Getting familiar with algorithms

Deep C - Assignment 2.1

1 Introduction

Algorithms is the common language of all computer scientists around the world. It is not a formal or precise language with particular syntax and rules, it merely describes the computations required to solve a particular program. It can be described in plain English, Mathematical form, Pseudo-code or even source code of another language.

Your job for this assignment is to use the base skeleton code provided to you and fill the-in the blanks with the required implementation according to your understanding of the next following algorithms.

2 Algorithm 1: Prime numbers

The first algorithms we are going to review uses a very simple mathematical rules. Prime numbers.

2.1 Definition

A primary number is one that can be divided only by 1 or it self. n is a prime number if:

$$n \in N \setminus \{0,1\}, \exists u, v \in N \setminus \{0,1\} | u.v = n \tag{1}$$

2.2 Assignment

Your first assignment is to implement the Prime verification function. This method verifies that its input n is a prime number. It returns 1 on success and 0 on failure. To make sure that an integer number can be divided by another, we use the mod operator %. This operator returns the rest of division of its left operand by its right one, i.e $4\%2 = 0$ and $9\%2 = 1$. Obviously, if the result of division is equal to zero, than the first number can be divided by the second in the ensemble of natural numbers N. That is going to be our decisive constraint for the first algorithm.

Algorithm 1: Prime number verification

```
Result: 1 if n is prime, 0 otherwise
i ← 0;
for i ← 2 to n/2 do
    if n%i = 0 then
        return 1;
    end
end
return 0;
```

2.3 Algorithm Details

The loop is used to test if the number n can be divided by various other numbers i. At first sight, we realize that we need to start our counter i starting from 2, since every number can be divided by 1, but the reason we end our iterations at $n/2$ because no number can be divided by another that is bigger than its half (except for the number it self). Also note that the division here is called integer division, it does not return a real number.

Recall that a Prime number can be divided only by 1 or itself, thus we exclude them from our iterations since we are trying to verify by counter example, i.e finding a number that can divide n. If we cannot, we return 0.

For example, let us consider $n \leftarrow 17$, we will try the following values using our counter $[2, 3, 4, 5, 6, 7, 8]$.

2.4 Test

Once done coding, you should test it using the following values and expected outputs:

2	1
13	1
24	0
15	0
173	1
89	1
33	0
711	0

3 Algorithm 2: First 10 Prime numbers

Now that we know how to verify if a given unsigned integer n is prime or not, let's search for the first 10 prime numbers!

3.1 Assignment

Your second and final exercise for this assignment is to implement the following algorithms which finds the first 10 prime numbers starting from 2.

Algorithm 2: Searching for the first 10 prime numbers

```
Result: Nothing
counter ← 0
number ← 2
while counter < 10 do
    if isPrime(number) then
        print(number)
        counter ← counter + 1
    end
    number ← number + 1
end
```

3.2 Algorithm Details

Our algorithms contains two variables, *counter* is used to measure the numbers of prime numbers found, while *number* is the candidate prime number to verify. We keep incrementing *number* upon each iteration and when it is a valid prime number, we also increment the counter. Note that our looping condition is $counter < 10$, this is a strict less-than because we started counting from 0. If we started from one, we should use $counter <= 10$ otherwise we will get 9 prime numbers.

3.3 Test details

Your work is complete if you see the following output: $[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]$

4 Running algorithms by hand

I highly advice you to run algorithms by hand and verify that they give the correct result. You should always try a small number when doing some unless you want spend hours manually running the algorithms. To do so, simply rewrite every iteration with its own value, Here is an example of manual execution of the previous algorithm by hand:

counter	*number*	isPrime
0	2	1
1	3	1
2	4	0
2	5	1
3	6	0
3	7	1
4	8	0
..	..	..

Notice how on each iteration, we display all the variables and the result of the function, then when $isPrime = 1$ we increment *counter*.

5 Conclusion

Lectures represent about 30% of the knowledge, the other 70% are acquired through practice, such as these assignments. You may some difficulties throughout your path, path you have to be persistent to achieve your goal. If you are stuck, read the assignment paper multiple times or refer to the forums for help. Wish you the best of luck.