Tecnología Digital 1: Introducción a la Programación

Trabajo Práctico 1

Escuela de Negocios, UTDT - Primer semestre de 2023

El objetivo de este TP es ganar experiencia en la escritura de código para resolver problemas pequeños, de manera controlada y usando buenas prácticas de programación.

Los nuevos asistentes de IA, como ChatGPT o CoPilot, son herramientas impresionantes que aceleran la escritura de código. Cuando son **guiados por el ojo crítico de un buen programador/a**, pueden generar soluciones excelentes, y así acortar los tiempos de desarrollo.

En esta materia, sería un error usar asistentes de IA para resolver TPs. No es lo mismo escribir código propio que analizar código ajeno. **Escribir código** (ya sea solo o en equipo) **es la única manera de desarrollar ese espíritu crítico** necesario para convertirse en buen programador/a.

Este TP está pensado para ser resuelto sin la ayuda de asistentes de IA. ¡Solo así tiene sentido!

Dicho esto, para quienes tengan muchas ganas de experimentar con asistentes de IA, les proponemos lo siguiente:

Opcionalmente, pueden entregar una segunda versión del código, generada (después de la primera) con la ayuda de un asistente de IA. Les pedimos que en el reporte escrito incluyan la secuencia de comandos usados para generar el código. Con esto, nosotros les podremos dar feedback también sobre el código generado de esta manera. Esta parte de la entrega no contará para la evaluación (no sumará ni restará puntos).

Insistimos en que es importante que primero resuelvan el TP sin ayuda, para ganar la experiencia que les servirá más adelante en la carrera (y también en los exámenes de esta materia).

Antes de comenzar, se recomienda resolver el Ejercicio 12 de la Guía 4, para entender el sistema de numeración binaria y cómo pasar números enteros de base 2 a 10 y viceversa.

Decimos que un número $n \in \mathbb{N}$ (es decir, un entero mayor que 0) es *k-dominante* (para $k = 0$ o $k = 1$) si y solo si más de la mitad de los símbolos en su representación binaria son el símbolo k.

Por ejemplo, 17 es un número 0-dominante, porque su representación binaria es `10001`, con tres 0s y dos 1s. El número 64 (`1000000`) también es 0-dominante. Por su parte, 13 (`1101`) y 127 (`1111111`) son 1-dominantes. Notar que existen números que no son 0-dominantes ni 1-dominantes, cuando coinciden las cantidades de 0s y 1s; por ejemplo, el 12 (`1100`) y el 50 (`110010`).

El objetivo de este trabajo es crear un programa en Python que ofrezca las siguientes funcionalidades:

- dado un string s, contar la cantidad de ocurrencias de `'0'` en s;
- dado un string s, contar la cantidad de ocurrencias de `'1'` en s;
- dados $k \in \{0, 1\}$ y $n \in \mathbb{N}$, determinar si n es k-dominante;
- dados $n, m \in \mathbb{N}$, con $n \leq m$, obtener la lista de números 0-dominantes entre n y m, inclusive en ambos casos (es decir, mayores o iguales a n, y menores o iguales a m), ordenada de menor a mayor;
- dado $n \in \mathbb{N}$, calcular la suma de todos los números 0-dominantes estrictamente menores que n.

Para cada una de estas funcionalidades requeridas, se pide:

a) Escribir la especificación de una función que resuelva el problema.

b) Construir un conjunto de casos de test, que se consideren suficientes para ilustrar y poner a prueba el comportamiento esperado.

c) Pensar un algoritmo y llevarlo a un programa en Python que resuelva el problema.

d) Asegurarse de que la función pasa los casos de test construidos.

e) Para los programas que contengan ciclos, demostrar que los mismos terminan, escribir predicados invariantes que describan el trabajo realizado, y usarlos para mostrar que los programas son correctos. En el caso de que una función sin ciclos utilice a una función auxiliar que contenga un ciclo, solo deberán darse estas justificaciones sobre la función auxiliar.

Al ejecutar el programa se deberá desplegar un menú desde el cual seleccionar la funcionalidad a la que se desea acceder. Dependiendo de la opción elegida y los argumentos ingresados, el programa tendrá que mostrar la respuesta que corresponde. A continuación puede observarse parte de una ejecución del programa en la cual el usuario consulta si el número 17 es o no 0-dominante:

```
Funciones disponibles
---------------------
A. Cantidad de 0 [s]
B. Cantidad de 1 [s]
C. Es k-dominante? [k, n]
D. 0-dominantes entre [n,m]
E. Suma de 0-dominantes menores que [n]
X. Finalizar

Seleccione una opción: C

Ingrese k: 0
Ingrese n: 17
El número 17 es 0-dominante.
```

La ejecución debe concluir únicamente si se selecciona la opción `Finalizar`; caso contrario, tiene que desplegar nuevamente el menú de opciones.

En la siguiente tabla se ilustran los mensajes que debe imprimir el programa para algunas funciones y valores ingresados:

Función seleccionada	Argumento(s)	Mensaje por pantalla
`Cantidad de 0 [s]`	`xyz_012`	`'0' aparece 1 vez en 'xyz_012'.`
`Cantidad de 0 [s]`	`0101010`	`'0' aparece 4 veces en '0101010'.`
`Cantidad de 1 [s]`	`xyz_012`	`'1' aparece 1 vez en 'xyz_012'.`
`Cantidad de 1 [s]`	`0101010`	`'1' aparece 3 veces en '0101010'.`
`Es k-dominante? [k,n]`	`0,17`	`El número 17 es 0-dominante.`
`Es k-dominante? [k,n]`	`1,13`	`El número 13 es 1-dominante.`
`0-dominantes entre [n,m]`	`1,17`	`0-dominantes entre 1 y 17: [4, 8, 16, 17]`
`0-dominantes entre [n,m]`	`10,15`	`0-dominantes entre 10 y 15: []`
`Suma de 0-dominantes menores que [n]`	`4`	`Suma de 0-dominantes menores que 4: 0`
`Suma de 0-dominantes menores que [n]`	`10`	`Suma de 0-dominantes menores que 16: 12`

Se deben entregar los siguientes cuatro archivos:

- **`kdominante.py`**, con el código de todas las funciones implementadas: `cantidad_de_0`, `cantidad_de_1`, `es_kdominante`, `cero_dominantes_entre`, `suma_cero_dominantes_menores`, y cualquier otra función auxiliar que se defina. Es obligatorio especificar el tipo de todas las variables y funciones mediante sugerencias de tipos (*type hints*). Las especificaciones de las funciones se deben incluir con el formato de *docstrings* presentado en clase. Este archivo **sólo** debe incluir la implementación de las funciones; es decir, **no** debe ser directamente ejecutable.
- **`kdominante-testing.py`**, con los tests de unidad de todas las funciones definidas.
- **`tp1.py`**, con el código principal para ejecutar el programa como se indica arriba. Este código es el encargado de imprimir el menú, invocar a las funciones definidas en `kdominantes.py` y mostrar los resultados por pantalla.
- **`informe.pdf`**, un documento con *(i)* la justificación de los casos de test elegidos (es decir, por qué son buenos candidatos para revelar la presencia de errores); y *(ii)* las respuestas al punto e) enunciado arriba.

Incluir cualquier aclaración adicional que se considere necesaria sobre cualquier parte del trabajo. Se espera que este documento sea conciso, de no más de cuatro páginas, y en formato PDF.

La carpeta adjunta `templates` contiene archivos de ejemplo para usar como referencia.

Observaciones y aclaraciones:

- El trabajo se debe realizar en grupos de **tres estudiantes**. La entrega consistirá en trabajo original realizado íntegra y exclusivamente por quienes integran el grupo.
- La fecha límite de entrega es el **sábado 15 de abril a las 23:55**. Los TPs entregados fuera de término serán aceptados hasta 48h pasada esta fecha, pero la demora incidirá negativamente en la nota.
- Los archivos deben subirse al formulario *TP1: Entrega* en la página de la materia en el campus virtual.
- El programa entregado debe correr correctamente en `Python3`.
- Las únicas bibliotecas que se permite importar son `typing` (para *type hints*), y `unittest` (para correr los tests en `kdominante-testing.py`).
- La función `print` solamente puede invocarse en el cuerpo principal del código, en el archivo `tp1.py`. Es decir, no puede usarse en las funciones requeridas ni en las funciones auxiliares.
- Sólo pueden usarse las instrucciones y estructuras de control vistas en clase. En particular, no pueden usarse iteradores (`for`).
- Para obtener la representación binaria de un número natural, se permite usar la función `bin` de Python, que devuelve un string con formato `'0b10100'`. No es necesario especificar, implementar ni verificar la función `bin` en este TP.
- Puede suponerse que el usuario siempre invocará las funciones de manera correcta. Es decir, si hay errores de tipo o valor de los argumentos provistos, no se espera comportamiento alguno del programa (podría colgarse o terminar en un error, por ejemplo).
- **Forma de evaluación:** Se evalúan tres aspectos de la entrega: (1) el *código*: no solo su correcto funcionamiento, sino también que se sigan las buenas prácticas de programación vistas desde la primera clase: uso adecuado de tipos y estructuras de control, claridad en los nombres de variables y funciones auxiliares, comentarios, reutilización de funciones, etc.; (2) las *especificaciones* de las funciones; y (3) la *verificación* de los programas entregados, mediante testing de unidad, demostraciones de terminación y correctitud de ciclos, según corresponda.