PROGRAMMING ASSIGNMENT REPORT

Name	**Email**
Kübra Kutlu	kubrakutlu9@gmail.com

July 27, 2019

Contents

1 Overview

This assignment consists of two main parts: Native command-line application for Linux written in C++ language and Python 3.x script. The native application decides whether a given positive integer N is prime or not by accepting one command-line argument. It achieves this by checking the remainder from the division by 2 which is shown in the code below.

Code 1: C++ function to find prime number.

```
1  void prime(int n)
2  {
3      int i;
4      bool isPrime = true;
5
6      for(i = 2; i <= n/2; ++i)
7      {
8          if(n % i == 0)
9          {
10             isPrime = false;
11             break;
12         }
13     }
14     if (isPrime && n != 0){
15         cout<<n<<" is a prime number."<<endl;
16         cout.flush();
17     }
18     else{
19         cout<<n<<" is not a prime number."<<endl;
20         cout.flush();
21     }
22 }
```

If the user does not provide any input, the application prints to stdout "You did not pass any number." whereas if the user gives an input that is non-integer or bigger than integer limit of C++, it prints "You entered an invalid character or it is bigger than int.". Main function of the code is given below:

Code 2: C++ main function for the native application.

```
int main(int argc, char* argv[])
{
    if(argc<2)
    {
        cout<<"You did not pass any number."<< endl;
        return -1;
    }
    else
    {
        char* p;
        errno=0;
        int N = (int) strtol(argv[1], &p, 10);

        if (*p != '\0' || errno !=0 || N > INT_MAX)
        {
        cout<<"You entered an invalid character "<<
        "or it is bigger than int."<<endl;
        cout.flush();
        return -1;
        }
        else
        {
        prime(N);
        }
    }
```

```
26 return 0;
27 }
```

Python script, on the other hand, calls the native application for all integer numbers from 1 to M, again M is provided by the user with a command-line. The process of calling the native application is done by using **subprocess** module of python. Thanks to this module, python script spawns the native application, connects to its output pipe and then obtain its return code. Moreover, the script records the run time of the native application by also using subprocess module. Note that the inaccuracy of PC for run time calculation is ignored. The python code for these implementations is as the following:

Code 3: Outputting code in the Python script.

```
1  import time
2  import subprocess
3  import os
4
5  M = input("Please enter a positive integer number: ")
6  print("\n")
7
8  COUNT_PRIME = 0
9  COUNT_NONPRIME = 0
10 EXECUTION_TIME_1 = 0
11 EXECUTION_TIME_2 = 0
12 CWD = os.getcwd() #current working directory
13 #print(cwd)
14
15 for pythonMessage in range(1, int(M)+1):
16     value = str(pythonMessage).encode("UTF-8")
17     timeoutInSeconds = 1 #timeout value
18     cmd = [CWD+"/Prime", value]
19     #Prime is the release file of native command application
```

```
    proc = subprocess.Popen(cmd, stdout=subprocess.PIPE)
    timeStarted = time.time() #start time
    cmdTimer = "sleep "+str(timeoutInSeconds)
    cmdKill = "kill "+str(proc.pid)+" 2>/dev/null"
    # Combine commands above.
    cmdTimeout = cmdTimer+" && "+cmdKill
    # Start timeout.
    procTimeout = subprocess.Popen(cmdTimeout, shell=True)
    output = proc.communicate()[0].decode()
    # End of the process.

    print(output)
    timeDelta = time.time() - timeStarted
```

Finally, the outputs of the native application are parsed as well as the two lines of text given in the code below:

Code 4: Outputting code in the Python script.

```
 if output == str(pythonMessage)+' is a prime number.'+'\n':
        COUNT_PRIME += 1
        EXECUTION_TIME_1 += timeDelta
    elif output == str(pythonMessage)+' is not a prime number.'+'\n':
        COUNT_NONPRIME += 1
        EXECUTION_TIME_2 += timeDelta
print("Number of primes: %d (total execution time: %.10f in seconds)"
      % (COUNT_PRIME, EXECUTION_TIME_1))
print("Number of non-primes: %d (total execution time: %.10f in seconds) \n"
      % (COUNT_NONPRIME, EXECUTION_TIME_2))
```

x is the number of prime (or non-prime) numbers between 1 and M and y is the total execution time taking for all runs of prime (or non-prime) numbers.

2 Building and Usage Instructions

1. Open the terminal and go to Prime folder's directory using "cd" command. To compile the C++ native application, enter "g++ main.cpp -o main". You can check the version of your gcc compiler by typing "g++ -v". After compiling the application, you can run it by providing a positive integer number. For example, if we want to see if 10 is a prime number or not, the terminal should like this:

```
kubrakutlu@kubrakutlu-HP-Notebook:~$ cd c++/RW_Test/Prime
kubrakutlu@kubrakutlu-HP-Notebook:~/c++/RW_Test/Prime$ g++ main.cpp -o main
kubrakutlu@kubrakutlu-HP-Notebook:~/c++/RW_Test/Prime$ ./main 10
10 is not a prime number.
kubrakutlu@kubrakutlu-HP-Notebook:~/c++/RW_Test/Prime$
```

2. Python script is inside of bin/Release folder and it is named "prime_call.py". Copy the path of this file and type it to the terminal as shown below:

```
kubrakutlu@kubrakutlu-HP-Notebook:~$ cd /home/kubrakutlu/c++/RW Test/Prime/bin/Release
```

3. You can check the current directory by typing "pwd" and list the files inside of the folder with "ls" command.

4. Type "python prime_call.py" and enter. You will see the following screen:

```
kubrakutlu@kubrakutlu-HP-Notebook:~/c++/RW_Test/Prime/bin/Release$ python prime_call.py
        ******************************************
        *****  Rightware Programming Assignment  *****
        ******************************************
Please enter a positive integer number:
```

5. Type a positive integer number. For example, if you type 10, you will see the following output:

```
Please enter a positive integer number: 10


1 is a prime number.

2 is a prime number.

3 is a prime number.

4 is not a prime number.

5 is a prime number.

6 is not a prime number.

7 is a prime number.

8 is not a prime number.

9 is not a prime number.

10 is not a prime number.

Number of primes: 5 (total execution time: 0.0187232494 in seconds)
Number of non-primes: 5 (total execution time: 0.0109035969 in seconds)
```

6. If you want to try a new number, again type "python prime_call.py" and enter a new number.