Assignments

Assignment-1
Hospital Management

```
class Patient {
    String name;
    Patient next;
    public Patient(String name) {
        this.name = name;
        this.next = null;
    }
}
class Main {
    private Patient head;
    public void addPatient(String name) {
        Patient newPatient = new Patient(name);
        if (head == null) {
            head = newPatient;
            return;
        }
        Patient temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newPatient;
    }
    public void addEmergencyPatient(String name) {
        Patient newPatient = new Patient(name);
        newPatient.next = head;
        head = newPatient;
    }
    public void dischargePatient(String name) {
        if (head == null) {
            System.out.println("Queue is empty. No patient to discharge.");
            return;
        }
        if (head.name.equals(name)) {
            System.out.println("Discharged: " + head.name);
            head = head.next;
            return;
        }
        Patient current = head;
        Patient prev = null;
        while (current != null && !current.name.equals(name)) {
            prev = current;
            current = current.next;
        }
        if (current == null) {
            System.out.println("Patient named \"" + name + "\" not found in the queue.");
            return;
        }
        prev.next = current.next;
```

```
        System.out.println("Discharged: " + current.name);
    }
    public void displayQueue() {
        if (head == null) {
            System.out.println("Queue is empty.");
            return;
        }
        Patient temp = head;
        System.out.println("Current Patient Queue:");
        while (temp != null) {
            System.out.print(temp.name + " -> ");
            temp = temp.next;
        }
        System.out.println("NULL");
    }
    public static void main(String[] args) {
        Main queue = new Main();
        queue.addPatient("John");
        queue.addPatient("Emma");
        queue.addEmergencyPatient("Drake");
        queue.addPatient("Sophia");
        queue.addEmergencyPatient("Mia");
        queue.displayQueue();
        System.out.println("\nDischarging patient: Emma");
        queue.dischargePatient("Emma");
        System.out.println("\nUpdated Queue:");
        queue.displayQueue();
    }
}
```

Output

```
Current Patient Queue:
Mia -> Drake -> John -> Emma -> Sophia -> NULL

Discharging patient: Emma
Discharged: Emma

Updated Queue:
Current Patient Queue:
Mia -> Drake -> John -> Sophia -> NULL


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment-2
Browser History

```
import java.util.Stack;
public class Main {
    private Stack<String> backStack;
    private Stack<String> forwardStack;
    private String currentPage;
    public Main() {
        backStack = new Stack<>();
        forwardStack = new Stack<>();
        currentPage = "Home";
    }
    public void visit(String url) {
        if (currentPage != null) {
            backStack.push(currentPage);
        }
        currentPage = url;
        forwardStack.clear();
        System.out.println("Visited: " + currentPage);
    }
    public void back() {
        if (backStack.isEmpty()) {
            System.out.println("No pages to go back to.");
            return;
        }
        forwardStack.push(currentPage);
        currentPage = backStack.pop();
        System.out.println("Went back to: " + currentPage);
    }
    public void forward() {
        if (forwardStack.isEmpty()) {
            System.out.println("No pages to go forward to.");
            return;
        }
        backStack.push(currentPage);
        currentPage = forwardStack.pop();
        System.out.println("Went forward to: " + currentPage);
    }
    public void current() {
        System.out.println("Current page: " + currentPage);
    }
    public static void main(String[] args) {
        Main b = new Main();
        b.current();
        b.visit("web1.com");
        b.visit("web2.com");
        b.visit("web3.com");
        b.back();
        b.back();
        b.back();
        b.forward();
```

```
        b.visit("web4.com");
        b.back();
        b.current();
    }
}
```

Main.java

```
    public void back() {
        if (backStack.isEmpty()) {
            System.out.println("No pages to go back to.");
            return;
        }
        forwardStack.push(currentPage);
        currentPage = backStack.pop();
        System.out.println("Went back to: " + currentPage);
    }
```

input

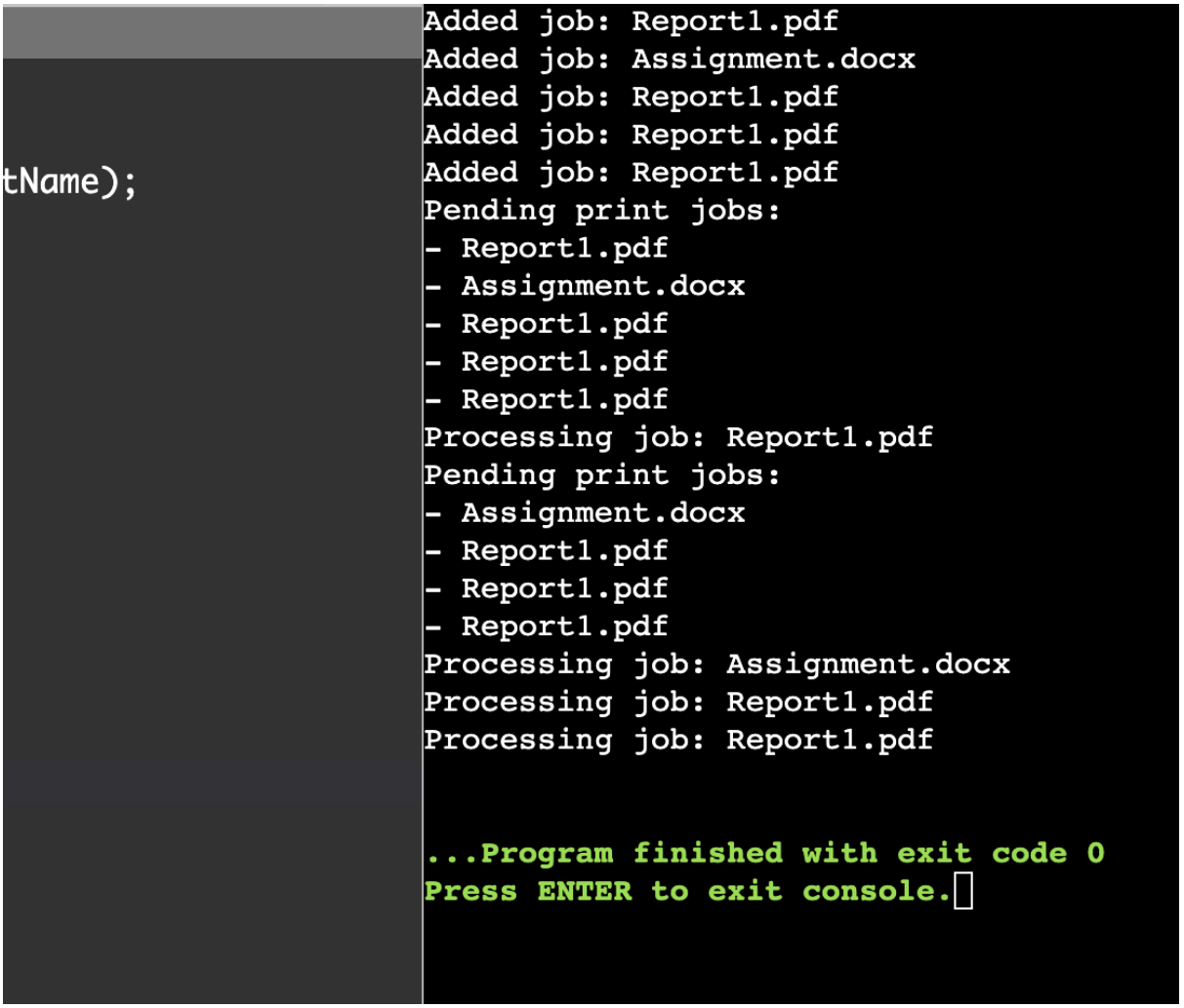
```
Current page: Home
Visited: web1.com
Visited: web2.com
Visited: web3.com
Went back to: web2.com
Went back to: web1.com
Went back to: Home
Went forward to: web1.com
Visited: web4.com
Went back to: web1.com
Current page: web1.com
```

Assignment-3

PrintQueue

```
class PrintJob {
    String documentName;
    PrintJob next;

    public PrintJob(String documentName) {
        this.documentName = documentName;
        this.next = null;
    }
}

class Main {
    private PrintJob front;
    private PrintJob rear;
    public void addJob(String documentName) {
        PrintJob newJob = new PrintJob(documentName);
        if (rear == null) {
            front = rear = newJob;
        } else {
            rear.next = newJob;
            rear = newJob;
        }
        System.out.println("Added job: " + documentName);
    }
    public void processJob() {
```

```
        if (front == null) {
            System.out.println("No print jobs to process.");
            return;
        }

        System.out.println("Processing job: " + front.documentName);
        front = front.next;

        if (front == null) {
            rear = null;
        }
    }
    public void viewPendingJobs() {
        if (front == null) {
            System.out.println("No pending print jobs.");
            return;
        }

        System.out.println("Pending print jobs:");
        PrintJob temp = front;
        while (temp != null) {
            System.out.println("- " + temp.documentName);
            temp = temp.next;
        }
    }

    public static void main(String[] args) {
        Main queue = new Main();

        queue.addJob("Report1.pdf");
        queue.addJob("Assignment.docx");
        queue.addJob("Report1.pdf");
        queue.addJob("Report1.pdf");
        queue.addJob("Report1.pdf");

        queue.viewPendingJobs();

        queue.processJob();

        queue.viewPendingJobs();

        queue.processJob();
        queue.processJob();
        queue.processJob();
    }
}
```



Assignment-4
Undo-Redo

```
import java.util.Stack;

public class Main {
    private Stack<String> us;
    private Stack<String> rd;

     Main() {
        us = new Stack<>();
        rd = new Stack<>();
    }

    public void performAction(String action) {
        us.push(action);
        rd.clear();
        System.out.println("Action: " + action);
    }

    public void undo() {
        if (us.isEmpty()) {
            System.out.println("Nothing to undo.");
```

```
            return;
        }

        String lastAction = us.pop();
        rd.push(lastAction);
        System.out.println("Undo: " + lastAction);
    }

    public void redo() {
        if (rd.isEmpty()) {
            System.out.println("Nothing to redo.");
            return;
        }

        String action = rd.pop();
        us.push(action);
        System.out.println("Redo: " + action);
    }

    public void currentState() {
        if (us.isEmpty()) {
            System.out.println("Document is empty.");
        } else {
            System.out.println("Current State: " + us.peek());
        }
    }

    public static void main(String[] args) {
        Main editor = new Main();

        editor.performAction("Report Preparation");
        editor.performAction("Assignment");
        editor.performAction("Neopat");

        editor.undo();
        editor.undo();
        editor.redo();

        editor.currentState();
    }
}
```

Main.java

```
47
48      public static void main(String[] args) {
49          Main editor = new Main();
50
51          editor.performAction("Report Preparation");
52          editor.performAction("Assignment");
53          editor.performAction("Neopat");
54
55          editor.undo();
56          editor.undo();
57          editor.redo();
```

input

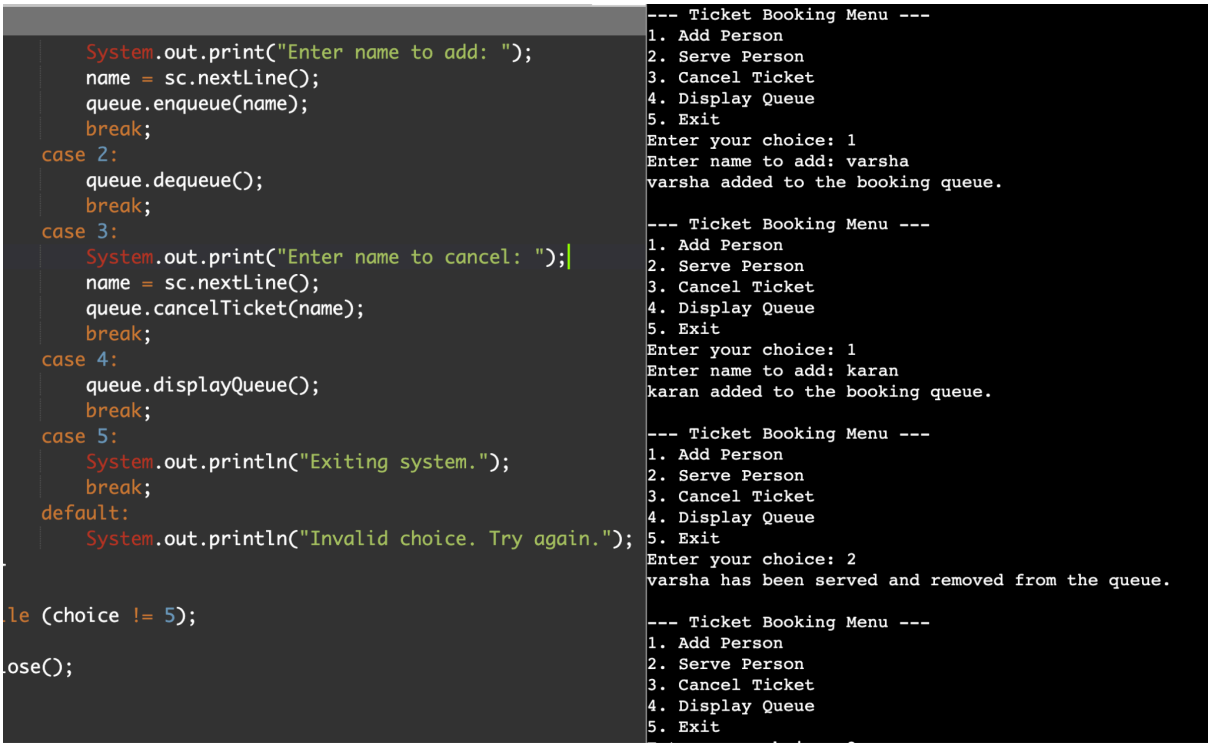
```
Action: Report Preparation
Action: Assignment
Action: Neopat
Undo: Neopat
Undo: Assignment
Redo: Assignment
Current State: Assignment
```

Assignment-5
Ticket Booking System

```
import java.util.Scanner;

class Node {
    String name;
    Node next;

    public Node(String name) {
        this.name = name;
        this.next = null;
    }
}

class Main {
    private Node front, rear;

    public Main() {
        front = rear = null;
    }

    public void enqueue(String name) {
        Node newNode = new Node(name);
        if (rear == null) {
            front = rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        System.out.println(name + " added to the booking queue.");
    }

    public void dequeue() {
```

```
    if (front == null) {
        System.out.println("Queue is empty. No person to serve.");
        return;
    }

    System.out.println(front.name + " has been served and removed from the queue.");
    front = front.next;
    if (front == null) rear = null;
}

public void cancelTicket(String name) {
    if (front == null) {
        System.out.println("Queue is empty.");
        return;
    }

    if (front.name.equals(name)) {
        dequeue();
        return;
    }

    Node prev = null, curr = front;
    while (curr != null && !curr.name.equals(name)) {
        prev = curr;
        curr = curr.next;
    }

    if (curr == null) {
        System.out.println(name + " not found in queue.");
        return;
    }

    prev.next = curr.next;
    if (curr == rear) rear = prev;

    System.out.println(name + "'s ticket has been cancelled.");
}

public void displayQueue() {
    if (front == null) {
        System.out.println("Queue is empty.");
        return;
    }

    System.out.print("Current Queue: ");
    Node temp = front;
    while (temp != null) {
        System.out.print(temp.name + " ");
        temp = temp.next;
    }
    System.out.println();
}
```

```
    public static void main(String[] args) {
        Main queue = new Main();
        Scanner sc = new Scanner(System.in);
        int choice;
        String name;

        do {
            System.out.println("\n--- Ticket Booking Menu ---");
            System.out.println("1. Add Person");
            System.out.println("2. Serve Person");
            System.out.println("3. Cancel Ticket");
            System.out.println("4. Display Queue");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter name to add: ");
                    name = sc.nextLine();
                    queue.enqueue(name);
                    break;
                case 2:
                    queue.dequeue();
                    break;
                case 3:
                    System.out.print("Enter name to cancel: ");
                    name = sc.nextLine();
                    queue.cancelTicket(name);
                    break;
                case 4:
                    queue.displayQueue();
                    break;
                case 5:
                    System.out.println("Exiting system.");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }

        } while (choice != 5);

        sc.close();
    }
}
```



```
        System.out.print("Enter name to add: ");
        name = sc.nextLine();
        queue.enqueue(name);
        break;
    case 2:
        queue.dequeue();
        break;
    case 3:
        System.out.print("Enter name to cancel: ");
        name = sc.nextLine();
        queue.cancelTicket(name);
        break;
    case 4:
        queue.displayQueue();
        break;
    case 5:
        System.out.println("Exiting system.");
        break;
    default:
        System.out.println("Invalid choice. Try again.");

le (choice != 5);

ose();
```

```
--- Ticket Booking Menu ---
1. Add Person
2. Serve Person
3. Cancel Ticket
4. Display Queue
5. Exit
Enter your choice: 1
Enter name to add: varsha
varsha added to the booking queue.

--- Ticket Booking Menu ---
1. Add Person
2. Serve Person
3. Cancel Ticket
4. Display Queue
5. Exit
Enter your choice: 1
Enter name to add: karan
karan added to the booking queue.

--- Ticket Booking Menu ---
1. Add Person
2. Serve Person
3. Cancel Ticket
4. Display Queue
5. Exit
Enter your choice: 2
varsha has been served and removed from the queue.

--- Ticket Booking Menu ---
1. Add Person
2. Serve Person
3. Cancel Ticket
4. Display Queue
5. Exit
```

Assignment-6

Car wash service queue

```
import java.util.Scanner;

class Node {
    String carNumber;
    Node next;

    Node(String carNumber) {
        this.carNumber = carNumber;
        this.next = null;
    }
}
class Main {
    private Node front, rear;

    Main() {
        front = rear = null;
    }
    public void addNormalCar(String carNumber) {
        Node newNode = new Node(carNumber);
        if (rear == null) {
            front = rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        System.out.println("Normal car " + carNumber + " added at the end.");
    }
    public void addVIPCar(String carNumber) {
```

```
    Node newNode = new Node(carNumber);
    if (front == null) {
        front = rear = newNode;
    } else {
        newNode.next = front;
        front = newNode;
    }
    System.out.println(" VIP car " + carNumber + " added at the front.");
}
public void removeCar() {
    if (front == null) {
        System.out.println("No cars in queue to remove.");
        return;
    }

    System.out.println(" Car " + front.carNumber + " washed and removed.");
    front = front.next;
    if (front == null) rear = null;
}

public void displayQueue() {
    if (front == null) {
        System.out.println("Queue is empty.");
        return;
    }

    System.out.print("Current Queue: ");
    Node temp = front;
    while (temp != null) {
        System.out.print(temp.carNumber + " ");
        temp = temp.next;
    }
    System.out.println();
}
public static void main(String[] args) {
    Main queue = new Main();
    Scanner sc = new Scanner(System.in);
    int choice;
    String carNumber;

    do {
        System.out.println("\nCar Wash Service Queue Menu:");
        System.out.println("1. Add Normal Car");
        System.out.println("2. Add VIP Car");
        System.out.println("3. Remove Car After Washing");
        System.out.println("4. Display Queue");
        System.out.println("5. Exit");
        System.out.print("Enter choice: ");
        choice = sc.nextInt();
        sc.nextLine();

        switch (choice) {
```

```
            case 1:
                System.out.print("Enter car number: ");
                carNumber = sc.nextLine();
                queue.addNormalCar(carNumber);
                break;
            case 2:
                System.out.print("Enter VIP car number: ");
                carNumber = sc.nextLine();
                queue.addVIPCar(carNumber);
                break;
            case 3:
                queue.removeCar();
                break;
            case 4:
                queue.displayQueue();
                break;
            case 5:
                System.out.println("Exiting Car Wash System.");
                break;
            default:
                System.out.println("Invalid choice. Try again.");
        }

    } while (choice != 5);

    sc.close();
  }
}
```

```
 System.out.print("Enter car nu
 carNumber = sc.nextLine();
 queue.addNormalCar(carNumber);
 break;
se 2:
 System.out.print("Enter VIP ca
 carNumber = sc.nextLine();
 queue.addVIPCar(carNumber);
 break;
se 3:
 queue.removeCar();
 break;
se 4:
 queue.displayQueue();
 break;
se 5:
 System.out.println("Exiting Ca
 break;
fault:
 System.out.println("Invalid ch


hoice != 5);

;
```

```
Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 1
Enter car number: KA02JW6000
Normal car KA02JW6000 added at the end.

Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 2
Enter VIP car number: KA02MF1200
 VIP car KA02MF1200 added at the front.

Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 4
Current Queue: KA02MF1200 KA02JW6000

Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 3
```

```
Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 3
 Car KA02MF1200 washed and removed.

Car Wash Service Queue Menu:
1. Add Normal Car
2. Add VIP Car
3. Remove Car After Washing
4. Display Queue
5. Exit
Enter choice: 5
Exiting Car Wash System.


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment-7
Library Book Stack
import java.util.Stack;

```
public class Main {
    Stack<String> books = new Stack<>();

    public void addBook(String book) {
        books.push(book);
    }

    public void removeBook() {
        if (!books.isEmpty()) {
            System.out.println("Removed book: " + books.pop());
        } else {
            System.out.println("No books to remove.");
        }
    }

    public void peekBook() {
        if (!books.isEmpty()) {
            System.out.println("Top book: " + books.peek());
        } else {
            System.out.println("No books in stack.");
        }
    }

    public static void main(String[] args) {
        Main stack = new Main();
        stack.addBook("Operating System");
        stack.addBook("Computer Architecture");
        stack.peekBook();
        stack.removeBook();
        stack.peekBook();
    }
}
```

Main.java

```
            System.out.println("No books in stack.");
        }
    }

    public static void main(String[] args) {
        Main stack = new Main();
        stack.addBook("Operating System");
        stack.addBook("Computer Architecture");
        stack.peekBook();
        stack.removeBook();
        stack.peekBook();
    }
}
```

input

```
Top book: Computer Architecture
Removed book: Computer Architecture
Top book: Operating System


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment-8
Expression Evaluator

```
import java.util.*;

public class Main {
    static int precedence(char op) {
        switch (op) {
            case '+': case '-': return 1;
            case '*': case '/': return 2;
        }
        return -1;
    }

    public static String infixToPostfix(String expr) {
        StringBuilder postfix = new StringBuilder();
        Stack<Character> stack = new Stack<>();

        for (char ch : expr.toCharArray()) {
            if (Character.isDigit(ch)) {
                postfix.append(ch);
            } else if (ch == '(') {
                stack.push(ch);
            } else if (ch == ')') {
                while (!stack.isEmpty() && stack.peek() != '(')
                    postfix.append(stack.pop());
                stack.pop();
            } else {
                while (!stack.isEmpty() && precedence(ch) <= precedence(stack.peek()))
                    postfix.append(stack.pop());
                stack.push(ch);
            }
```

```
        }
        while (!stack.isEmpty()) {
            postfix.append(stack.pop());
        }
        return postfix.toString();
    }

    public static int evaluatePostfix(String postfix) {
        Stack<Integer> stack = new Stack<>();

        for (char ch : postfix.toCharArray()) {
            if (Character.isDigit(ch)) {
                stack.push(ch - '0');
            } else {
                int b = stack.pop();
                int a = stack.pop();
                switch (ch) {
                    case '+': stack.push(a + b); break;
                    case '-': stack.push(a - b); break;
                    case '*': stack.push(a * b); break;
                    case '/': stack.push(a / b); break;
                }
            }
        }
        return stack.pop();
    }

    public static void main(String[] args) {
        String expr = "98+(25+32*67/45)-12";
        String postfix = infixToPostfix(expr);
        System.out.println("Postfix: " + postfix);
        System.out.println("Result: " + evaluatePostfix(postfix));
    }
}
```

```
54          return stack.pop();
55      }
56
57      public static void main(String[] args) {
58          String expr = "98+(25+32*67/45)-12";
59          String postfix = infixToPostfix(expr);
60          System.out.println("Postfix: " + postfix);
61          System.out.println("Result: " + evaluatePostfix(postfix));
62      }
63  }
64
```

input

```
Postfix: 98253267*45/++12-
Result: -1


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment-9
Reverse Queue

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<>();
        Stack<String> stack = new Stack<>();

        queue.offer("10");
        queue.offer("20");
        queue.offer("30");
        queue.offer("40");
        queue.offer("50");

        while (!queue.isEmpty()) {
            stack.push(queue.poll());
        }
        while (!stack.isEmpty()) {
            queue.offer(stack.pop());
        }

        System.out.println("Reversed Queue: " + queue);
    }
}
```

Main.java

```
        Stack<String> stack = new Stack<>();

        queue.offer("10");
        queue.offer("20");
        queue.offer("30");
        queue.offer("40");
        queue.offer("50");

        while (!queue.isEmpty()) {
            stack.push(queue.poll());
        }
        while (!stack.isEmpty()) {
            queue.offer(stack.pop());
        }

        System.out.println("Reversed Queue: " + queue);
    }
}

```

input

```
Reversed Queue: [50, 40, 30, 20, 10]


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment-10
Student Admission

```
import java.util.LinkedList;

public class Main {
    public static void main(String[] args) {
        LinkedList<String> admissionQueue = new LinkedList<>();

        admissionQueue.addLast("Varsha");
        admissionQueue.addLast("Kunal");

        admissionQueue.addFirst("VIP Karan");

        String admittedStudent = admissionQueue.removeFirst();
        System.out.println("Admitted: " + admittedStudent);

        System.out.println("Remaining Queue: " + admissionQueue);
    }
}
```

```java
        LinkedList<String> admissionQueue = new LinkedList<>();

        admissionQueue.addLast("Varsha");
        admissionQueue.addLast("Kunal");

        admissionQueue.addFirst("VIP Karan");

        String admittedStudent = admissionQueue.removeFirst();
        System.out.println("Admitted: " + admittedStudent);

        System.out.println("Remaining Queue: " + admissionQueue);
    }
}

```

input

```
Admitted: VIP Karan
Remaining Queue: [Varsha, Kunal]


...Program finished with exit code 0
Press ENTER to exit console.
```