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

IIC2343 - Arquitectura de Computadores (II/2019)

Tarea 1: Representación de números, operaciones bitwise y almacenamiento de datos

Conceptos que les van a servir para el resto de sus vidas.

Fecha límite de entrega: Jueves 29 de Agosto a las 21:00

1. Motivación

En esta tarea practicarán el uso de distintos tipos de datos, operaciones bitwise y formas de almacenar datos. Esto con el objetivo de que se familiaricen con los contenidos del curso y su aplicación a distintos escenarios, usando el lenguaje de programación C.

2. Desarrollo

La tarea se desarrollará individualmente por medio de la participación de un concurso en la plataforma HakerRank. Para ingresar al concurso abra el siguiente enlace:

https://www.hackerrank.com/t1-iic2343-2019-2

Dentro de la plataforma encontrará una serie de desafíos, también descritos como ejercicios en este documento. Para resolverlos deberá completar el código que se ofrece dentro de cada uno, pudiendo ejecutarlo en la misma plataforma, de tal modo que el programa sea capaz de procesar el texto de entrada y producir la salida adecuada el problema en cuestión. Para estos dos requisitos se recomienda que investigue las funciones "printf" y "scanf" de la librería "stdio.h" respectivamente.

Podrá encontrar códigos base para los desafíos en los anexos de este documento. Y como una breve referencia a C, podrán introducirse en los distintos aspectos del lenguaje usando los Jupyter Notebooks disponibles en el siguiente enlace:

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

3. Entrega

La entrega de cada desafío del concurso se realiza por medio del envió del código desarrollado, luego de lo cual la plataforma entregara el puntaje obtenido respecto a los casos de prueba de dicho desafío. Puede enviar el código las veces que estime conveniente antes del Jueves 29 de Agosto a las 21:00.

4. Evaluación

El concurso en Hackerrank contiene 13 problemas, cada uno con un puntaje asociado. En total son 600 mil puntos. Cada participante cuenta con una base de 100 mil puntos y su nota en esta tarea será el puntaje conseguido, dividido en 100 mil.

5. Ejercicios

A continuación de describen los ejercicios propuestos para esta tarea y sus condiciones.

5.1. Un DCCuento de navidad (1 puntos)

El pasado 25 de Junio se celebró la navidad en el DCC, tras una propuesta apoyada de forma unánime en una votación compuesta íntegramente por mí. En esta, se manifestó una gran preocupación por la disonancia entre la decoración navideña y la estación del año, por lo que se decidió trasladar la fecha de esta festividad.

Para la celebración, los distintos cuerpos de ayudantes se comprometieron con determinados elementos para la fiesta. El segundo mejor cuerpo de ayudantes, Programación Avanzada, se comprometió con el árbol y su decoración.[1] Mientras que nosotros, el mejor cuerpo de ayudantes, nos encargaríamos de la comida y bebida. El profesor Yadran fue el encargado de hacer las compras, pero por una falla en los componentes de hardware fundamentales que componían nuestra lista de compras, terminó llegando con más de 900 cafés, abriendo así la cafetería navideña de Don Yadran. ¡Ayúdanos a programar una solución por software que evite que eso pase el próximo año! Adicionalmente, para una mayor eficiencia, un excelente ayudante de Estructuras de Datos nos sugirió realizar BFS (**NO** lo implementarán, ¡tranquilos!) por el supermercado entre varias personas, para luego comparar nuestras listas y determinar qué productos ya fueron comprados.

Requisitos:

Para esto deberás hacer un programa que sea capaz de manejar la lista de lo que ya ha sido comprado. La lista se representará como **un** entero sin signo de 32 bits, donde cada bit representará un elemento de la lista, tomando el valor 0 si no ha sido comprado y 1 ya lo fue. Los elementos están enumerados del 0 al 31 (es decir, la indexación parte en 0) contando desde el bit más significativo al menos significativo.

Desafíos:

1.1) Determinar cual es el índice del primer elemento de la lista que falta por comprar **(0,3 puntos)**

input: Una línea con un entero sin signo de 32 bit en decimal, que representa la lista.
output: una línea con un entero en decimal, que representa el índice del elemento.

1.2) Determinar el número de elementos que faltan por comprar **(0,3 puntos)**

input: Una línea con un entero sin signo de 32 bit en decimal, que representa la lista.
output: una línea con un entero en decimal, que representa el número de elementos.

1.3) Resumir los elemento comprados de múltiples listas en una sola lista, sin importar que estén duplicados. **(0,4 puntos)**

[1] Funcionó bien aunque todos seguimos preguntándonos por qué usaron un algarrobo, un árbol que parece más de verano, si movimos la fiesta al invierno precisamente por la decoración.

input: una línea con un entero sin signo en decimal, que representa el número de listas, seguido por ese número de líneas, donde cada una tiene un entero sin signo de 32 bit en decimal, que representa una de las lista.

output: Una línea con un entero sin signo de 32 bit en decimal, que representa la lista.

5.2. Autopsia.float (1,2 puntos)

El malvado doctor Van Elliott desea enseñarte a ti, su fiel asistente, a diseccionar datos tipo float y así conocer su estructura interna. Sigue sus macabras enseñanzas, toma el hacha para disecciones de alta precisión y haz un programa que pueda calcular:

Requisitos:

Complete el código de tal manera que pueda, según la especificación de float del estándar IEEE754, extraer la información requerida.

Desafíos:

2.1) El exponente del dato **(0,5 puntos)**

input: Una línea de texto con un float en decimal.

output: Una línea con un entero en decimal correspondiente al exponente ya procesado.

2.2) El signo del dato **(0,5 puntos)**

input: Una línea de texto con un float en decimal.

output: Una línea con un 0 si es positivo y un 1 si es negativo.

2.3) Imprimir el dato con solo dos decimales (truncando[2]) **(0,2 puntos)**

input: Una línea de texto con un float en decimal.

output: Una línea de texto con un float en decimal con solo 2 decimales de precisión.

5.3. ¡Emilio regresa del pasado! (1 puntos)

Emilio, un místico ex-ayudante del curso, regresa del pasado para llevar a cabo la labor de ser ayudante del curso una vez más antes de morir, sin embargo, para poder conectar los portales para que pueda moverse por el espacio-tiempo hasta nuestra era, es necesario establecer la conexión por WiFi del portal. Ayúdanos a configurar la dirección IPv4[3], para lograr nuestro objetivo.

Requisitos:

Investigar como se relacionan la IP y la máscara de subred en el protocolo IPv4 para realizar programas que permitan:

[2] Lamentablemente en los supermercados no venden 2.24922187460000 Kg de queso.

[3] Considera que una dirección IP bajo el protocolo IPv4 corresponde a 32 bits, y que cada subred tiene una máscara asociada.

Desafíos:

3.1) Obtener el identificador de red según la IP y la máscara de subred. **(0,3 puntos)**

input: Una línea con un entero sin signo de 32 bit en decimal, cuyos 4 bytes corresponden a una IP. Seguida por otra línea con un entero sin signo de 32 bit en decimal, cuyos 4 bytes corresponden a una máscara de subred.

output: Una línea con 4 enteros sin signo en decimal separados por puntos, donde cada entero corresponde a uno de los bytes del identificador de la subred.

3.2) Obtener el número de bits del identificador del host a partir de la máscara de subred. **(0,3 puntos)**

input: Una línea con un entero en decimal, cuyos 4 bytes corresponden a una máscara de subred.

output: Una línea con un entero en decimal, que representa el número de bits del identificador del host.

3.3) Indicar si dos direcciones IP comparten la misma subred, a partir de estas y una máscara de subred. **(0,4 puntos)**

input: Una línea con un entero sin signo de 32 bit en decimal, cuyos 4 bytes corresponden a una IP. Seguida por otra línea con un entero sin signo de 32 bit en decimal, que cuyos 4 bytes corresponden a una segunda IP. Y una línea con un entero sin signo de 32 bit en decimal, cuyos 4 bytes corresponden a una máscara de subred.

output: Una línea que contenga "True" o "False" según corresponda.

5.4. Nibbles, el hamster astronauta (1 puntos)

En un capítulo de Los Simpsons, Homero y Bart pretenden enviar a Nibbles, su hamster, al espacio. Sin embargo, cuando el cohete se desvía Homero intenta asistirlo, diciéndole que con una palanca azul podrá hacer descender el cohete. ¿Problema? Los hamsters solo ven en blanco y negro. ¡Ayuda a Homero a ver como un hamster y poder darle las indicaciones correctas a Nibbles para que pueda descender su cohete de forma segura!

Requisitos:

Para esto deberás a partir de un entero de 16 bits obtener los valores RGB originales almacenados en este: 5 bits para el rojo, 6 bits para el verde y otros 5 bits para el azul. Y luego transformar el color RGB en un color en escala de grises, representado por un entero de 5 bits.

Para calcular la el color en la escala de grises usarando el **promedio de Springfield**. Para esto, primero, debemos aumentar los valores de 5 bits a 6. En segundo lugar, se debe obtener el **promedio de Springfield** entre estos tres valores. Normalmente, para calcular el promedio sumamos los n elementos, y a esa suma la dividimos por n. Pero para el **promedio de Springfield**, además sumar los n elementos le adicionamos un uno, y luego lo dividimos en la cantidad de elementos, tal como el promedio regular.[4]

Finalmente, al **promedio de Springfield** se divide por dos obtener 5 bits como resultado. Esta última división debe dar un resultado aproximando por redondeo simple, es decir, se aproximar hacia abajo en caso de que el primer decimal es menor a 5, y hacia arriba en caso contrario.

Desafíos:

4.1) Extraer los datos de colores y calcular su equivalente en escala de grises. **(1 puntos)**

input: Una línea con un entero sin signo de 16 bit en decimal, que representa los valores RGB.

output: Una línea con los valores de los colores con el formato "R: %u G: %u B: %u", seguido por una línea con un entero en decimal, que representa el color en escala de grises.

[4] todo es un nombre inventado para decir que al numerador del promedio se le suma uno.

5.5. El Profesor Layton y los Caracteres de Control. (0,6 puntos)

En la codificación ASCII existen varios tipos de caracteres[5] de control que no tienen una representación gráfica pero cumplen un rol dentro del texto. Dicen que en una misteriosa secuencia de caracteres, al ser separada entre caracteres de control e imprimibles, mostrará el camino hacia un mágico tesoro que nos dará el superpoder de pasar todos nuestros ramos. El Profesor Layton, famoso arqueólogo y aficionado a los puzle desea ayudarte a descifrar el secreto, pero necesita primero filtrar los caracteres que sirven de los que no.

Requisitos:

Para eso deberás crear un programa que sea capaz de leer caracteres, discernir su tipo respecto a la tabla ASCII e imprimir solo aquellos que sean imprimibles.

Desafíos:

5.1) Filtrar los caracteres recibidos, imprimiendo solo aquellos que son imprimibles. **(0,6 puntos)**

input: Una línea con una cadena de caracteres.

output: Una línea con una cadena de caracteres.

5.6. JessiNet (1,2 puntos)

Jessica tiene una empresa de telecomunicaciones y necesita que le hagas un programa que genere y verifique los checksums de los mensajes que envía y recibe mediante el protocolo NMEA0183, utilizado en los sistemas GPS para asegurar la integridad del mensaje, ya que la pieza de hardware que lo hacía se rompió.[6]

Requisitos:

Debe lograr calcular el checksum descrito en por el protocolo NMEA0183. En términos prácticos ,para estos desafíos, son mensajes de hasta 80 caracteres ASCII delimitados por "$" al principio y *LF* al final. Justo antes de *LF* va un checksum escrito como "*CS", donde "CS" son 2 caracteres hexadecimales en mayúsculas que representan el resultado de aplicar XOR entre todos los caracteres del mensaje entre "$" y "*", sin incluirlos. Pueden leer una descripción más detallada en el cuarto párrafo de esta dirección: https://www.gpsinformation.org/dale/nmea.htm.

Desafíos:

6.1) Calcule el checksum e imprímalo en pantalla, usando dos caracteres hexadecimales en mayúscula. **(0,6 puntos)**

input: Una línea con una cadena de caracteres que inicia con "$" y termina con "*".

output: Una línea con dos caracteres hexadecimales en mayúscula, que representan el checksum.

6.2) Verifique el checksum de una secuencia recibida e imprima en pantalla si este es correcto o no. **(0,6 puntos)**

[5] Un caracter no es más que un byte en un computador, representado por el tipo de dato "char", es al mismo tiempo un número entero de 8 bits con signo.

[6] Este tipo de operaciones pueden hacerse tanto por software como por hardware, sin embargo, funciona más rápido cuando se hace por hardware. Aunque requiere hardware especializado.

input: Una línea, con una cadena de caracteres que inicia con "$" y termina con un "*" seguida de dos caracteres hexadecimales en mayúscula.

output: Una línea que contenga "True" o "False" según corresponda.

6. Contacto

Cualquier pregunta sobre la tarea, ya sean de enunciado, contenido o sobre aspectos administrativos deben comunicarse con los ayudantes creando issues en el Syllabus del Github del curso.

7. Anexo

Códigos base desafío 1.1 y 1.2

```c
#include <stdio.h>

int main() {
    unsigned int lista;
    int numero = 0;
    scanf("%u\n",&lista);
    //Completar Aquí

    //Fin
    printf("%i\n",numero);
    return 0;
}
```

Códigos base desafío 2.1 y 2.2

```c
#include <stdio.h>

int main() {
    unsigned int lista;
    int numero = 0;
    scanf("%u\n",&lista);
    //Completar Aquí

    //Fin
    printf("%i\n",numero);
    return 0;
}
```

Códigos base para los otros desafíos

```c
#include <stdio.h>

int main() {
    //Completar Aquí

    //Fin
    return 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.