Lab7

Part 1: Jacobsthal number are an integer sequence named after Ernst Jacobsthal. The sequence starts at 0 and 1, then each following number is found by adding the number before it to twice the number before that.

$$J_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ J_{n-1} + 2J_{n-2} & \text{if } n > 1. \end{cases}$$

The sequence is defined as:
The first few numbers in the sequence are:
0, 1, 1, 3, 5, 11, 21, 43, 85, 171, 341, 683, 1365, 2731, …

Write a Java program that contains the following three functions:

1. long Jacobsthal_recursive(int n);
2. long Jacobsthal_iterative(int n);

The output should be in the following format:

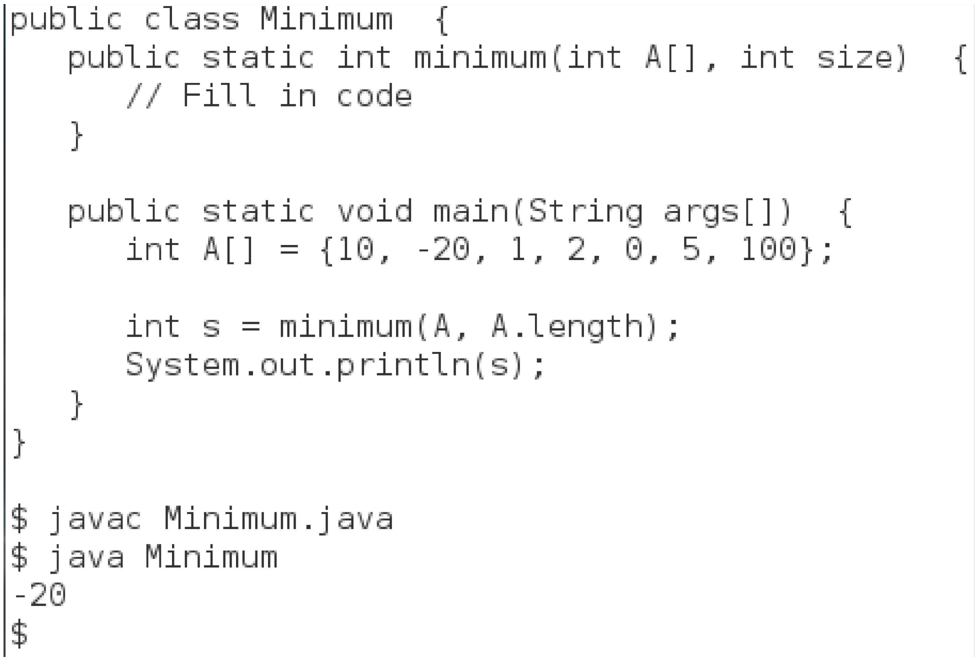
```
$ java Jacobsthal 10
Recursive version: 0, 1, 1, 3, 5, 11, 21, 43, 85, 171

Time taken to execute recursive version: XX.XX msec

Iterative version: 0, 1, 1, 3, 5, 11, 21, 43, 85, 171

Time taken to execute iterative version: XX.XX msec
```

(a) Find out the argument *x* that maximises the Jacobsthal number that can be printed **before**
overflowing. In other words, find where *f(x)* is the Jacobsthal function.

$$\arg\max_{x} f(x)$$

(b) What is the Jacobsthal number at the argument *x*?

Part2: Write a recursive method that returns the smallest value in the first *size* elements of an array. The signature of the method is:
int minimum(int A[], int size)
Here is some framework to get you started:



```
public class Minimum  {
   public static int minimum(int A[], int size)  {
      // Fill in code
   }

   public static void main(String args[])  {
      int A[] = {10, -20, 1, 2, 0, 5, 100};

      int s = minimum(A, A.length);
      System.out.println(s);
   }
}

$ javac Minimum.java
$ java Minimum
-20
$
```

Part3: Complete code in CS401BinaryTree.java file