PROBLEM I.8: PRIMALITY FUNCTION

ROSIE KEY

1. Prime Check Function

The `prime check` function takes an input `natNum` (technically, an integer, but it's assumed to be a natural number) and determines if it is a prime number. If it is a prime number, then the function returns "True". If it's composite, it returns "False".

The first thing the function does is it creates a variable called `flagVar` that is equal to zero. The purpose of this variable is to be used in a later if statement that determines if the statement "the input is prime" is true or not. Then, a for loop runs with terms `ii` ranging from one to `natNum` minus one, looking at each natural number up to the input. An if statement within the for loop determines whether or not the input is divisible by the `ii`th term with the exception of `ii` equal to one. If the loop detects a number besides one that `natNum` is divisible by, the value of `flagVar` is changed to one to indicate that the number is composite. Finally, the previously mentioned if statement runs, and depending on the value of `flagVar`, "True" or "False" is returned.

2. Finding the Nth Term

For finding the nth term of a sequence of prime numbers, an input called `n` and an empty list called `nList` is created to store the prime numbers. Then, a variable called `count` is created and is initally equal to two. The reason for this is that count represents the first term in the prime sequence, and plugging one or zero into the `prime check` will return an error due to those inputs simply being out of range for the definition's for loop. While the inputted length for the prime sequence is known, the upper boundaries of the list aren't since it requires more than indices increasing at a constant rate. Because of this, a while loop was used under the condition that the length of `nList` is less than `n`. Within the while loop, an if statement checks if `count` is prime using the `prime check` function. If `count` is prime, then it is added to `nList`. Outside the if loop, `count` increasing by one until the list has accumulated enough terms to be greater than the user input.

Date: February 2019.