Tareef Dedhar
20621325

Assignment 1: Question 3

Acknowledgments. I have completed this question with no outside sources.

Testing primality [10 marks]

Analyze the time complexity of the following pseudocode in terms of n using big-O notation. For this analysis, each operation on integers (including multiplication and squaring) takes constant time.

Algorithm 1: IsPrime(n)

```
j ← 2;
while j^2 ≤ n do
    k ← 2;
    while j * k ≤ n do
        if j * k = n then
            return False;
        k ← k + 1;
    j ← j + 1;
return True;
```

Solution. We shall analyze the algorithm in order, by scope blocks. First, we assign j the initial value of 2. This takes constant time. Then, we have a loop. This loop iterates on j, and if we reduce $j^2 \leq n$ to $j \leq \sqrt{n}$, and notice that j increments one at a time, we can see this loop will iterate $\sqrt{n}$ times in the worst case.

As for what we are doing $\sqrt{n}$ times, we have another loop, from k=2 to j*k≤n (by 1 each time) in the worst case. Since j is at least 2, in the worst case, this loop will iterate n/2 times. This loop performs an if check and multiplication, which are constant time operations, so the inner loop is O(n).

So, since we are looping $\sqrt{n}$ times, and in each of those loops entering another loop of O(n), our total worst-case runtime is $O(n\sqrt{n})$.