CSC 1012 Programming Assignment #1 Summer’19

points deducted for late submission.

Using Python, write the program for the following.

You must submit a copy of the source code, and the output, on Black Board (make sure you have several runs for the output). All pages must be stapled, with your name, course number and project number.

*Problem Statement:*

Douglas Hofstadter’s Pulitzer-prize-winning book Gödel, Escher, Bach

contains many interesting mathematical puzzles, many of which can be expressed in the form of computer programs.

Hofstadter mentions a wonderful problem that is well within the scope of the control statements. The problem can be expressed as follows:

Pick some positive integer and call it n.

If n is even, divide it by two.

If n is odd, multiply it by three and add one.

Continue this process until n is equal to on

On page 401 of the Vintage edition, Hofstadter illustrates this process with the following example, starting with the number 15:

15 is odd, so I make 3n + 1: 46

46 is even, so I take half: 23

23 is odd, so I make 3n+1: 70

70 is even, so I take half: 35

35 is odd, so I make 3n + 1: 106

106 is even, so I take half: 53

53 is odd, so I make 3n+1: 160

160 is even, so I take half: 80

80 is even, so I take half: 40

40 is even, so I take half: 20

20 is even, so I take half: 10

10 is even, so I take half: 5

5 is odd, so I make 3n+1: 16

16 is even, so I take half: 8

8 is even, so I take half: 4

4 is even, so I take half: 2

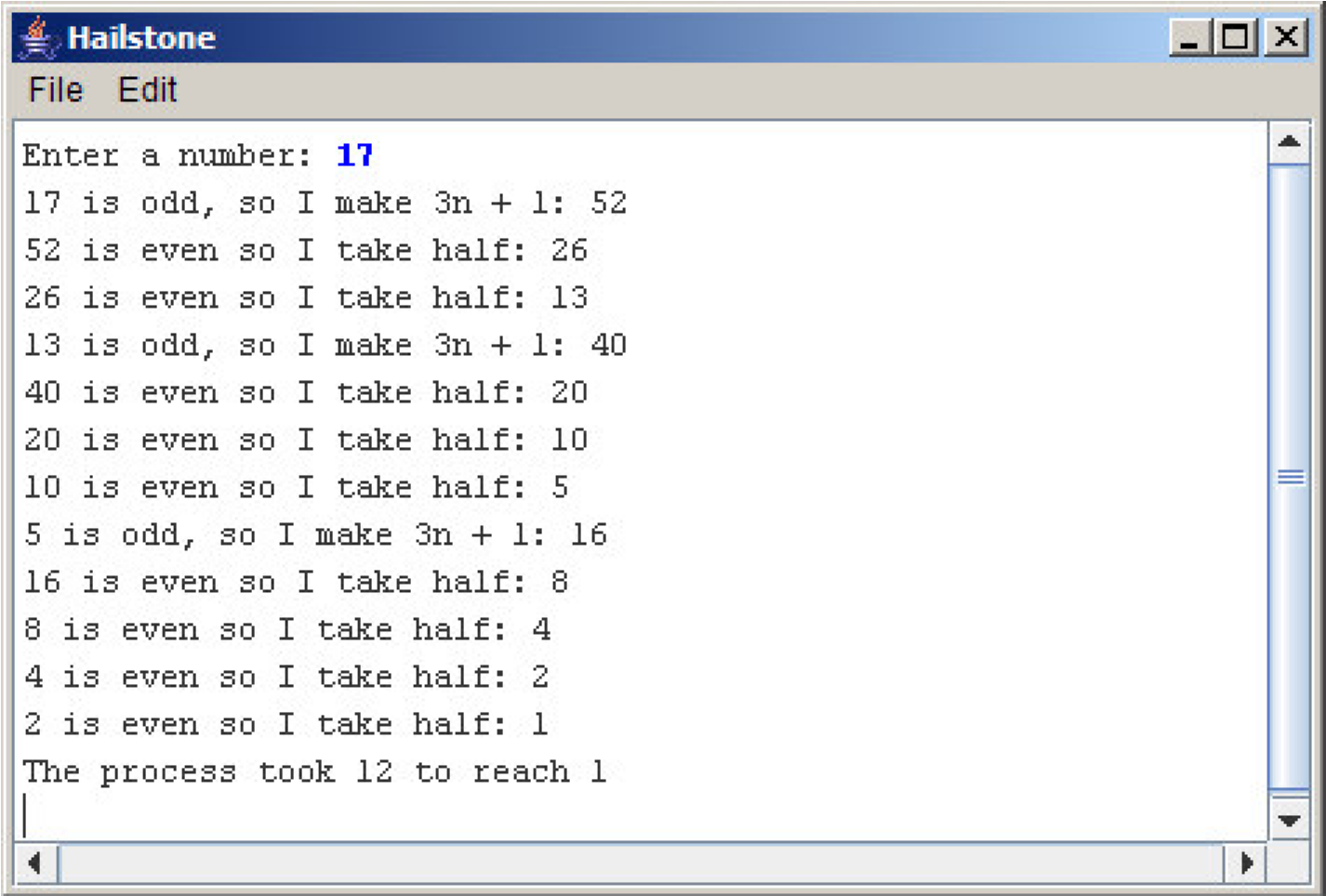
2 is even, so I take half: 1

As you can see from this example, the numbers go up and down, but eventually—at least for all numbers that have ever been tried—comes down to end in 1. In some respects, this process is reminiscent of the formation of hailstones, which get carried upward by the winds over and over again before they finally descend to the ground.

Because of this analogy, this sequence of numbers is usually called the **Hailstone sequence**, although it goes by many other names as well.

Write a ***recursive*** **ConsoleProgram** and/or a GUI interface, that reads in a number from the user and then displays the Hailstone sequence for that number, just as in Hofstadter’s book, followed by a line showing the number of steps taken to reach 1.

For example, your program should be able to produce a sample run that looks like this:



The fascinating thing about this problem is that no one has yet been able to prove that it always stops. The number of steps in the process can certainly get very large. How many steps, for example, does your program take when n is 27

The program output must show all steps performed and must have several runs, try an input with 15 as the first run and another input with 25 , etc..