K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

LAB FILE

DATA STRUCTURES

(ENCS 253)

SUBMITTED BY:

B. TECH CSE

SUBMITTED TO:

MS SUMAN

ASSISTANT PROFESSOR

SOET

School of Engineering & Technology

K. R. MANGALAM UNIVERSITY

Sohna, Haryana 122103, India

INDEX

S.No	PROGRAM	DATE	SIGNATURE
1.	Design, Develop and Implement a menu driven Program for the following Array operations a. Creating an Array of N Integer Elements b. Display of Array Elements c. Inserting an Element at a given valid Position (POS) d. Deleting an Element at a given valid Position(POS) e. Exit.	23-9-24	
2	Design, Develop and Implement a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a. Push an Element on to Stack b. Pop an Element from Stack c. Demonstrate how Stack can be used to check Palindrome d. Demonstrate Overflow and Underflow situations on Stack e. Display the status of Stack f. Exit	30-9-24	
3	Design, Develop and Implement a Program for converting an Infix Expression to PostfixExpression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.	21-10-24	
4	Design, Develop and Implement a menu driven Program for the following operations on QUEUE of Characters a. Insert an Element on to Linear QUEUE b. Delete an Element from Linear QUEUE	28-10-24	
5	Design, Develop and Implement a menu driven Program in for the following operations on Circular QUEUE of Characters a. Insert an Element on to Circular QUEUE b. Delete an Element from Circular QUEUE c. Demonstrate Overflow and Underflow situations on Circular QUEUE d. Display	04-11-24	
6	Design, Develop and Implement a menu driven Program for the following operations onSingly Linked List (SLL) a. Create a SLL. b. Insert at Beginning c. Insert at Last d. Insert at any random location	11-11-24	

AIM: Design, Develop and Implement a menu driven Program for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements

c. Inserting an Element at a given valid Position (POS)

d. Deleting an Element at a given valid Position(POS)

e. Exit.

CODE:

```
import java.util.Scanner;
public class ArrayOperations {
   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] array = new int[n];
        int size = 0; // Tracks the actual number of elements

        while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Create Array");
        System.out.println("2. Display Array");
        System.out.println("3. Insert Element");
        System.out.println("4. Delete Element");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1: // Create Array
           System.out.println("Enter elements:");
            for (int i = 0; i < n; i++) {
            array[i] = scanner.nextInt();
            }
            size = n;
           System.out.println("Array created.");
            break;

            case 2: // Display Array
           System.out.println("Array elements:");
            for (int i = 0; i < size; i++) {
            System.out.print(array[i] + " ");
            }
            System.out.println();
            break;

            case 3: // Insert Element
            if (size >= n) {
            System.out.println("Array is full. Cannot insert element.");
            } else {
            System.out.print("Enter element to insert: ");
            int element = scanner.nextInt();
            System.out.print("Enter position to insert (1 to " + (size + 1) + "): ");
            int pos = scanner.nextInt() - 1;
            if (pos >= 0 && pos <= size) {
                    for (int i = size; i > pos; i--) {
                    array[i] = array[i - 1];
```

```
                    }
                    array[pos] = element;
                    size++;
                System.out.println("Element inserted.");
                } else {
                System.out.println("Invalid position.");
                }
                }
                break;

                case 4: // Delete Element
            System.out.print("Enter position to delete (1 to " + size + "): ");
                int delPos = scanner.nextInt() - 1;
                if (delPos >= 0 && delPos < size) {
                for (int i = delPos; i < size - 1; i++) {
                    array[i] = array[i + 1];
                }
                size--;
                System.out.println("Element deleted.");
                } else {
                System.out.println("Invalid position.");
                }
                break;

                case 5: // Exit
            System.out.println("Exiting...");
                scanner.close();
                return;

                default:
                System.out.println("Invalid choice. Please try again.");
        }
        }
    }
}
```

OUTPUT:

```
Enter the number of elements in the array: 4
Menu:
1. Create Array
2. Display Array
3. Insert Element
4. Delete Element
5. Exit
Enter your choice: 1
Enter elements:
3
4
5
4
Array created.
```

AIM: Design, Develop and Implement a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)
a. Push an Element on to Stack
b. Pop an Element from Stack
c. Demonstrate how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit

CODE:

```
import java.util.Scanner;
public class StackOperations {
   private static final int MAX = 10;
   private int[] stack = new int[MAX];
   private int top = -1;

   public boolean isFull() {
         return top == MAX - 1;
   }

   public boolean isEmpty() {
         return top == -1;
   }

   public void push(int element) {
         if (isFull()) {
         System.out.println("Stack Overflow. Cannot push element.");
         } else {
         stack[++top] = element;
         System.out.println("Pushed " + element + " onto the stack.");
         }
   }

   public void pop() {
         if (isEmpty()) {
         System.out.println("Stack Underflow. Cannot pop element.");
         } else {
         int element = stack[top--];
         System.out.println("Popped " + element + " from the stack.");
         }
   }

   public boolean isPalindrome() {
         int start = 0;
         int end = top;
         while (start < end) {
         if (stack[start] != stack[end]) {
                  return false;
         }
         start++;
         end--;
         }
         return true;
   }

   public void display() {
         if (isEmpty()) {
```

```
        System.out.println("Stack is empty.");
        } else {
        System.out.print("Stack elements: ");
        for (int i = 0; i <= top; i++) {
                System.out.print(stack[i] + " ");
        }
        System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StackOperations stackOps = new StackOperations();

        while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Push an element");
        System.out.println("2. Pop an element");
        System.out.println("3. Check if stack is a palindrome");
        System.out.println("4. Display stack");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
                case 1: // Push an Element
            System.out.print("Enter element to push: ");
                int element = scanner.nextInt();
                stackOps.push(element);
                break;

                case 2: // Pop an Element
                stackOps.pop();
                break;

                case 3: // Check if Stack is a Palindrome
                if (stackOps.isPalindrome()) {
                System.out.println("Stack is a palindrome.");
                } else {
                System.out.println("Stack is not a palindrome.");
                }
                break;

                case 4: // Display Stack
                stackOps.display();
                break;

                case 5: // Exit
            System.out.println("Exiting...");
                scanner.close();
                return;

                default:
            System.out.println("Invalid choice. Please try again.");
        }
        }
    }
}
```

OUTPUT:

```
Menu:
1. Push an element
2. Pop an element
3. Check if stack is a palindrome
4. Display stack
5. Exit
Enter your choice: 1
Enter element to push: 3
Pushed 3 onto the stack
```

AIM: Design, Develop and Implement a Program for converting an Infix Expression to PostfixExpression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

CODE:

```
infix = input("Enter infix Expression: ")
postfix = ""
stack = []
order = {"+": 1, "-": 1, "*": 2, "/": 2, "^": 3}

for i in infix:
    if i.isalnum():  # Check if the character is an operand
        postfix += i
    elif i == "(":
        stack.append(i)
    elif i == ")":
        while stack and stack[-1] != "(":
        postfix += stack.pop()
        stack.pop()  # Pop the '('
    else:
        while stack and stack[-1] != "(" and order[stack[-1]] >= order[i]:
        postfix += stack.pop()
        stack.append(i)

while stack:
    postfix += stack.pop()

print("Postfix Expression:", postfix).
```

OUTPUT: Enter infix Expression: a+(b-c)
Postfix Expression: abc-+

AIM: Design, Develop and Implement a menu driven Program for the following operations on QUEUE of Characters

a. Insert an Element on to Linear QUEUE

b. Delete an Element from Linear QUEUE

CODE:

```
import java.util.Scanner;
public class LinearQueue {
   private static final int MAX = 10;
   private char[] queue = new char[MAX];
   private int front = -1;
   private int rear = -1;

   // Method to check if the queue is full
   public boolean isFull() {
         return rear == MAX - 1;
   }

   // Method to check if the queue is empty
   public boolean isEmpty() {
         return front == -1 || front > rear;
   }

   // Method to insert an element into the queue
   public void enqueue(char element) {
         if (isFull()) {
         System.out.println("Queue Overflow. Cannot insert element.");
         } else {
         if (front == -1) {
                  front = 0;
         }
         queue[++rear] = element;
         System.out.println("Inserted " + element + " into the queue.");
         }
   }

   // Method to delete an element from the queue
   public void dequeue() {
         if (isEmpty()) {
         System.out.println("Queue Underflow. Cannot delete element.");
         } else {
         char element = queue[front++];
         System.out.println("Deleted " + element + " from the queue.");
         if (front > rear) {
                  front = rear = -1; // Reset the queue if it becomes empty
         }
         }
   }

   // Method to display the elements of the queue
   public void display() {
         if (isEmpty()) {
         System.out.println("Queue is empty.");
         } else {
         System.out.print("Queue elements: ");
         for (int i = front; i <= rear; i++) {
                  System.out.print(queue[i] + " ");
```

```
        }
        System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinearQueue queue = new LinearQueue();

        while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Insert an element into the queue");
        System.out.println("2. Delete an element from the queue");
        System.out.println("3. Display the queue");
        System.out.println("4. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
                case 1: // Enqueue operation
            System.out.print("Enter character to insert: ");
                char element = scanner.next().charAt(0);
                queue.enqueue(element);
                break;

                case 2: // Dequeue operation
                queue.dequeue();
                break;

                case 3: // Display the queue
                queue.display();
                break;

                case 4: // Exit
            System.out.println("Exiting...");
                scanner.close();
                return;

                default:
            System.out.println("Invalid choice. Please try again.");
        }
        }
    }
}
```

OUTPUT:

```
Menu:
1. Insert an element into the queue
2. Delete an element from the queue
3. Display the queue
4. Exit
Enter your choice: 1
Enter character to insert: 3
Inserted 3 into the queue.
```

AIM: Design, Develop and Implement a menu driven Program in java for the following operations on Circular QUEUE of Characters
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display

CODE:

```
import java.util.Scanner;
public class CircularQueue {
   private static final int MAX = 5;
   private char[] queue = new char[MAX];
   private int front = -1;
   private int rear = -1;

   // Method to check if the queue is full
   public boolean isFull() {
         return (rear + 1) % MAX == front;
   }

   // Method to check if the queue is empty
   public boolean isEmpty() {
         return front == -1;
   }

   // Method to insert an element into the circular queue
   public void enqueue(char element) {
         if (isFull()) {
         System.out.println("Queue Overflow. Cannot insert element.");
         } else {
         if (isEmpty()) {
                  front = 0;
         }
         rear = (rear + 1) % MAX;
         queue[rear] = element;
         System.out.println("Inserted " + element + " into the queue.");
         }
   }

   // Method to delete an element from the circular queue
   public void dequeue() {
         if (isEmpty()) {
         System.out.println("Queue Underflow. Cannot delete element.");
         } else {
         char element = queue[front];
         if (front == rear) {
                  front = rear = -1; // Reset queue if it becomes empty
         } else {
                  front = (front + 1) % MAX;
         }
         System.out.println("Deleted " + element + " from the queue.");
         }
   }

   // Method to display the elements of the circular queue
   public void display() {
         if (isEmpty()) {
         System.out.println("Queue is empty.");
         } else {
```

```
        System.out.print("Queue elements: ");
        int i = front;
        while (true) {
                System.out.print(queue[i] + " ");
                if (i == rear) break;
                i = (i + 1) % MAX;
        }
        System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CircularQueue queue = new CircularQueue();

        while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Insert an element into the queue");
        System.out.println("2. Delete an element from the queue");
        System.out.println("3. Display the queue");
        System.out.println("4. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
                case 1: // Enqueue operation
            System.out.print("Enter character to insert: ");
                char element = scanner.next().charAt(0);
                queue.enqueue(element);
                break;

                case 2: // Dequeue operation
                queue.dequeue();
                break;

                case 3: // Display the queue
                queue.display();
                break;

                case 4: // Exit
            System.out.println("Exiting...");
                scanner.close();
                return;

                default:
            System.out.println("Invalid choice. Please try again.");
        }
        }
    }
}
```

OUTPUT:

```
Menu:
1. Insert an element into the queue
2. Delete an element from the queue
3. Display the queue
4. Exit
Enter your choice: 1
Enter character to insert: 3
Inserted 3 into the queue.
```

AIM: Design, Develop and Implement a menu driven Program for the following operations onSingly Linked List (SLL)
a. Create a SLL.
b. Insert at Beginning
c. Insert at Last
d. Insert at any random location
e. Delete from Beginning
f. Delete from Last
g. Delete node after specified location
h. Search for an element
i. Show
j. Exit

CODE:

```
 import java.util.Scanner;
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

public class SinglyLinkedList {
    private Node head;

    // Method to insert at the beginning of the list
    public void insertAtBeginning(int data) {
        Node newNode = new Node(data);
        newNode.next = head;
        head = newNode;
        System.out.println(data + " inserted at the beginning.");
    }

    // Method to insert at the end of the list
    public void insertAtEnd(int data) {
        Node newNode = new Node(data);
        if (head == null) {
        head = newNode;
        } else {
        Node temp = head;
        while (temp.next != null) {
                temp = temp.next;
        }
        temp.next = newNode;
        }
        System.out.println(data + " inserted at the end.");
    }

    // Method to insert at a specific position
    public void insertAtPosition(int data, int position) {
        Node newNode = new Node(data);
        if (position == 1) {
        newNode.next = head;
        head = newNode;
        } else {
        Node temp = head;
```

```
        for (int i = 1; i < position - 1 && temp != null; i++) {
                temp = temp.next;
        }
        if (temp != null) {
                newNode.next = temp.next;
                temp.next = newNode;
                System.out.println(data + " inserted at position " + position);
        } else {
           System.out.println("Invalid position.");
        }
        }
}

// Method to delete from the beginning
public void deleteFromBeginning() {
        if (head == null) {
        System.out.println("List is empty.");
        } else {
        System.out.println("Deleted " + head.data + " from the beginning.");
        head = head.next;
        }
}

// Method to delete from the end
public void deleteFromEnd() {
        if (head == null) {
        System.out.println("List is empty.");
        } else if (head.next == null) {
        System.out.println("Deleted " + head.data + " from the end.");
        head = null;
        } else {
        Node temp = head;
        while (temp.next.next != null) {
                temp = temp.next;
        }
        System.out.println("Deleted " + temp.next.data + " from the end.");
        temp.next = null;
        }
}

// Method to delete a node after a specific position
public void deleteAfterPosition(int position) {
        if (head == null) {
        System.out.println("List is empty.");
        } else {
        Node temp = head;
        for (int i = 1; i < position && temp != null; i++) {
                temp = temp.next;
        }
        if (temp != null && temp.next != null) {
           System.out.println("Deleted " + temp.next.data + " after position " + position);
                temp.next = temp.next.next;
        } else {
           System.out.println("Invalid position.");
        }
        }
}

// Method to search for an element
```

```
public boolean search(int data) {
        Node temp = head;
        while (temp != null) {
        if (temp.data == data) {
                return true;
        }
        temp = temp.next;
        }
        return false;
}

// Method to display the list
public void display() {
        if (head == null) {
        System.out.println("List is empty.");
        } else {
        Node temp = head;
        System.out.print("Singly Linked List: ");
        while (temp != null) {
                System.out.print(temp.data + " ");
                temp = temp.next;
        }
        System.out.println();
        }
}

public static void main(String[] args) {
        SinglyLinkedList list = new SinglyLinkedList();
        Scanner scanner = new Scanner(System.in);

        while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. Insert at Beginning");
        System.out.println("2. Insert at End");
        System.out.println("3. Insert at Position");
        System.out.println("4. Delete from Beginning");
        System.out.println("5. Delete from End");
        System.out.println("6. Delete after Position");
        System.out.println("7. Search for Element");
        System.out.println("8. Display List");
        System.out.println("9. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
                case 1:
           System.out.print("Enter data to insert at the beginning: ");
              int dataBegin = scanner.nextInt();
           list.insertAtBeginning(dataBegin);
              break;

              case 2:
           System.out.print("Enter data to insert at the end: ");
              int dataEnd = scanner.nextInt();
              list.insertAtEnd(dataEnd);
              break;

              case 3:
           System.out.print("Enter data to insert: ");
```

```
                int dataPos = scanner.nextInt();
            System.out.print("Enter position to insert at: ");
                int position = scanner.nextInt();
            list.insertAtPosition(dataPos, position);
                break;

                case 4:
                list.deleteFromBeginning();
                break;

                case 5:
                list.deleteFromEnd();
                break;

                case 6:
            System.out.print("Enter position after which to delete: ");
                int delPosition = scanner.nextInt();
            list.deleteAfterPosition(delPosition);
                break;

                case 7:
            System.out.print("Enter element to search for: ");
                int searchData = scanner.nextInt();
                if (list.search(searchData)) {
                System.out.println(searchData + " found in the list.");
                } else {
                System.out.println(searchData + " not found in the list.");
                }
                break;
                case 8:
                list.display();
                break;

                case 9:
            System.out.println("Exiting...");
                scanner.close();
                return;
                default:
            System.out.println("Invalid choice. Please try again.");
        }
        }
    }
}
```

OUTPUT:

```
Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete from Beginning
5. Delete from End
6. Delete after Position
7. Search for Element
8. Display List
9. Exit
Enter your choice: 1
Enter data to insert at the beginning: 3
3 inserted at the beginning.
```