Lesson Plan

Java Array - 3

PW SKILLS

Pre-requisites:

- Java Syntax
- Loops

List of patterns involved:

Problem solving on Arrays

1. Sort the array of 0's and 1's
2. Move all negative numbers to beginning and positive to end with constant extra space.
3. Sort the array of 0's , 1's and 2's
4. Merge two sorted arrays
5. Find the next permutations of Array
6. Trapping Rain Water

Q1. Sort the array of 0's and 1's .

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] array = {1, 0, 1, 0, 0, 1, 1, 0, 1, 0};

        System.out.print("Original Array: ");
        displayArray(array);

        moveZerosToFront(array);

        System.out.print("Modified Array: ");
        displayArray(array);
    }

    static void moveZerosToFront(int[] arr) {
        int i = 0;
        int j = 0;

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

```
    static void displayArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:

```
Original Array: 1 0 1 0 0 1 1 0 1 0
Modified Array: 0 0 0 0 0 1 1 1 1 1
```

```
Original Array: 1 0
Modified Array: 0 1
```

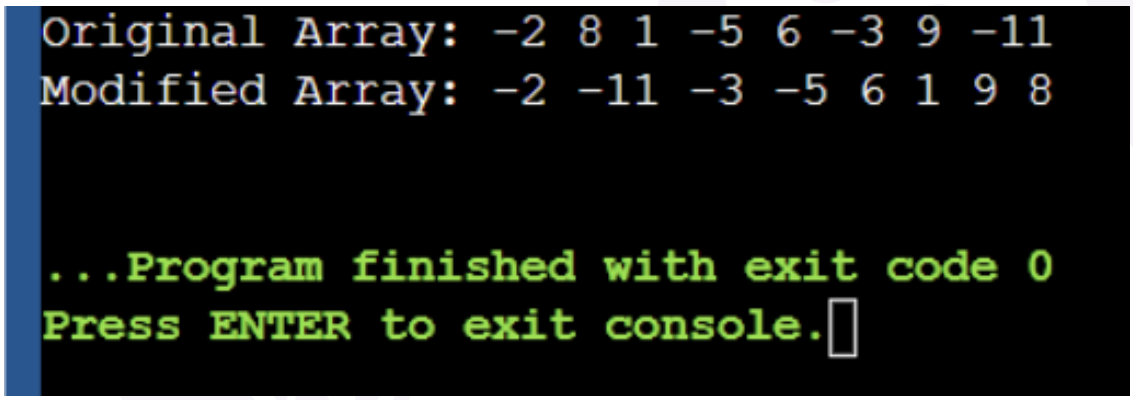
Q2. Move all negative numbers to beginning and positive to end with constant extra space.

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] array = {-2, 8, 1, -5, 6, -3, 9, -11};

        System.out.print("Original Array: ");
        displayArray(array);

        moveNegativesToBeginning(array);

        System.out.print("Modified Array: ");
        displayArray(array);
    }

    static void moveNegativesToBeginning(int[] arr) {
        int i = 0;
        int j = arr.length - 1;

        while (i ⩽ j) {
            if (arr[i] < 0) {
                i++;
            } else if (arr[j] ⩾ 0) {
                j--;
            } else {
```

```
                // Swap negative at i with positive at j
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                i++;
                j--;
            }
        }
    }

    static void displayArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:



Q3. Sort the array of 0's , 1's and 2's.

To sort an array containing 0s, 1s, and 2s, you can use the Dutch National Flag algorithm. This algorithm efficiently sorts an array with three distinct values and that too in O(n) time complexity.

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] array = {2, 0, 1, 2, 1, 0, 1, 2, 0};
        // Displaying the original array
        System.out.print("Original Array: ");
        displayArray(array);

        // Sort the array of 0s, 1s, and 2s
        sortColors(array);
```

```
        // Displaying the sorted array
        System.out.print("Sorted Array: ");
        displayArray(array);
    }

    static void sortColors(int[] arr) {
        int low = 0;
        int high = arr.length - 1;
        int i = 0;

        while (i <= high) {
            if (arr[i] == 0) {
                // Swap 0 at i with element at low
                int temp = arr[i];
                arr[i] = arr[low];
                arr[low] = temp;
                i++;
                low++;
            } else if (arr[i] == 2) {
                // Swap 2 at i with element at high
                int temp = arr[i];
                arr[i] = arr[high];
                arr[high] = temp;
                high--;
            } else {
                i++;
            }
        }
    }

    static void displayArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:

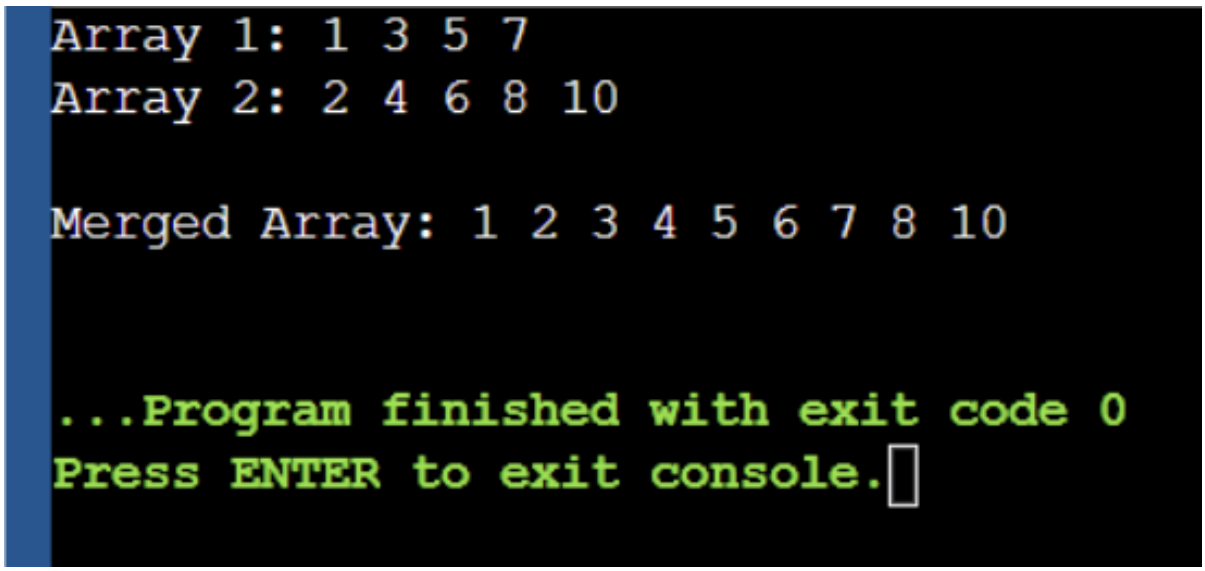
```
Original Array: 2 0 1 2 1 0 1 2 0
Sorted Array: 0 0 0 1 1 1 2 2 2


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

Q4. Merge two sorted arrays.

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] arr1 = {1, 3, 5, 7};
        int[] arr2 = {2, 4, 6, 8, 10};

        System.out.print("Array 1: ");
        displayArray(arr1);

        System.out.print("Array 2: ");
        displayArray(arr2);

        int[] mergedArray = mergeSortedArrays(arr1, arr2);

        System.out.print("\nMerged Array: ");
        displayArray(mergedArray);
    }

    static int[] mergeSortedArrays(int[] arr1, int[] arr2) {
        int n1 = arr1.length;
        int n2 = arr2.length;
        int[] mergedArray = new int[n1 + n2];

        int i = 0, j = 0, k = 0;

        while (i < n1 && j < n2) {
            if (arr1[i] ⩽ arr2[j]) {
                mergedArray[k++] = arr1[i++];
            } else {
                mergedArray[k++] = arr2[j++];
            }
        }

        while (i < n1) {
            mergedArray[k++] = arr1[i++];
        }

        while (j < n2) {
            mergedArray[k++] = arr2[j++];
        }

        return mergedArray;
    }

    static void displayArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:



Q5. Find the next permutations of Array .

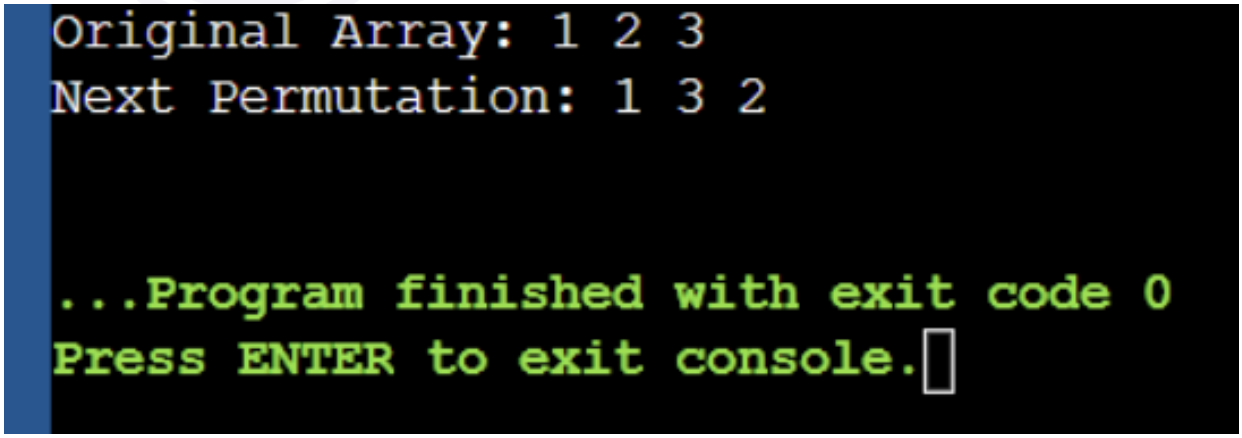
Note :- If not possible then print the sorted order in ascending order.

Code:

```
import java.util.Arrays;

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

        System.out.print("Original Array: ");
        displayArray(arr);

        nextPermutation(arr);

        System.out.print("Next Permutation: ");
        displayArray(arr);
    }

    static void nextPermutation(int[] arr) {
        int i = arr.length - 2;

        // Find the largest index i such that arr[i] < arr[i + 1]
        while (i ⩾ 0 && arr[i] ⩾ arr[i + 1]) {
            i--;
        }

        if (i ⩾ 0) {
            // Find the largest index j greater than i such that
arr[i] < arr[j]
            int j = arr.length - 1;
```

```
            while (arr[j] <= arr[i]) {
                j--;
            }

            // Swap the elements at indices i and j
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }

        // Reverse the elements from index i + 1 to the end of
the array
        reverseArray(arr, i + 1, arr.length - 1);
    }

    static void reverseArray(int[] arr, int start, int end) {
        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }

    static void displayArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:



Q6. Trapping Rain Water

Code:

```
class Main {
 public static int maxWater(int[] arr, int n) {

  int res = 0;

  for (int i = 1; i < n - 1; i++) {

   int left = arr[i];
   for (int j = 0; j < i; j++) {
    left = Math.max(left, arr[j]);
   }
   int right = arr[i];
   for (int j = i + 1; j < n; j++) {
    right = Math.max(right, arr[j]);
   }
   res += Math.min(left, right) - arr[i];
  }
  return res;
 }

 public static void main(String[] args) {
  int[] arr = {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1};
  int n = arr.length;

  System.out.print("Rain Water Trapped = " + maxWater(arr, n));
 }
}
```

Output:

```
Rain Water Trapped = 6

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