UNIVERSIDAD DE SAN BUENAVENTURA CALI

Universidad de San Buenaventura Cali
Facultad de Ingeniería
Departamento de Ciencias y Tecnologías de la Información
Laboratorio de Software I
Taller Dos Punto Uno
"Cohesión y acoplamiento"

Objetivos Pedagógicos

El objetivo del Taller Dos es seguir fomentado la adquisición de competencias de diseño y trabajo en equipo, por parte de los estudiantes.

Instrucciones

Este punto hace parte del Taller Dos, que se trabajará en parejas.

El taller consolidado, se debe enviar a aulas virtuales, en los tiempos y modos establecidos por el docente.

Como descripción general de las entregas, se debe enviar, en un archivo comprimido de extensión .zip, el código fuente que da solución al problema (i.e., main.zip), los archivos de prueba que apliquen para el caso, y un reporte en PDF con la captura de pantalla de la ejecución del código, de acuerdo con las especificaciones e ilustrando el cumplimiento de todos los desafíos. En su reporte debe incluir capturas de pantalla del código, su ejecución y prueba. En el reporte se debe incluir la dirección del Replit empleado, si el código se desarrolló empleando dicha plataforma. Tanto el archivo .PDF, como el de código fuente deben indicar los nombres de los integrantes del equipo (i.e. nombre completo, código, correo electrónico). No se deben enviar ejecutables, ni archivos pertenecientes a la configuración de Replit, ni de ningún otro entorno de desarrollo.

La fecha y plazo de entrega se informará en clase y en Aulas Virtuales de manera oportuna.

Este punto se debe abordar empleando dos lenguajes: C++ y Python. Ósea se entregarán dos soluciones a este mismo problema, cada uno escrito en los lenguajes anteriormente mencionados. Adicionalmente, sobre estas soluciones se realizará *code review* individual y en el lenguaje seleccionado por el docente para cada estudiante.

Cuestionario

1. Buscador de Primos

Un número primo es un número positivo, mayor a 1, que tiene únicamente como divisores enteros exactos a sí mismo y a la unidad. Por ejemplo, el número 2 es el único primo par. Valores como por ejemplo 5, 7, 11, 13, 103, 107 y 109, satisfacen la condición de primalidad - i.e., son número primos. Por lo contrario, valores como

el 10, 20, 30, por ejemplo, no son primos, dado que son divisibles entre 2, 5, y 10, aparte de si mismos y la unidad – i.e., tienen más de dos divisores. Los valores de 0 y 1, no se consideran como primos.

Los diez primeros valores de números primos se ilustran en la Tabla 1, a continuación.

Tabla 1. Diez primeros números primos.

10 primeros Valores Primos
2
3
5
7
11
13
17
19
23
29

Por otra parte, la Tabla 2, muestra ejemplos de números, que no son primos y su respectivo más pequeño divisor – diferente de la unidad.

Tabla 2. Números no primos y su respectivo más pequeño divisor.

Numero	Más pequeño divisor
4	2
33	3
45	3
87	3
88	2
121	11
155	5
169	13
187	11
221	13

Los números primos son de interés en la criptografía, entre otras ramas de la matemática y la ingeniería. Por eso, dado una serie de números resulta de interés el poder saber si son primos o no.

Desafío Uno Diseñe e implemente una función llamada esPrimo/es_primo, que mediante una instrucción de ciclo resuelva la funcionalidad de determinar si un número es primo, o en caso contrario, de no ser primo permita saber cuál es el respectivo más pequeño divisor.

Adicionalmente, dado un archivo de texto de entrada (“input.txt”), conteniendo números enteros positivos en el rango [0, 7778742049], se debe determinar si cada uno de los valores presentes en el archivo corresponde o no al de un número primo. En caso de que un valor sea un número primo se debe generar un mensaje

indicándolo, compuesto por el número primo analizado y la palabra *true*. En caso contrario, que el número no sea primo, se debe imprimir un mensaje compuesto por el número, la palabra *false* y el respectivo más pequeño divisor, aparte de la unidad y sí mismo.

Dicho cálculo aplica para todos los números contenidos en el archivo de entrada. Los mensajes de cada cálculo se deben imprimir a la consola (i.e., salida estándar ó std::cout en C++).

La estructura del archivo "input.txt" incluye en la primea línea, la cantidad de números que deben ser verificados, seguido por la frase unsigned int, y luego los valores a analizar, cada uno en una línea del archivo. En la consola se imprime una línea por cada valor analizado. Un ejemplo de entradas y de salidas se ilustra en la Tabla 3.

Desafío Dos: Diseñe y desarrolle un programa que, apoyado y empleando la función esPrimo/es_primo, calcule y genere las salidas para el archivo de entrada, acorde con lo anteriormente descrito.

Tabla 3. Ejemplo de archivos de entrada y de salida, respectivamente.

input.txt	Impresión por consola
12	240 false 2
unsigned int	5 true
240	21 false 3
5	25 false 5
21	89 true
25	1 false 1
89	0 false 0
1	193 true
0	361 false 19
193	437 false 19
361	491 true
437	999 false 3
491	
999	

Desafío Tres:

Los valores que SI sean primos y la palabra true, se deben imprimir en consola empleado el color verde.

Los valores que NO sean primos, la palabra false y el primer divisor, se deben imprimir en consola empleado el color rojo.

2. Restricciones de diseño e implementación

- Este punto del taller se debe abordar empleando dos lenguajes: C++ y Python.
- La lectura de los datos, debe estar desacoplada del cálculo. Esto implica que no se debe ir realizando ningún cálculo sobre los valores leídos a medida que estos se van leyendo.

- Se deberá leer todo el archivo de entradas, cargar su contenido (total o parcialmente) en alguna estructura de datos, y luego si posteriormente realizar el procesamiento.
- Se puede asumir que el archivo no está corrupto y se ajusta adecuadamente a la estructura especificada en la Tabla 3.
- La funcionalidad de la impresión a colores no debe estar en el mismo archivo de código que el de la función principal (i.e., función main o de punto de entrada). En consecuencia, se deben emplear múltiples piezas de código, a nivel físico de archivo, y a nivel lógico de organización.