DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2020)

Tarea 1

Fecha de entrega: 8 de abril de 2020, 18:59 hrs.

Entrega y evaluación

Esta tarea se compone de dos partes, una programada que se corregirá mediante *hackerrank* y una teórica que se corregirá mediante *canvas*. **Deberás subir, además, el código que utilices para cada pregunta a *canvas* (de no hacerlo se considera que no entregaste la tarea)**. Nos reservamos el derecho a correr tu código en local, aplicar los *tests* de esta forma y conservar este puntaje.

Link al contest en canvas:

https://www.hackerrank.com/iic2343-20201-t1

Además, en el cuerpo de ayudantes preparamos un material para que aprendas todo lo que necesitas de C para esta tarea:

https://mybinder.org/v2/gh/IIC2343/jupyter-gcc/master

Es un *Jupyter Notebook*, por lo que puedes ejecutar las celdas.

Ponderación

Las preguntas teóricas valen un 30 % de la nota, mientras que las preguntas prácticas valen un 70 % de la nota.

Código base para cada pregunta

Para cada pregunta encontrarás en *canvas* un código base que puedes utilizar. Este contiene las declaraciones de una o más variables, recibir el input y a veces el *print* del resultado.

1. Y2k38

El efecto 2038 o Y2k38 está vinculado a la representación de la fecha y hora en gran parte de los sistemas digitales de 32 bits. Para conocer la fecha estos toman como base el 1 de enero de 1970 a las 00:00 horas y cuentan cada segundo a partir de ahí. Sin embargo el 19 de enero de 2038 ocurrirá un overflow en la variable utilizada como contador, esto ocasionará que el valor guardado se interprete como un número negativo y por consiguiente la fecha se interprete como 12 de diciembre de 1901, lo que tiene como consecuencia que gran parte del software construido para sistemas de 32 bits falle o se comporte de manera errática.

1. Dado un número de bits, determina en cuánto tiempo ocurrirá el overflow. Implrímelo en formato `yyyy/dd hh:mm:ss`. Considera los años de 365 días.

Donde:

- `yyyy`: año con cuatro cifras.
- `dd`: días con al menos cifras.
- `hh`: horas con dos cifras.
- `mm`: minutos con dos cifras.
- `ss`: segundos con dos cifras.

Además, el número de bits será siempre mayor o igual a 0 y menor o igual a 64.

Para responder en Canvas

1. Investiga sobre el problema del año Y2k38, describe una solución por *software* y explica algún problema relacionado a su aplicación.
2. Menciona y describe un ejemplo de la vida real en el que este error haya causado problemas. Incluye tus fuentes.

2. Shifting

Las operaciones *bitwise*, también llamadas *bit a bit*, operan a nivel individual sobre cada *bit* y son las operaciones primitivas soportadas a nivel de hardware. Entre ellas se encuentran los *shifts* que consisten a grandes rasgos en desplazar los *bits* en alguna dirección.

1. Se te entregará un número entero con signo. Deberás realizar las operaciones *shift left* y *shift right* **lógicos** (SHL y SHR). Luego tendrás que imprimir en pantalla el resultado obtenido en binario en *big endian*.

Para responder en Canvas

1. Investiga los *shifts* lógicos y aritméticos. ¿Cuál es la diferencia entre ellos en cuanto al resultado obtenido?

3. Colores

En las pantallas LED los colores se codifican según la intensidad de la luz roja, verde y azul (RGB, por sus nombres en inglés), por otro lado en impresión es común codificar los colores de acuerdo a la proporción de los pigmentos cian, magenta, amarillo y negro (CMYK, por sus nombres en inglés, a excepción de la K que es negro). A estas codificaciones se les conoce como *modelos de color* y son una representación matemática de los colores usando vectores en varias dimensiones. Así el color negro en RGB se representa con el vector $(0, 0, 0)$ es decir la ausencia de luz, mientras que en CMYK es representado por $(0, 0, 0, 100)$, con el pigmento K (negro) al máximo.

Nota: si bien CMYK va de 0 a 1 o 0 a 100 %, nosotros usaremos representaciones donde los valores van de 0 al máximo entero representable con los bits que estemos tomando. Por ejemplo, si elegimos 8 *bits*, los valores irán de 0 a 255.

Descripciones de formatos

La siguientes codificaciones en 16 bits fueron creadas para este enunciado únicamente con motivos académicos, no son (necesariamente) el formato real en que los sistemas almacenan los colores en formato RGB y CMYK.

- Formato RGB - 5 6 5 bits para el rojo, verde y azul, respectivamente.
- Formato CMYK - 4 4 4 4 bits para el cyan, magenta, amarillo y negro, respectivamente.

Transformación de RGB a CMYK

1. Separa los valores R, G y B en 3 variables de 16 *bits* cada una. Sin signo. Escalados[1] a 8 bits.
2. Sean R_v, G_v, B_v los valores escalados, calcular los valores C, M, Y, K como sigue:
   - $K = 255 - max(R_v, G_v, B_v)$
   - $C = 255 \left(\frac{255 - R_v - K}{255 - K} \right)$
   - $M = 255 \left(\frac{255 - G_v - K}{255 - K} \right)$
   - $Y = 255 \left(\frac{255 - B_v - K}{255 - K} \right)$
3. Escalar los valores de C, M, Y, K a 4 bits y concatenar en el orden correspondiente.

Transformación de CMYK a RGB

1. Separa los valores C, M, Y y K en 4 variables de 16 bits cada una. Sin signo. Escaladas a 8 bits. Sean C_v, M_v, Y_v, K_v respectivamente.
2. Calcular los valores RGB como sigue:
   - $R = \frac{(255 - C)(255 - k)}{255}$
   - $G = \frac{(255 - M)(255 - k)}{255}$
   - $B = \frac{(255 - Y)(255 - k)}{255}$
3. Escalar y concatenar los valores a su correspondiente formato.

Deberás hacer...

1. Conversor RGB a CMYK.
2. Conversor CMYK a RGB.

[1] Ver el final de la sección.

Para responder en Canvas

1. Sea un valor A al que se le aplica una conversión RGB a CMYK obteniéndose A', si a A' se le aplica una conversión CYMK a RGB cuyo resultado es A'', existen casos en los que $A \neq A''$. Explique porqué ocurre esto y de un ejemplo.

Anexo: escalar valores

Entiéndase como escalar a calcular la proporción numérica correspondiente a la nueva base. Por ejemplo, si tomamos el 12 en 4 *bits* tenemos "`1100`" y queremos saber cuánto sería en 8 *bits*. Para ello calculamos:

$$\frac{12}{15} = \frac{X}{255}$$

12 es a 15 (el valor máximo representable con 4 bits) como X es a 255 (el valor más alto representable con 8 bits).

Con esto obtenemos que 12 es 204 escalado a 8 bits, que en binario resulta ser "`11001100`".

4. Dígito verificador (RUT)

El código ASCII *(del inglés American Standard Code for Information Interchange)* se utiliza para manejar caracteres alfanuméricos codificándolos como enteros con signo de 8 *bits*. La tabla original tiene 128 caracteres, de los cuales 32 son caracteres especiales que no tienen una representación gráfica, estos son los llamados "caracteres de control". Los restantes se reparten entre símbolos, dígitos, letras mayúsculas y minúsculas.

Se te entregarán RUT's como string por *stdin*, sin puntos y con guión de la forma XXXXXXXX sin guión ni dígito verificador, donde X son dígitos. El largo del RUT irá de 1 a 255 dígitos.

Ten presente que algunos RUTs vienen de otro sistema, en el que colocan algunos caracteres de control como autenticación adicional interna. Deberás ignorar los caracteres de control presentes en el *string* a la hora de realizar el cálculo.

1. Investiga como se calcula el dígito verificador. Luego, crea un programa que lo calcule e imprima en pantalla. Si el dígito verificador es una "K", imprímela en minúscula.

5. Antecesor y sucesor

Dado un número en punto flotante, IEEE754, deberás imprimir en pantalla con 150 cifras decimales el antecesor y el sucesor del número de acuerdo a la siguiente definición.

Antecesor

Sea a un *float* que tiene la estructura definida por la convención IEEE754. Definiremos el *antecesor* de a como:

$antecesor = max(\{x < a \mid x \in$ todos los posibles valores representables por la convención IEEE754$\})$

Nota: diremos que el antecesor de +0 es -0.

Sucesor

Sea a un *float* que tiene la estructura definida por la convención IEEE754. Definiremos el *sucesor* de a como:

$$sucesor = min(\{x > a \mid x \in \text{todos los posibles valores representables por la convención IEEE754}\})$$

Nota: diremos que el sucesor de -0 es +0.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.