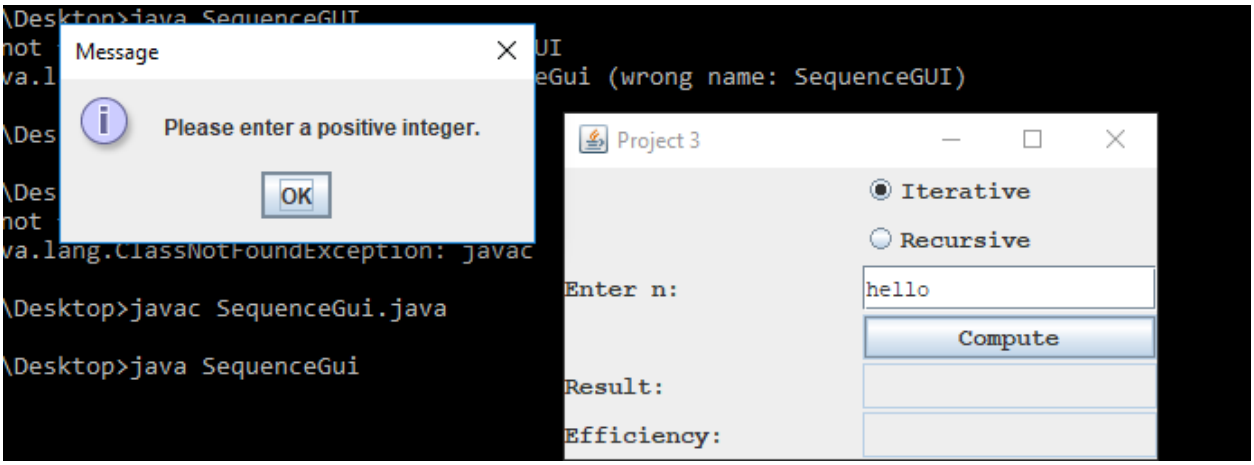
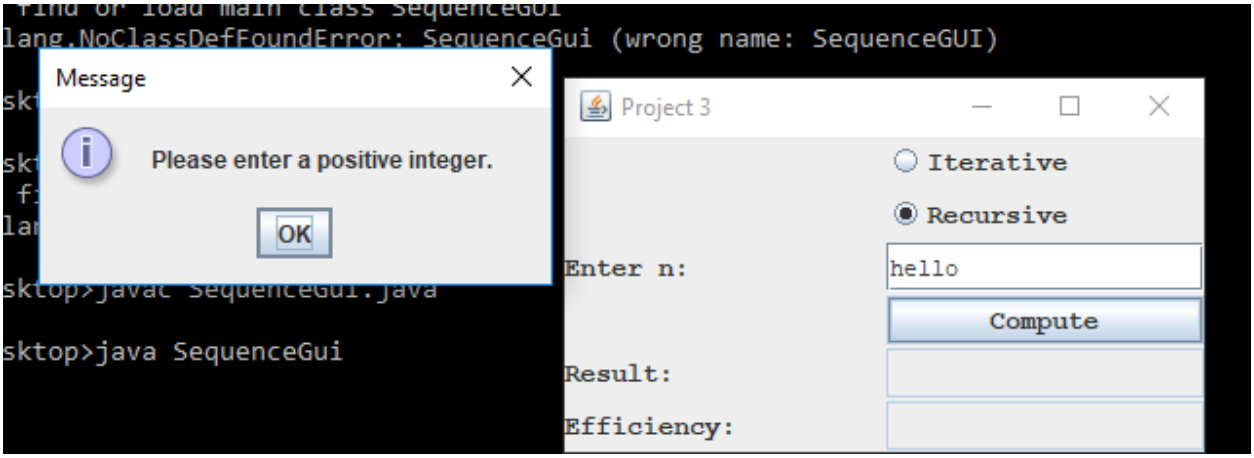
Project 3 – CMIS 242

Input	Expected Output	Actual Output	Pass?
1. Iterative, hello	Please enter a positive integer.	Please enter a positive integer.	Yes
2. Recursive, hello	Please enter a positive integer.	Please enter a positive integer.	Yes
3. Iterative, 0	Result: 0 Efficiency: 0	Result: 0 Efficiency: 0	Yes
4. Recursive, 0	Result: 0 Efficiency: 1	Result: 0 Efficiency: 1	Yes
5. Iterative, -1	Result: -1 Efficiency: -1	Result: -1 Efficiency: -1	Yes
6. Recursive, -1	Result: -1 Efficiency: -1	Result: -1 Efficiency: -1	Yes
7. Iterative, 2	Result: 2 Efficiency: 2	Result: 2 Efficiency: 2	Yes
8. Iterative, 3	Result: 5 Efficiency: 3	Result: 5 Efficiency: 3	Yes
9. Iterative, 5	Result: 29 Efficiency: 5	Result: 29 Efficiency: 5	Yes
10. Recursive, 2	Result: 2 Efficiency: 3	Result: 2 Efficiency: 3	Yes
11. Recursive, 3	Result: 5 Efficiency: 5	Result: 5 Efficiency: 5	Yes
12. Recursive, 5	Result: 29 Efficiency: 15	Result: 29 Efficiency: 15	Yes
13. Close program	Writing values for n = 0 to sequence_output.csv	Writing values for n = 0 to sequence_output.csv	Yes

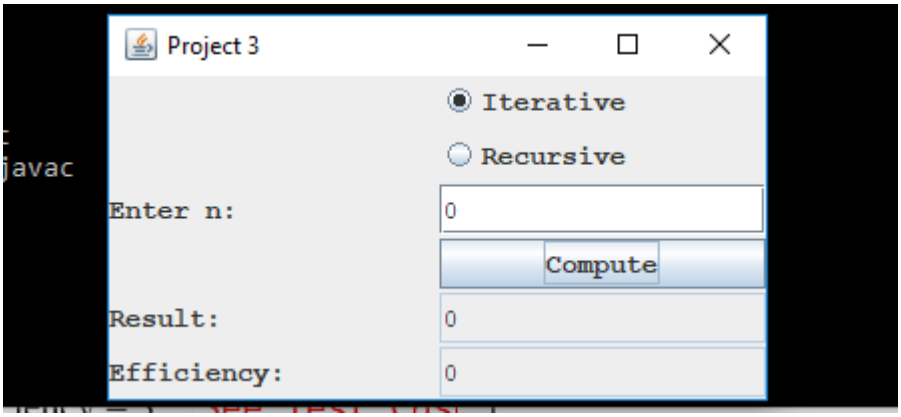
Test Screenshot 1:



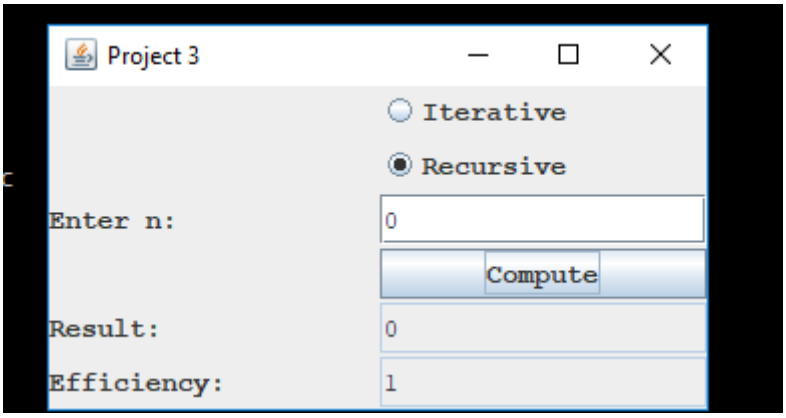
Test Screenshot 2:



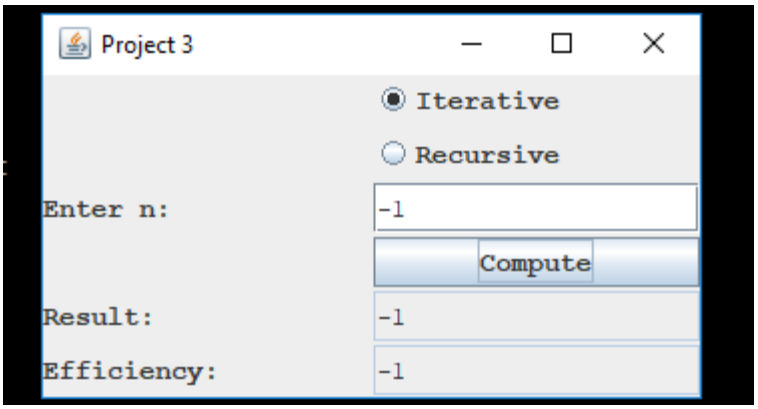
Test Screenshot 3:



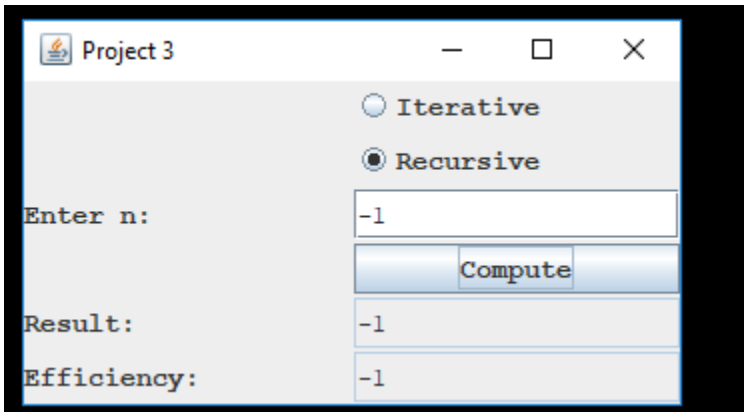
Test Screenshot 4:



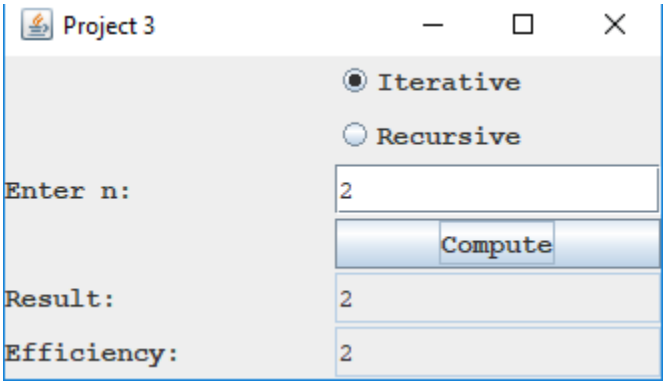
Test Screenshot 5:



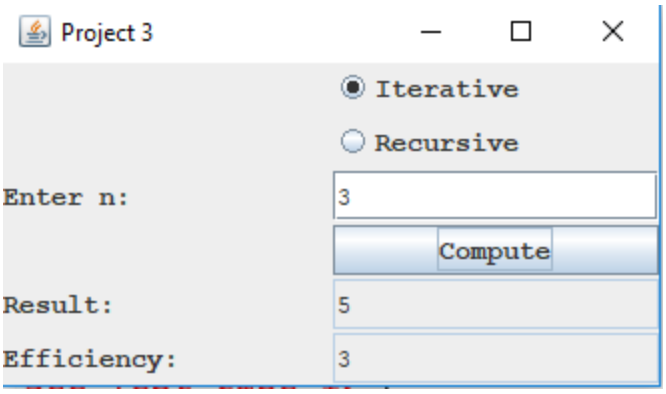
Test Screenshot 6:



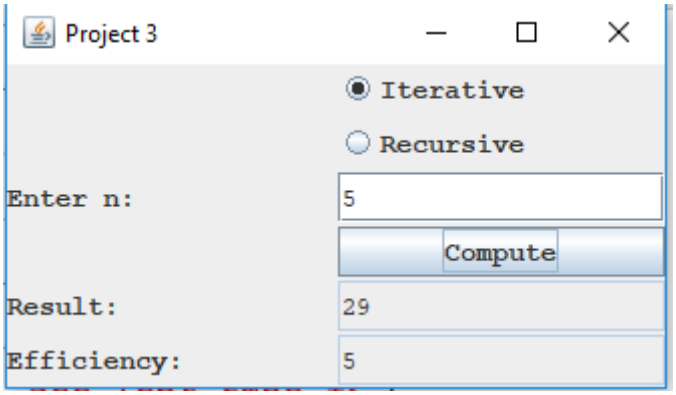
Test Screenshot 7:



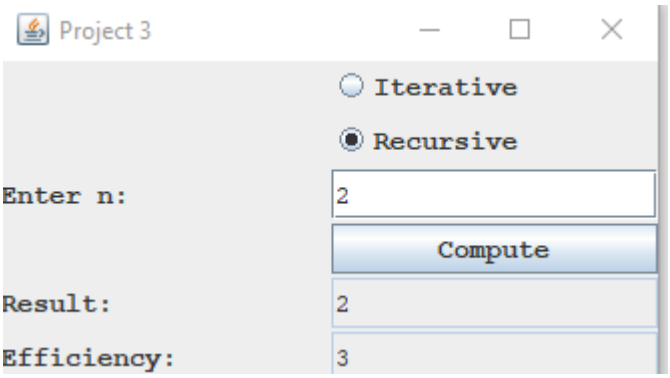
Test Screenshot 8:



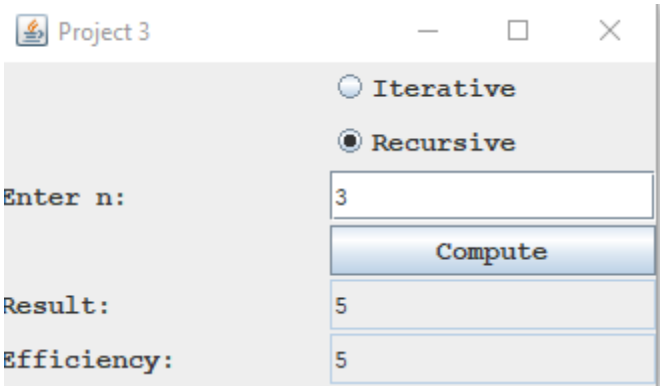
Test Screenshot 9:



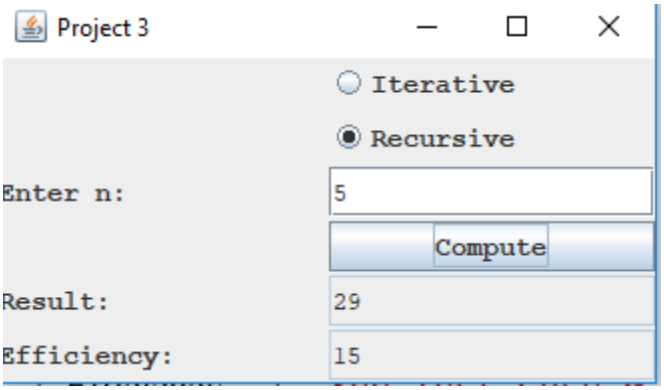
Test Screenshot 10:



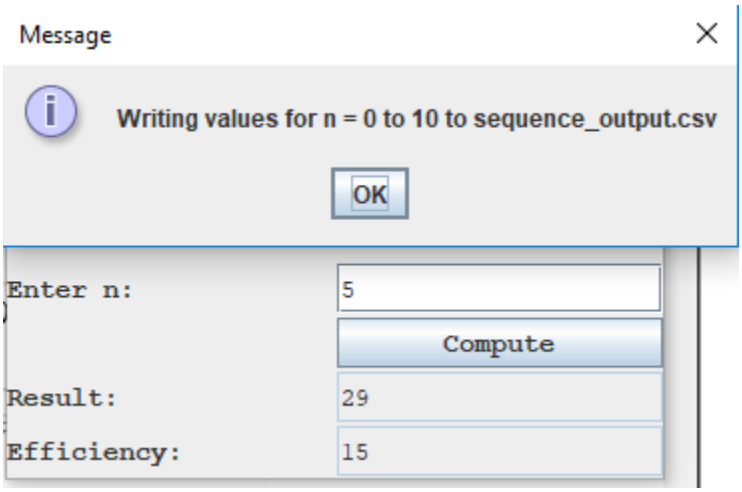
Test Screenshot 11:



Test Screenshot 12:



Test Screenshot 13:



Efficiency Chart

The below chart compares the efficiency of the iterative and recursive functions for values of n between 0 and 10. The iterative efficiency is measured by the number of times the loop in the method iterates. In the case of n=0, the loop in the iterative method is not run at all. The iterative loop will always run n times.

The recursive efficiency is measured by the number of times the recursive method is called. In the base cases of n = 0 and n = 1, the recursive method only gets called once. As n increases, additional calls to the recursive method will be made as each value of n gets broken down to the base cases.

For example, when n=2 the efficiency is 3: one call is made for n=2, which then makes 2 calls for n=1 and n=0 (1 + 1 + 1). When n=3 the efficiency is 5 because the method is called once for the n value of 3 then you have the efficiency of n=2 plus the efficiency of n=1, i.e. 1 + e(n=2) + e(n=1). Similarly, the efficiency of :

n=4 is 9: 1 + e(n=3) + e(n=2) = 1 + 5 + 3
n=5 is 15: 1 + e(n=4) + e(n=3) = 1 + 9 + 5