

Programming Problems

22nd Annual Computer Science Programming Contest

Department of Mathematics and Computer Science
Western Carolina University

5 April 2011

Problem One: Add Times

Represent a time by the format

days hours minutes

using a 24 hour clock. So for example,

3 15 45

means

3 days 15 hours 45 minutes

You are to write a Java program that reads from the user two times in this format and then output their sum written in correct format.

Example Run

```
Please enter the first time: 3 15 45
Please enter the second time: 2 12 35
```

```
The sum of their times is: 6 4 20
```

This is since

```
  3 days 15 hours 45 minutes
+  2 days 12 hours 35 mintues
```

equals

```
  6 days 4 hours 20 minutes
```

Problem Two: Air Force

Consider an `AirForce` class that models a nation's air force.

It has two fields:

1. an `int` field, that represents the number of fighters in the Air Force and
2. an `int` field that represents the number of bombers in the Air Force.

The constructors and methods are:

1. It has a default constructor that initializes the two `int` fields to 0.
2. It has a `buyFighters` method that takes an `int` value as a parameter and adds it to the current number of fighters for that Air Force and does not return anything.
3. It has a `buyBombers` method that does the same thing for the bombers field.
4. It has an `attacks` method that does not return a value. It pits one `AirForce` object against another. It has one parameter, which is a reference to another `AirForce` object.

The current object first simulates sending its fighters against the enemy object as follows:

- (a) If the current object has more fighters than the enemy object (or the same number of fighters), the method reduces the number of fighters of the enemy object by half (for instance, if the current object has 12 fighters and the enemy object has 11, the enemy object's fighter count is reduced to 5).
- (b) If the current object has fewer fighters than the enemy object, reduce the current object's fighter count by half.

The current object then simulates sending its bombers against the enemy object as follows:

- (a) If the current object has more bombers than the enemy object, reduce the enemy object's bomber count to 0.
 - (b) If the bomber count for the current object is less than or the same as the enemy object's, reduce the current object's bomber count to 0 (for instance, if the current object has 10 bombers and the enemy object has 5 bombers, reduce the enemy object's bomber count to 0).
5. It has a method called `defeats` that has one parameter which is a reference to another `AirForce` object and returns either `true` or `false`. If the total number of fighters and bombers of the current object is greater than or equal to the total number of fighters and bombers of the other object, `defeats` returns `true`. Otherwise, return `false`.
 6. It has a `toString` method, which returns a `String` describing the current state (number of fighters and bombers) of the object.

Write an implementation of the `AirForce` class. Use the following driver to test your class

```
class AFDriver
{
    public static void main(String[] args)
    {
        AirForce usa = new AirForce();
    }
}
```

```
AirForce badGuy = new AirForce();
usa.buyFighters(10);
badGuy.buyFighters(12);
usa.buyBombers(10);
badGuy.buyBombers(10);
badGuy.attacks(usa);
System.out.println("USA: " + usa.toString());
System.out.println("BadGuy: " + badGuy.toString());
if (usa.defeats(badGuy))
    System.out.println("USA wins");
else
    System.out.println("BadGuy wins");
}
}
```

Output should be:

```
USA: # of fighters is 5; # of bombers is 10
BadGuy: # of fighters is 12; # of bombers is 0
USA wins
```

Problem Three: Index is Value?

Write a Java program that includes a method that when passed a sorted array of distinct integers (that is $A_1 < A_2 < \dots < A_N$), determines using an *efficient* algorithm if there exists an integer i such that $A[i] == i$ and returns true if such an integer exists. In other words, return true if for some array element, the value of the array element is the same as the index of the array element.

The method must have the following header.

```
public boolean indexIsValueExists(int[] array)
```

One approach that is not efficient is the brute force algorithm of looking at every array element and comparing its value with its index until such an array element is found or the end of the array is reached.

However, there is a better, more efficient, approach that only has to examine a few of the array elements in order to determine whether such an array element exists. Your program is to use this more efficient approach. A hint is that you should take advantage of the fact that the array is sorted.

```
// array element 2 has value 2
```

```
Example Run:
```

```
Enter the length of the array: 3
```

```
Enter the integers in the array: -1 0 2
```

```
Array element with index as value? true
```

```
// no array element has its value as its index
```

```
Example Run:
```

```
Enter the length of the array: 3
```

```
Enter the integers in the array: -5 2 6
```

```
Array element with index as value? false
```

Problem Four: Fast Intersection

Let $A = a_0, \dots, a_{n-1}$ and $B = b_0, \dots, b_{n-1}$ be two arrays of integers with each array having no duplicate integers. Consider the problem of finding their intersection, i.e., the set C of all the integers that are in both A and B . One approach, the brute force solution, is to compare every integer in one set with every integer in the other set. However that solution has poor performance since there are so many comparisons.

A better approach is to sort both arrays first and then ... do what?

Your program must develop this better solution that uses presorting and involves many fewer comparisons than the brute force solution. You can use the following code to sort your arrays first

```
import java.util.Arrays;

...
Arrays.sort(someArray);
```

where `someArray` is the name of an array you are sorting.

Example Run:

```
Please enter the length of the first array: 6
Please enter the first array: 2 4 5 3 7 1
Please enter the length of the second array: 5
Please enter the second array: 8 2 3 11 6
```

```
The intersection is: 2 3
```

Example Run:

```
Please enter the length of the first array: 6
Please enter the first array: 2 4 5 3 7 1
Please enter the length of the second array: 5
Please enter the second array: 8 9 10 11 6
```

```
The intersection is: empty
```

You can use `ArrayLists` instead of arrays if you prefer. In that case the code you use is

```
import java.util.Collections;

...
Collections.sort(someArrayList);
```