charAt(i);

//if the scanned character is an operand, add it to the output string.

if((c>='a' && c<='z') || (c>='A' && c<='Z') || (c >= '0' && c<='9'))

result.append(c);

// if the scanned character is an '(', push it to the stack.

else if (c=='(')

st.push('(');

else if (c==')')

{

while (st.peek()!='(') {

result.append(st.pop());

}

st.pop();

}

// if an operator is scanned

else {

while(!st.isEmpty() && (prec(c)<prec(st.peek()) || prec(c) == prec(st.peek()))) {

result.append(st.pop());

}

st.push(c);

}

}

// Pop all the remaining elements from the stack

while(!st.isEmpty()) {

result.append(st.pop());

}

System.out.println(result.toString());

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

MyStack1 st = new MyStack1(15);

System.out.println("Please enter infix expression: ");

String exp = sc.nextLine();

InfixToPostfixDemo ip = new InfixToPostfixDemo();

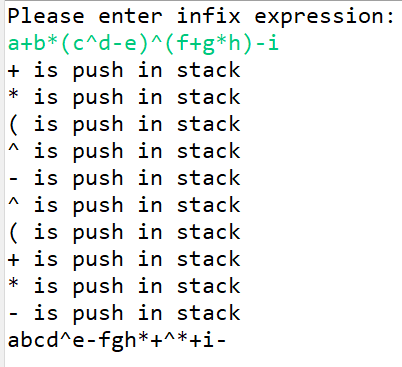
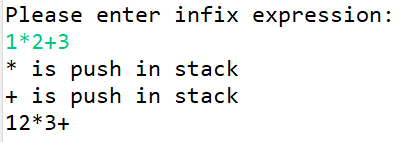
ip.infixToPostfix(exp, st);

sc.close();

}

}

**Output:**

**Conclusion : Implementation of linear data structures successfully.**