Arrays - Assignment

Task 1: Array Sorting and Searching

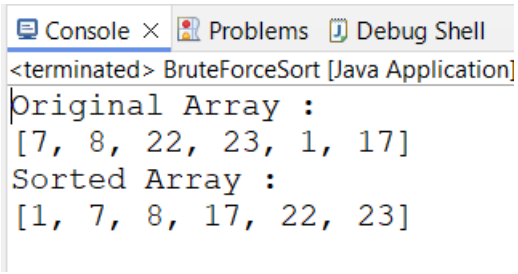
a) Implement a function called BruteForceSort that sorts an array using the brute force approach. Use this function to sort an array created with InitializeArray.

Code:

```
package arraysortingandsearching;
import java.util.Arrays;
import java.util.Random;

public class BruteForceSort {

public static int[] initializeArray(int size, int minValue, int maxValue) {
int[] arr = new int[size];
Random random = new Random();
for(int i=0; i<size; i++) {
arr[i] = random.nextInt((maxValue - minValue)+1);
}
return arr;
}

public static void bruteForceSorting(int[] arr) {
int n = arr.length;
for(int i=0; i<n-1; i++) {
int minIndex = i;
for(int j=i+1; j<n; j++) {
if(arr[j] < arr[minIndex]) {
minIndex = j;
}
}
int temp = arr[minIndex];
```

```
arr[minIndex] = arr[i];
arr[i] = temp;
}
}

public static void main(String[] args) {
int[] arr = initializeArray(6, 0, 25);
System.out.println("Original Array : ");
System.out.println(Arrays.toString(arr));
bruteForceSorting(arr);
System.out.println("Sorted Array : ");
System.out.println(Arrays.toString(arr));
}
}
```

Output:



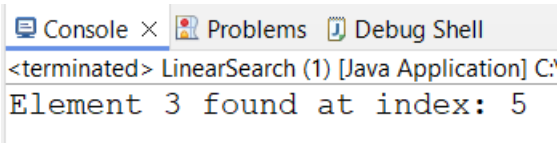
b) Write a function named PerformLinearSearch that searches for a specific element in an array and returns the index of the element if found or -1 if not found.

Code:

```
package arraysortingandsearching;
public class LinearSearch {

public static int performLinearSearch(int[] arr, int target) {
for(int i=0; i<arr.length; i++) {
if(arr[i] == target) {
```

```
return i;
}
}
return -1;
}

public static void main(String[] args) {
int[] arr = {5, 9, 2, 4, 8, 3, 1, 7, 6};
int target = 3;

int index = performLinearSearch(arr,target);
if(index != -1) {
System.out.println("Element " + target + " found at index: "+ index);
}
else {
System.out.println("Element "+ target + " not found");
}
}
}
```

Output:



Task 2: Two-Sum Problem

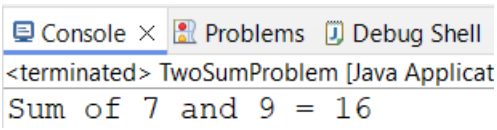
a) Given an array of integers, write a program that finds if there are two numbers that add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice. Optimize the solution for time complexity.

Code:

```
package twosumproblem;
import java.util.Arrays;
```

```
public class TwoSumProblem {
public static void main(String[] args) {
int[] arr = {5, 9, 2, 4, 8, 3, 1, 7, 6};
int target = 16;
boolean isFound = false;
Arrays.sort(arr);
int l = 0;
int h = arr.length -1;
while(l < h) {
if(arr[l] + arr[h] == target) {
isFound = true;
break;
}
else if(arr[l] + arr[h] < target) {
l++;
}
else {
h--;
}
}
if(isFound) {
System.out.println("Sum of "+ arr[l] + " and " + arr[h] + " = "+ target);
}
else {
System.out.println("Elements not found");
}
}
}
```

Output:



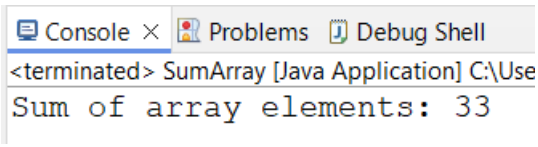
Task 3: Understanding Functions through Arrays

a) Write a recursive function named SumArray that calculates and returns the sum of elements in an array, demonstarte with example.

Code:

```
package sumarray;

public class SumArray {

public static int sumArray(int[] arr, int n) {
if(n==1) {
return arr[0];
}
return arr[n-1] + sumArray(arr, n-1);
}

public static void main(String[] args) {
int[] arr = {5, 9, 2, 3, 1, 7, 6};
int sum = sumArray(arr,arr.length);
System.out.println("Sum of array elements: " + sum);
}
}
```

Output:



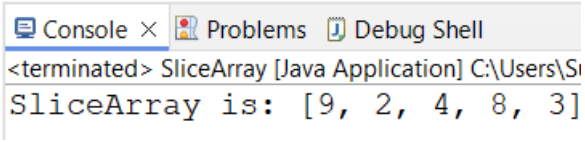
Task 4: Advanced Array Operations

a) Implement a method SliceArray that takes an array, a starting index, and an end index, then returns a new array containing the elements from the start to the end index.

Code:

```
package advancearrayoperations;
import java.util.Arrays;

public class SliceArray {

public static int[] sliceArray(int[] arr, int startIn, int endIn) {
int sliceArrLength = endIn - startIn + 1;
int[] sliceArr = new int[sliceArrLength];
for(int i=0; i<sliceArrLength; i++) {
sliceArr[i] = arr[startIn + i];
}
return sliceArr;
}

public static void main(String[] args) {
int[] arr = {5, 9, 2, 4, 8, 3, 1, 7, 6};
int startIn = 1;
int endIn = 5;

int[] slicedArr = sliceArray(arr,startIn,endIn);
System.out.println("SliceArray is: " + Arrays.toString(slicedArr));
}
}
```

Output:



b) Create a recursive function to find the nth element of a Fibonacci sequence and store the first n elements in an array.

Code:

```
package advancearrayoperations;
import java.util.Arrays;

public class Fibonacci {
public static int fibonacci(int n) {
if(n<=1) {
return n;
}
return fibonacci(n-1) + fibonacci(n-2);
}

public static void fibonacciArray(int[] arr, int n, int index) {
if(n<=0) {
return;
}
arr[index] = fibonacci(index);
fibonacciArray(arr, n-1, index+1);
}
public static void main(String[] args) {
int n =13;
int[] arr = new int[n];
fibonacciArray(arr,n,0);
System.out.println("Fibonacci Sequence: ");
System.out.println(Arrays.toString(arr));
int nthEle = arr[n-1];
System.out.println(n+"th Fibonacci number is: "+ nthEle);
}
}
```

Output:

```
Fibonacci Sequence:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
13th Fibonacci number is: 144
```