K. R. MANGALAM UNIVERSITY, GURUGRAM, HARYANA, INDIA

Practical File

Data Structure Report file

Name: Aditya Kumar Singh

Roll Number: 2401420002

Course: B-Tech CSE (Data Science)

Semester: 3rd

Submitted To:
Signature:

Dr. Swati Gupta

1. Browser History Navigation System (Using Stack Concept) :

BrowserHistory.java

```
.util.Stack;

public class BrowserHistory {
    Stack<String> back = new Stack<>();
    Stack<String> forward = new Stack<>();
    String currentPage = "Home";

    void visit(String url) {
        back.push(currentPage);
        currentPage = url;
        forward.clear();
        System.out.println("Visited: " + currentPage);
    }

    void goBack() {
        if (!back.isEmpty()) {
            forward.push(currentPage);
            currentPage = back.pop();
            System.out.println("Back to: " + currentPage);
        } else {
            System.out.println("No pages in back history");
        }
    }

    void goForward() {
        if (!forward.isEmpty()) {
            back.push(currentPage);
            currentPage = forward.pop();
            System.out.println("Forward to: " + currentPage);
        } else {
            System.out.println("No pages in forward history");
        }
    }

    public static void main(String[] args) {
        BrowserHistory bh = new BrowserHistory();
        bh.visit("google.com");
        bh.visit("youtube.com");
        bh.visit("github.com");
        bh.goBack();
        bh.goBack();
        bh.goForward();
    }
```

OUTPUT:

Output

```
Visited: google.com
Visited: youtube.com
Visited: github.com
Back to: youtube.com
Back to: google.com
Forward to: youtube.com
```

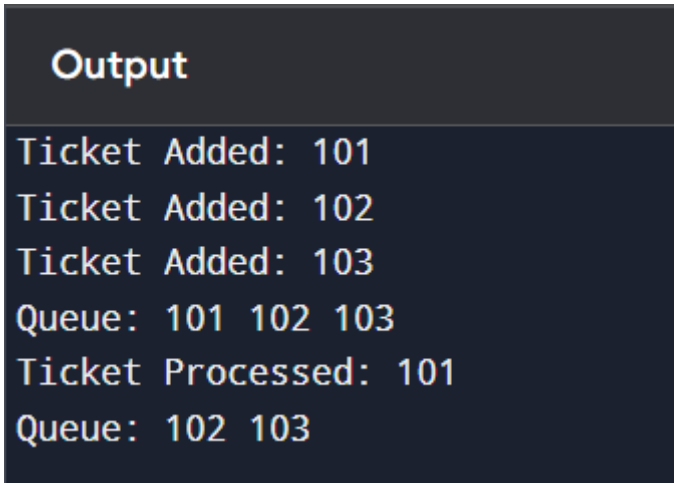
2. Ticketing System Using Queue (Linear Queue Implementation)

```java
class LinearQueue {
    int front = -1, rear = -1;
    int size;
    int[] queue;

    LinearQueue(int size) {
        this.size = size;
        queue = new int[size];
    }

    void enqueue(int ticket) {
        if (rear == size - 1) {
            System.out.println("Queue is Full");
            return;
        }
        if (front == -1) front = 0;
        queue[++rear] = ticket;
        System.out.println("Ticket Added: " + ticket);
    }

    void dequeue() {
```

```java
    void dequeue() {
        if (front == -1 || front > rear) {
            System.out.println("Queue is Empty");
            return;
        }
        System.out.println("Ticket Processed: " + queue[front++]);
    }

    void display() {
        if (front == -1 || front > rear) {
            System.out.println("Queue is Empty");
            return;
        }
        System.out.print("Queue: ");
        for (int i = front; i <= rear; i++)
            System.out.print(queue[i] + " ");
        System.out.println();
    }

    public static void main(String[] args) {
        LinearQueue q = new LinearQueue(5);
```

```
86
87      public static void main(String[] args) {
88          LinearQueue q = new LinearQueue(5);
89          q.enqueue(101);
90          q.enqueue(102);
91          q.enqueue(103);
92          q.display();
93          q.dequeue();
94          q.display();
95      }
96  }
```

OUTPUT :



3. Singly Linked List Operations (Insert, Delete, Search, Display)

```
100 class SinglyLinkedList {
101     class Node {
102         int data;
103         Node next;
104         Node(int data) { this.data = data; }
105     }
106
107     Node head;
108
109     void insert(int data) {
110         Node newNode = new Node(data);
111         newNode.next = head;
112         head = newNode;
113     }
114
115     void delete(int key) {
116         Node temp = head, prev = null;
117
118         if (temp != null && temp.data == key) {
119             head = temp.next;
120             return;
```

```
121         }
122         while (temp != null && temp.data != key) {
123             prev = temp;
124             temp = temp.next;
125         }
126         if (temp == null) return;
127
128         prev.next = temp.next;
129     }
130
131     void search(int key) {
132         Node temp = head;
133         while (temp != null) {
134             if (temp.data == key) {
135                 System.out.println("Found: " + key);
136                 return;
137             }
138             temp = temp.next;
139         }
140         System.out.println("Not Found");
141     }
142
```

```
142
143     void display() {
144         Node temp = head;
145         System.out.print("List: ");
146         while (temp != null) {
147             System.out.print(temp.data + " ");
148             temp = temp.next;
149         }
150         System.out.println();
151     }
152
153     public static void main(String[] args) {
154         SinglyLinkedList s = new SinglyLinkedList();
155         s.insert(10);
156         s.insert(20);
157         s.insert(30);
158         s.display();
159         s.search(20);
160         s.delete(20);
161         s.display();
162     }
```

OUTPUT:

```
Output
List: 30 20 10
Found: 20
List: 30 10
```

4. Circular Singly Linked List (Insert, Search, Delete, Display)

```
166
167 class CircularLinkedList {
168     class Node {
169         int data;
170         Node next;
171         Node(int data) { this.data = data; }
172     }
173
174     Node last = null;
175
176     void insert(int data) {
177         Node newNode = new Node(data);
178         if (last == null) {
179             last = newNode;
180             last.next = last;
181         } else {
182             newNode.next = last.next;
183             last.next = newNode;
184             last = newNode;
185         }
186     }
```

```
187
188     void delete(int key) {
189         if (last == null) return;
190
191         Node curr = last.next, prev = last;
192
193         while (curr.data != key) {
194             if (curr == last) return;
195             prev = curr;
196             curr = curr.next;
197         }
198
199         if (curr == last && curr.next == last) {
200             last = null;
201             return;
202         }
203
204         if (curr == last) last = prev;
205         prev.next = curr.next;
206     }
207
```

```
208     void search(int key) {
209         if (last == null) {
210             System.out.println("List Empty");
211             return;
212         }
213         Node temp = last.next;
214         do {
215             if (temp.data == key) {
216                 System.out.println("Found: " + key);
217                 return;
218             }
219             temp = temp.next;
220         } while (temp != last.next);
221 
222         System.out.println("Not Found");
223     }
224 
225     void display() {
226         if (last == null) {
227             System.out.println("List Empty");
228             return;
229         }
230         Node temp = last.next;
231         System.out.print("Circular List: ");
232         do {
233             System.out.print(temp.data + " ");
234             temp = temp.next;
235         } while (temp != last.next);
236         System.out.println();
237     }
238 
239     public static void main(String[] args) {
240         CircularLinkedList c = new CircularLinkedList();
241         c.insert(10);
242         c.insert(20);
243         c.insert(30);
244         c.display();
245         c.search(20);
246         c.delete(20);
247         c.display();
248     }
249 }
```

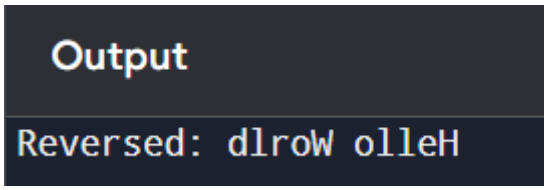
OUTPUT:

```
Output
Circular List: 10 20 30
Found: 20
Circular List: 10 30
```

5. Reverse a String Using Stack

```java

import java.util.Stack;

public class ReverseString {
    public static void main(String[] args) {
        String str = "Hello World";
        Stack<Character> stk = new Stack<>();

        for (char c : str.toCharArray()) {
            stk.push(c);
        }

        String rev = "";
        while (!stk.isEmpty()) {
            rev += stk.pop();
        }

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

OUTPUT:



6. Check Balanced Parentheses Using Stack

```java
import java.util.Stack;

public class BalancedParentheses {
    static boolean isBalanced(String exp) {
        Stack<Character> stack = new Stack<>();

        for (char ch : exp.toCharArray()) {
            if (ch == '(' || ch == '{' || ch == '[') stack.push(ch);
            else if (ch == ')' || ch == '}' || ch == ']') {
                if (stack.isEmpty()) return false;

                char top = stack.pop();
                if ((ch == ')' && top != '(') ||
                    (ch == '}' && top != '{') ||
                    (ch == ']' && top != '[')) return false;
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        String exp = "{(a+b)*[c-d]}";
        System.out.println(isBalanced(exp) ? "Balanced" : "Not Balanced");
    }
}
```

OUTPUT: