Algoritmos y Estructuras de Datos I
Segundo Cuatrimestre 2020

Guía Práctica
EJERCICIOS DE TALLER

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

1. Testing

Ejercicio 1. `int puntaje(int n)`
Sofía está jugando un juego. Tiene una bolsa con bolitas y mete la mano para sacar un puñado. Gana puntos según la cantidad de bolitas con las siguientes reglas:

- Si la cantidad de bolitas es menor que 10, gana dos puntos por cada bolita que sacó. Si no, un punto por cada una.
- Además, si la cantidad de bolitas que sacó es múltiplo de 3, gana 10 puntos. Si no, pierde 10 puntos.

Dado el programa que calcula la cantidad de puntos qué ganó Sofía. Armar casos de test estructurales para el programa, los test deben cubrir todas las decisiones (branches) del código.

Ejercicio 2. Armar casos de test estructurales para el programa del ejercicio **5 de la Guía de laboratorio de Matrices y Tableros**: `int contarPicos(vector<vector<int>> m)` , los test deben cubrir todas las decisiones (branches) del código tanto en funciones principales como en auxiliares que hayan definido.

Ejercicio 3. `bool sandia(int porciones)`
Sara y Flor consiguieron un cantidad de porciones de sandía para el postre. Quisieran dividir las porciones en dos partes tales que cada una coma una cantidad par de porciones (no necesariamente la misma cantidad cada una). El objetivo es decidir si es posible realizar la división dada la cantidad de porciones.

1. Pensar y escribir los tests que crean necesarios antes de programar la solución. ¿Cuáles son las particiones de dominio?
2. Programar en C++ una solución al problema que pase los tests antes creados.
3. Extender (si es necesario) el conjunto de casos del ítem anterior para que cubra todas líneas de código.
4. Extender (si es necesario) el conjunto de casos del ítem anterior para que cubra todas las ramas de decisiones (branches) del programa.

Ejercicio 4. Armar casos de test **funcionales** para el programa del ejercicio **6-a de la Guía de Matrices y Tableros**: `tuple<tuple<int, int>, vector<tuple<int, int, int>>> aTriplas(vector<vector<int> > m)`. ¿Son estos tests resistentes a distintas implementaciones?

Ejercicio 5. `int sopaDeLetras(vector<vector<char>> m)`
Dada una sopa de letras m — matriz de caractéres con valores del abecedario en minúscula — se quiere encontrar la cantidad de veces que aparece la palabra *sopa* escrita tanto de izquierda a derecha como de arriba a abajo.

1. Pensar y escribir los tests que crean necesarios antes de programar la solución.
2. Programar en C++ una solución al problema que pase los tests antes creados.
3. Extender (si es necesario) el conjunto de casos del ítem anterior para que cubra todas líneas de código.
4. Extender (si es necesario) el conjunto de casos del ítem anterior para que cubra todas las ramas de decisiones (branches) del programa.

Ejercicio 6. *Filas Que Suman*
Programar las siguientes funciones y escribir sus respectivos casos de test.

a) `vector<int> filasQueSumanN(vector<vector<int>> m, int n)`, que dada una matriz de enteros y un numero devuelve los numeros de las filas que suman exactamente n

b) `vector<int> filasDondeAlgunosSumanN(vector<vector<int>> m, int n)`, que dada una matriz de enteros y un numero devuelve los numeros de las filas donde algun subconjunto de dicha fila sume exactamente n. Cada fila puede aparecer a lo sumo una vez en el resultado.