Problems
Algorithms, Corazza

Write Java programs that solve these problems as efficiently as possible. Be ready to explain why you believe each of your solutions is maximally efficient. For these problems, how can you tell whether your solutions are correct? How can you tell if your solution is more efficient than another person's solution? How can you express your solution in a way that could be implemented in most other programming languages, so that you are not dependent on Java?

Problem 1. Given an array `arr` of distinct integers and an integer z, determine whether `arr` contains two distinct numbers x and y so that $x + y = z$. Solve the problem by implementing the following Java method.

```
public static boolean sumFound(int[] arr, int z) {
   if(arr==null || arr.length < 2) {
      return false;
   }
   //implement
}
```

This method returns `true` if the input array contains two numbers whose sum is the input integer z, `false` otherwise.

Examples

(A) If input is $[1, 4, 2, 3], 5$, output should be `true` since $2 + 3 = 5$.

(B) If input is $[3, 3, 4, 7], 6$, output should be `false`; although $3 + 3 = 6$, it is not true that $3, 3$ are distinct.

(C) If input is $[1], 2$, output should be `false`; although $1 + 1 = 2$, the 1s are not distinct.

Problem 2. Implement the following Java method.

```
public static int secondSmallest(int[] arr) {
   if(arr==null || arr.length < 2) {
      throw new IllegalArgumentException("Input array too small");
   }
   //implement
}
```

This method returns the second smallest element of the input array.

Examples

(A) If input is $[1, 4, 2, 3]$, output should be 2.

(B) If input is $[3, 3, 4, 7]$, output should be 3. (Smallest is 3, and second smallest is 3.)

(C) If input is $[9]$, your program will throw an exception.

Problem 3. Given a list $L = [s_0, s_1, \ldots, s_{n-1}]$ of n distinct positive integers and a non-negative integer k, determine whether there is a subset of L the sum of whose values is k. Do this by implementing the following Java method.

```
public static boolean sumFound(List list, int k) {
   //implement
}
```

Hint. Use a solution to Problem 6 as part of your solution to this problem. A solution in the form of a jar file is provided for you so that you can try using it here even if you do not have your own solution to Problem 6.

Examples

(A) If input is $[1, 3, 9, 4, 8, 5]$ and $k = 21$, return `true` (since $9 + 4 + 8 = 21$)

(B) If input is $[1, 3, 9, 4, 8, 5]$ and $k = 22$, return `true` (since $1 + 9 + 4 + 8 = 22$)

(C) If input is $[1, 3, 9, 4, 8, 5]$ and $k = 31$, return `false` (since even the sum of *all* elements of L is less than 31)

(D) If input is $[1, 3, 9, 4, 8, 5]$ and $k = 0$, return `true` (since the sum of the *empty subarray* is 0)

Problem 4. *Sorting.* Do not use Java library sorting routines (since we need to be able to analyze our own solutions).

(A) You are given an ArrayList of Integer objects. Write a program that efficiently sorts the list; use the following method signature:

```
public static void sort(ArrayList<Integer> list){
   //implement
}
```

Note that sorting is to be done *in-place*; this means that the original input list is modified so that it becomes sorted.

(B) You are given a linked list containing Integer objects. Write a program that efficiently sorts the list; use the following method signature.

```
public static void sort(LinkedList<Integer> list){
   //implement
}
```

Problem 5. *Searching.* Given an array `arr` of ints and an int z.

(A) Determine whether z is in `arr`.

(B) Suppose the array `arr` is already known to be in sorted order. Can you write a more efficient program for determining whether z is in `arr`? If yes, why do you think your solution to (B) is faster?

Use the following method signature:

```
public static boolean find(int[] arr, int z) {
   \implement
}
```

Examples

(A) If input is $[2, 8, 3, 4], 3$, output should be `true`.

(B) If input is $[2, 8, 3, 4], 5$, output should be `false`.

(C) If input is $[2, 3, 4, 8], 3$, output should be `true`.

(D) If input is $[2, 3, 4, 8], 5$, output should be `false`.

Problem 6. Given a List L without duplicate elements, return another list that consists of all subsets of L (without duplicates). Use the following method signature:

```
public static List<Set> powerSet(List list) {
   //implement
}
```

Examples

(A) If input is $\{1\}$, output should be $[\{\}, \{1\}]$.

(B) If input is $\{1, 2\}$, output should be $[\{\}, \{1\}, \{2\}, \{1, 2\}]$.

Problem 7. Write a Java method that generates the following sequence of numbers:

$$a = \langle 0, 1, 1, 2, 3, 5, 8, \ldots \rangle.$$

The pattern is generated by the following formula:

$$a_0 = 0; \;\; a_1 = 1 \;\; a_n = a_{n-1} + a_{n-2}.$$

Do this by implementing the following method:

```
public static int generate(int n){
   if(n == 0) return 0;
   if(n == 1) return 1;
   //implement
}
```

Test your solution inside a `main` method:

```
for(int i = 0; i < 20; ++i) {
   System.out.println(generate(i));
}
```

Your output should be $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765$.

Problem 8. Given two positive integers x, y, find the smallest positive integer z with the property that both x and y are factors of z. Do this by implementing the following Java method.

```
public static int smallestCommon(int x, int y) {
   //implement
}
```

Examples

(A) If input is $4, 6$, return 12.

(B) If input is $3, 5$, return 15.

(C) If input is $7, 14$, return 14.