Student ID: 202112047

IT623 - Lab Assignment 3

1. **Program to rearrange array elements having positive and negative elements such that the negative elements appear before the positive ones**.

   Code:

```
public class Program1 {
	static void print(int arr[]) {
		for (int i = 0; i < arr.length; i++)
			System.out.print(arr[i]+" ");

	}

	static void rearrange(int arr[]) {
		int key, j;
		for (int i = 0; i < arr.length; i++) {
			key = arr[i];

			if (key > 0)
				continue;

			j = i - 1;
			while (j >= 0 && arr[j] > 0) {
				arr[j + 1] = arr[j];
				j = j - 1;
			}
			arr[j + 1] = key;
		}
	}
```

```
        public static void main(String[] args) {
        int arr[]= {1,100,-1,5,-3,5,-8,-4};
        System.out.println("Original Array : ");
        print(arr);
        System.out.println("\n\nRearrange Array : ");
        rearrange(arr);
        print(arr);
    }
}
```

Output Snapshot:

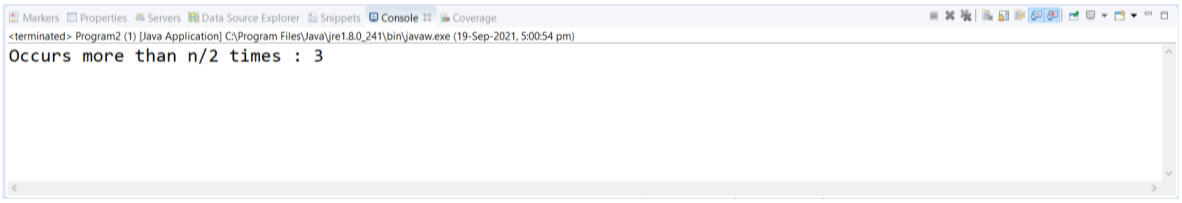
2. **Program to find the number which occurs more than n/2 times where n is the number of elements in the array(If n is odd then take floor(n/2)).**

Code:

```
public class Program2 {

    static void findOccurs(int arr[]) {
        int maxCount = 0, index = -1;
        for (int i = 0; i < arr.length; i++) {
            int count = 0;
```

```
                for (int j = 0; j < arr.length; j++) {
                        if (arr[i] == arr[j])
                                count++;
                }

                if (count > maxCount) {
                        maxCount = count;
                        index = i;
                }
        }

        if (maxCount > (arr.length / 2))
                System.out.println("Occurs more than n/2 times :
"+arr[index]);
        else
                System.out.println("No more than occurs");
    }

    public static void main(String[] args) {
        int arr[] = { 1, 2, 3, 1, 3, 2, 2, 2, 3, 3, 3, 3, 3, 3 };

        findOccurs(arr);
    }

}
```

Output Snapshot:



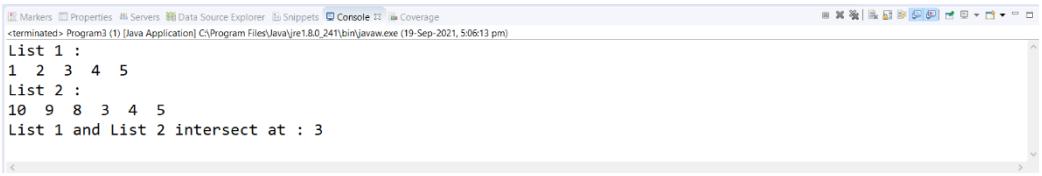
3. Program that finds the intersection of two linked lists.

Code:

```
import java.util.HashSet;

class Node {
	int data;
	Node next;

	Node(int x) {
		data = x;
		next = null;
	}
}

class Program3 {

	public static void main(String[] args) {

		Node n1 = new Node(1);
		n1.next = new Node(2);
		n1.next.next = new Node(3);
		n1.next.next.next = new Node(4);
		n1.next.next.next.next = new Node(5);

		Node n2 = new Node(10);
		n2.next = new Node(9);
		n2.next.next = new Node(8);
		n2.next.next.next = n1.next.next;

		System.out.println("List 1 :");
		print(n1);
		System.out.println("List 2 :");
```

```
        print(n2);

        System.out.println("List 1 and List 2 intersect at : " +
    intersectNode(n1, n2).data);

    }

    public static void print(Node n) {
        Node current = n;
        while (current != null) {
            System.out.print(current.data + "  ");
            current = current.next;
        }
        System.out.println();
    }

    public static Node intersectNode(Node n1, Node n2) {

        HashSet<Node> hs = new HashSet<Node>();
        while (n1 != null) {
            hs.add(n1);
            n1 = n1.next;
        }
        while (n2 != null) {
            if (hs.contains(n2)) {
                return n2;
            }
            n2 = n2.next;
        }
        return null;
    }
}
```

Output Snapshot:

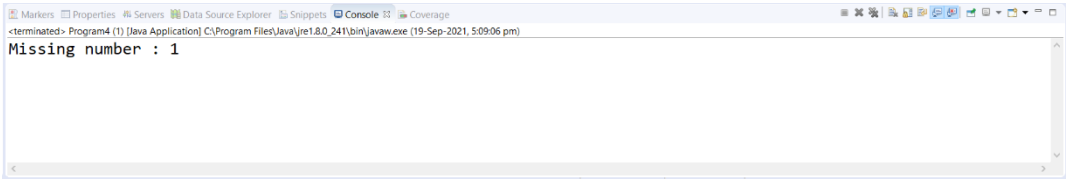


4. **Find the missing number in an unsorted array between 1 to n.**

Code:

```
import java.util.Arrays;

public class Program4 {

	public static void main(String[] args) {
		int[] no = { 3, 4, 5, 2 };

		Arrays.sort(no);

		Integer missing = null;

		for (int i = 0; i < no.length; i++) {
			int index = i + 1;
			missing = index;
			break;
		}
		System.out.println("Missing number : " + missing);
	}

}
```

Output Snapshot:



5. **Merge two sorted linked list such that the resultant list is also a sorted list.**

Code:

```
public class Program5 {

Node head = null;
Node tail = null;

static class Node {
        int data;
        Node next;

        Node(int x) {
                data = x;
                next = null;
        }
}

void add(int data) {
        Node node = new Node(data);
```

```
            if (head == null) {
                head = node;
                tail = node;
        } else {
                tail.next = node;
                tail = node;
        }
}

void print() {
        Node current = head;

        if (head == null) {
                System.out.println("List is empty");
        }

        while (current != null) {
                System.out.print(current.data + ", ");
                current = current.next;
        }
}

static Node merge(Node h1, Node h2) {
        if (h1 == null)
                return h2;
        if (h2 == null)
                return h1;

        if (h1.data < h2.data) {
                h1.next = merge(h1.next, h2);
                return h1;
        } else {
                h2.next = merge(h1, h2.next);
                return h2;
```

```
        }
}

static void printList(Node node) {
        while (node != null) {
                System.out.printf("%d ", node.data);
                node = node.next;
        }
}

public static void main(String[] args) {
        Program5 p1 = new Program5();
        Program5 p2 = new Program5();

        p1.add(1);
        p1.add(2);
        p1.add(5);
        p1.add(9);
        p1.add(10);

        p2.add(2);
        p2.add(3);
        p2.add(6);
        p2.add(11);

        System.out.println("List 1 : ");
        p1.print();

        System.out.println("\nList 2 : ");
        p2.print();

        Node mergedhead = merge(p1.head, p2.head);
        System.out.println("\nMerged Two sorted linked list : ");
        printList(mergedhead);
```

```
        }
}
```

Output Snapshot: