COMP 2401 -- Assignment #4

Due: Monday, November 24, 2014 at 9:00 PM

Goal

You will implement the *Memory Hunter* functions from Assignment #3 using a linked list to track the blocks of dynamically allocated memory, rather than an array.

Learning Objectives

- get familiar with more complex dynamic memory operations by managing a linked list

Instructions

1. Data structures

Modify the **`HeapType`** data type to store a singly linked list of **`BlockType`** structures, implemented as we saw in the "Advanced Linked Lists" section of the course notes. You must create at least one new data type to achieve this.

2. Memory Hunter functions

Implement the Memory Hunter functions from Assignment #3 to work with the **`heap`** variable declared in **`main`** as a linked list of blocks, rather than an array.

Notes:

- blocks must be kept in the order in which they are allocated, so you must add each block to the **end** of the list
- the **`main`** function will be identical to the one in Assignment #3
- the function prototypes will be identical to the ones in Assignment #3
- the output will be identical to Assignment #3, except for the "total heap usage" reported by valgrind

Constraints

- do **not** use any global variables
- compound data types **must** be passed by reference, not by value
- you must manage your memory! use valgrind to find and fix memory leaks
- you must reuse functions everywhere possible
- your program must be thoroughly commented

Submission

You will submit in *cuLearn*, before the due date and time, one **`tar`** file that includes all the following:

- all source code
- a Makefile
- a readme file, which must include:
  - a preamble (program author, purpose, list of source/header/data files)
  - exact compilation command
  - launching and operating instructions

Grading

- **Marking breakdown:** The grading scheme is posted in cuLearn.
- **Deductions:**
  - Up to 50 marks for any of the following:
    - the code does not compile using gcc in the VM provided for the course
    - unauthorized changes have been made to the code provided for you, where applicable
    - code cannot be tested because it doesn't run
  - Up to 20 marks for any of the following:
    - your program is not broken down into multiple reusable, modular functions
    - your program uses global variables (unless otherwise explicitly permitted)
    - your program passes compound data types by value
    - the Makefile or the readme file is missing or incomplete
  - Up to 10 marks for missing comments or other bad style (indentation, identifier names, etc.)
- **Bonus marks:**
  - Up to 5 extra marks are available for fun and creative additional features