Algorithm and Data Structures – Lab Assignment

Array | Stack | Queue

1. Create an array of size 10 with the values 10,20,30,40.
i) Traverse the array and display the elements
ii) Read a number from user and insert it to the array so that the array still remains in sorted order.

```java
package com.main;

import com.arrayops.ArrayOperations;

import java.util.Scanner;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        int[] array = {10, 20, 30, 40, 0, 0, 0, 0, 0, 0};
        int size = 4; // Initial size

        ArrayOperations arrayOps = new ArrayOperations();
        arrayOps.traverseArray(array, size);

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to insert in sorted order: ");
        int numberToInsert = scanner.nextInt();

        arrayOps.insertSorted(array, size, numberToInsert);
        size++; // Increase the size after insertion

        System.out.println("Array after insertion:");
        arrayOps.traverseArray(array, size);
    }
}
```

```java
package com.arrayops;

public class ArrayOperations {
    public void traverseArray(int[] array, int size) {
        System.out.println("Traversing the array:");
        for (int i = 0; i < size; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }

    public void insertSorted(int[] array, int size, int numberToInsert) {
        int i = size - 1;

        // Shift elements to the right to make room for the new element
        while (i >= 0 && array[i] > numberToInsert) {
            array[i + 1] = array[i];
            i--;
        }

        // Insert the new element
        array[i + 1] = numberToInsert;
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cda
c/DSA/DAY1$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jr
e/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/
d18520872ddb8dc5065b4e34520e7418/redhat.java/jdt_ws/DAY1_69323
fea/bin com.main.AppMain
Traversing the array:
10 20 30 40
Enter a number to insert in sorted order: 15
Array after insertion:
Traversing the array:
10 15 20 30 40
```

2. Write a Java class named ArrayDemo with
the following methods * data members
* an empty array of size 10 as data member of the class
Following methods
i) A method that accept a value from user and store it to the array at the last position
ii) A method to traverse the array and display all the elements
Call the methods from main method.

```java
package com.main;

import com.arrayops.ArrayDemo;

import java.util.Scanner;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        ArrayDemo arrayDemo = new ArrayDemo();

        Scanner scanner = new Scanner(System.in);

        for (int i = 0; i < 10; i++) {
            System.out.print("Enter a value to add to the array: ");
            int value = scanner.nextInt();
            arrayDemo.acceptValue(value);
        }

        arrayDemo.traverseArray();
    }
}
```

```java
package com.arrayops;

import java.util.Scanner;

public class ArrayDemo {
    private int[] array = new int[10];
    private int size = 0;

    public void acceptValue(int value) {
        if (size < array.length) {
            array[size] = value;
            size++;
            System.out.println("Value " + value + " added to the array.");
        } else {
            System.out.println("Array is full. Cannot add more values.");
        }
    }

    public void traverseArray() {
        System.out.println("Traversing the array:");
        for (int i = 0; i < size; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/Day1
2$  cd /media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/Day1\ 2 ; /usr/b
in/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.co
nfig/Code/User/workspaceStorage/4111a0cc4b1798c61039f446a7b73c39/redhat.j
ava/jdt_ws/Day1\ 2_e7736d7c/bin com.main.AppMain
Enter a value to add to the array: 25
Value 25 added to the array.
Enter a value to add to the array: 15
Value 15 added to the array.
Enter a value to add to the array: 35
Value 35 added to the array.
Enter a value to add to the array: 45
Value 45 added to the array.
Enter a value to add to the array: 55
Value 55 added to the array.
Enter a value to add to the array: 65
Value 65 added to the array.
Enter a value to add to the array: 75
Value 75 added to the array.
Enter a value to add to the array: 85
Value 85 added to the array.
Enter a value to add to the array: 95
Value 95 added to the array.
Enter a value to add to the array: 05
Value 5 added to the array.
Traversing the array:
25 15 35 45 55 65 75 85 95 5
```

3. Write a Java class named ArrayDemo with an array as data member and the following methods

a. A method that takes a position and a value as parameter and insert the value at the position

b. A method that takes a position as parameter and delete the element at the position

c. A method that takes a value as parameter and delete the value from the array

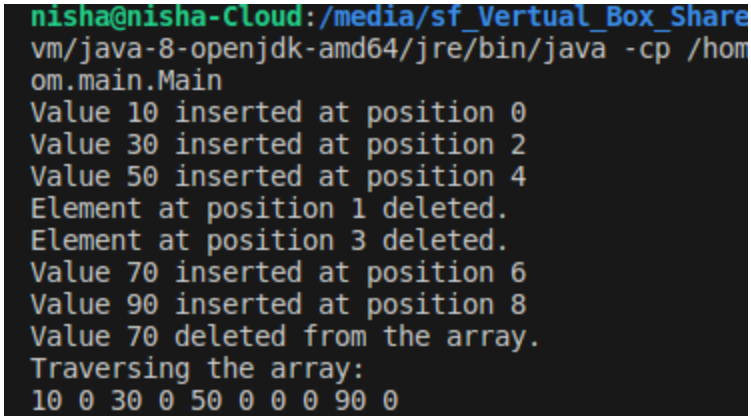
d. Traverse the array and display the elements

Create a class with the main method. Create an array inside the main method. Call the methods in ArrayDemo class from the main method

```java
package com.main;
import com.arrayops.ArrayDemo;

public class Main {
    Run|Debug
    public static void main(String[] args) {
        ArrayDemo arrayDemo = new ArrayDemo(size:10);

        // Insert values at specific positions
        arrayDemo.insertValue(position:0, value:10);
        arrayDemo.insertValue(position:2, value:30);
        arrayDemo.insertValue(position:4, value:50);

        // Delete elements at specific positions
        arrayDemo.deleteElement(position:1);
        arrayDemo.deleteElement(position:3);

        // Insert more values
        arrayDemo.insertValue(position:6, value:70);
        arrayDemo.insertValue(position:8, value:90);

        // Delete a specific value
        arrayDemo.deleteValue(value:70);

        // Display the array
        arrayDemo.traverseArray();
    }
}
```

```java
package com.arrayops;

public class ArrayDemo {
    private int[] array;

    public ArrayDemo(int size) {
        array = new int[size];
    }

    public void insertValue(int position, int value) {
        if (position < 0 || position >= array.length) {
            System.out.println("Invalid position. Element not inserted.");
        } else {
            array[position] = value;
            System.out.println("Value " + value + " inserted at position " + position);
        }
    }

    public void deleteElement(int position) {
        if (position < 0 || position >= array.length) {
            System.out.println("Invalid position. Element not deleted.");
        } else {
            array[position] = 0;
            System.out.println("Element at position " + position + " deleted.");
        }
    }

    public void deleteValue(int value) {
        for (int i = 0; i < array.length; i++) {
            if (array[i] == value) {
                array[i] = 0;
                System.out.println("Value " + value + " deleted from the array.");
            }
        }
    }
```

```java
    public void traverseArray() {
        System.out.println("Traversing the array:");
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }
}
```



4. Write a Java class StackDemo with methods for the following functionalities.
* Determine whether a stack is empty.
* Determine whether a stack is full.
* Push a new item to the stack. Before pushing method must check whether queue is full.

Remove (pop) from the stack the item that was added most recently and display it. Before removing method must check whether queue is empty
Use the above methods try the following from the main method
• Read n inputs from user and push to array
• Remove top k elements from the stack and display.
• Push few more elements to stack
• Remove all the items from the stack and display

```java
package com.main;

import com.stackops.StackDemo;
import java.util.Scanner;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the stack: ");
        int stackSize = scanner.nextInt();

        StackDemo stack = new StackDemo(stackSize);

        System.out.print("Enter the number of inputs (n): ");
        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            System.out.print("Enter input " + (i + 1) + ": ");
            int input = scanner.nextInt();
            stack.push(input);
        }

        System.out.print("Enter the number of elements to remove (k): ");
        int k = scanner.nextInt();

        for (int i = 0; i < k; i++) {
            Integer popped = stack.pop();
            if (popped != null) {
                System.out.println("Popped: " + popped);
            }
        }

        stack.push(element:100);
        stack.push(element:200);

        System.out.println("Elements left in the stack:");
        stack.displayStack();
    }
}
```

```java
package com.stackops;

public class StackDemo {
    private int arr[];
    private int top;
    private int maxSize;

    // Constructor to initialize the stack with a given size
    public StackDemo(int size) {
        arr = new int[size];
        top = -1; // Initialize the top pointer to -1 (empty stack)
        maxSize = size; // Store the maximum size of the stack
    }

    // Check whether the stack is empty
    public boolean isStackEmpty() {
        return top == -1;
    }

    // Check whether the stack is full
    public boolean isStackFull() {
        return top == maxSize - 1;
    }

    // Push a new item onto the stack
    public void push(int element) {
        if (isStackFull()) {
            System.out.println("Stack is full, cannot push " + element);
        } else {
            top++; // Increment the top pointer
            arr[top] = element; // Add the element to the stack
            System.out.println("Pushed " + element + " onto the stack.");
        }
    }
```

```
    // Pop (remove) the item from the top of the stack
    public Integer pop() {
        if (isStackEmpty()) {
            System.out.println("Stack is empty, cannot pop.");
            return null;
        } else {
            int item = arr[top]; // Get the item from the top of the stack
            top--; // Decrement the top pointer
            return item;
        }
    }

    // Display the elements in the stack
    public void displayStack() {
        if (isStackEmpty()) {
            System.out.println("Stack is empty.");
        } else {
            System.out.println("Elements in the stack:");
            for (int i = top; i >= 0; i--) {
                System.out.println(arr[i]);
            }
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/D
SA/Day14$  cd /media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/D
ay14 ; /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/jav
a -cp /home/nisha/.config/Code/User/workspaceStorage/e3d8a0cb06bc
8d08a3bd34fa579a4345/redhat.java/jdt_ws/Day14_bd24c18a/bin com.ma
in.AppMain
Enter the size of the stack: 5
Enter the number of inputs (n): 5
Enter input 1: 5
Pushed 5 onto the stack.
Enter input 2: 10
Pushed 10 onto the stack.
Enter input 3: 15
Pushed 15 onto the stack.
Enter input 4: 20
Pushed 20 onto the stack.
Enter input 5: 25
Pushed 25 onto the stack.
Enter the number of elements to remove (k): 2
Popped: 25
Popped: 20
Pushed 100 onto the stack.
Pushed 200 onto the stack.
Elements left in the stack:
Elements in the stack:
200
100
15
10
5
```

5.Write a Java class QueueDemo with methods for the following functionalities.

* Determine whether a Queue is empty.
* Determine whether a Queue is full.
* En-queue an item to the Queue . Before en-queuing check whether queue is full.
* De-queue an item from the queue . Before removing method must check whether queue is empty

Use the above methods try the following from the main method

• Read 5 inputs from user and do enqueue operation
• Remove 3 elements from the queue and display.
• En-queue few more elements
• De-Queue aII the items from the queue and display

```java
package com.main;

import com.queueopsdemo.QueueDemo;

import java.util.Scanner;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the queue: ");
        int queueSize = scanner.nextInt();

        QueueDemo queue = new QueueDemo(queueSize);

        System.out.println("Enter 5 inputs to enqueue:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Input " + (i + 1) + ": ");
            int input = scanner.nextInt();
            queue.enqueue(input);
        }

        System.out.print("Enter the number of elements to dequeue: ");
        int k = scanner.nextInt();

        for (int i = 0; i < k; i++) {
            Integer dequeued = queue.dequeue();
            if (dequeued != null) {
                System.out.println("Dequeued: " + dequeued);
            }
        }
```

```
        System.out.println("Enter more elements to enqueue:");
        while (!queue.isQueueFull()) {
            System.out.print("Input: ");
            int input = scanner.nextInt();
            queue.enqueue(input);
        }

        System.out.println("Dequeue all the items from the queue:");
        while (!queue.isQueueEmpty()) {
            Integer dequeued = queue.dequeue();
            if (dequeued != null) {
                System.out.println("Dequeued: " + dequeued);
            }
        }
    }
}
```

```java
package com.queueopsdemo;

public class QueueDemo {
    private int arr[];
    private int front;
    private int rear;
    private int maxSize;

    public QueueDemo(int size) {
        arr = new int[size];
        front = 0;
        rear = -1;
        maxSize = size;
    }

    // Determine whether the queue is empty
    public boolean isQueueEmpty() {
        return rear < front;
    }

    // Determine whether the queue is full
    public boolean isQueueFull() {
        return rear == maxSize - 1;
    }

    // Enqueue an item to the queue
    public void enqueue(int element) {
        if (isQueueFull()) {
            System.out.println("Queue is full, cannot enqueue " + element);
        } else {
            arr[++rear] = element;
            System.out.println("Enqueued " + element + " into the queue.");
        }
    }
```

```java
    // Dequeue an item from the queue
    public Integer dequeue() {
        if (isQueueEmpty()) {
            System.out.println("Queue is empty, cannot dequeue.");
            return null;
        } else {
            int item = arr[front++];
            return item;
        }
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/DAy15$  /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/618d5c5c38a1b9c8ed34bd6908d03308/redhat.java/jdt_ws/DAy15_bd1635ab/bin com.main.AppMain
Enter the size of the queue: 5
Enter 5 inputs to enqueue:
Input 1: 10
Enqueued 10 into the queue.
Input 2: 12
Enqueued 12 into the queue.
Input 3: 14
Enqueued 14 into the queue.
Input 4: 16
Enqueued 16 into the queue.
Input 5: 18
Enqueued 18 into the queue.
Enter the number of elements to dequeue: 3
Dequeued: 10
Dequeued: 12
Dequeued: 14
Enter more elements to enqueue:
Dequeue all the items from the queue:
Dequeued: 16
Dequeued: 18
```