1.A

**public** **class** SelectionSort {

**public** **static** **void** main(String[] args) {

**int**[] a= {38,52,9,18,6,62,13,0};

**int** temp,min;

**for**(**int** i=0;i<a.length;i++)

{

min=i;

**for**(**int** j=0+i;j<a.length;j++)

{

**if**(a[j]<a[min])

{

temp=a[j];

a[j]=a[min];

a[min]=temp;

}

}

}

**for**(**int** i=0;i<a.length;i++)

{

System.***out***.print(a[i] + " ");

}

}

}

1.B

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[] array = {4, 8, 15, 16, 23, 42};

int target = 16;

int index = PerformLinearSearch(array, target);

if (index != -1) {

System.out.println("Element " + target + " found at index " + index);

} else {

System.out.println("Element " + target + " not found in the array.");

}

}

}

2.A

import java.util.HashMap;

public class TwoSum {

public static int[] twoSum(int[] nums, int target) {

HashMap<Integer, Integer> map = new HashMap<>();

for (int i = 0; i < nums.length; i++) {

int complement = target - nums[i];

if (map.containsKey(complement)) {

return new int[]{map.get(complement), i};

}

map.put(nums[i], i);

}

return new int[]{-1, -1};

}

public static void main(String[] args) {

int[] nums = {2, 7, 11, 15};

int target = 9;

int[] result = twoSum(nums, target);

if (result[0] != -1) {

System.out.println("Indices: [" + result[0] + ", " + result[1] + "]");

System.out.println("Numbers: [" + nums[result[0]] + ", " + nums[result[1]] + "]");

} else {

System.out.println("No valid solution found.");

}

}

}

3.A

public class Main {

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5};

int sum = SumArray(array, 0);

System.out.println("Sum of elements in the array: " + sum);

}

public static int SumArray(int[] arr, int n) {

if (n <= 0) {

return 0;

}

return arr[n - 1] + SumArray(arr, n - 1);

}

}

4.A

public static int[] SliceArray(int[] arr, int startIndex, int endIndex) {

if (startIndex < 0 || endIndex >= arr.length || startIndex > endIndex) {

throw new IllegalArgumentException("Invalid start or end index");

}

int[] slicedArray = new int[endIndex - startIndex + 1];

for (int i = startIndex; i <= endIndex; i++) {

slicedArray[i - startIndex] = arr[i];

}

return slicedArray;

}

4.B

public static int[] fibonacci(int n) {

int[] fibArray = new int[n];

return fibonacciHelper(n, fibArray, 0);

}

private static int[] fibonacciHelper(int n, int[] fibArray, int index) {

if (index == n) {

return fibArray;

}

if (index == 0) {

fibArray[index] = 0;

} else if (index == 1) {

fibArray[index] = 1;

} else {

fibArray[index] = fibArray[index - 1] + fibArray[index - 2];

}

return fibonacciHelper(n, fibArray, index + 1);

}