**Program 7 - Building a Heap**

*Due Date: 11:00 pm, May 1, 2020*

*All programs will be tested on remote system.*

# Part 1: Heap Initialization

* Create an array based **Min-Heap**. You should have a header file, heap.h, with the following:
  + Heap struct containing the following:
    - Data pointer to store an array of data object in heap structure.
    - Maximum capacity (follows the geometric expansion).
    - Current size.
    - Function pointers to store the addresses of push, pop and peek functions.
  + Function declarations for all your heap functions to be implemented in heap.c
* You should have a source file, heap.c, that implements:
  + Heap \* initHeap(Data \* d, int size);
    - The function should take a data array and its size.
    - Initializes the Heap struct, sets its fields accordingly and returns a pointer with the internal array heapified.
    - Heap order will be based on the priority field of the Data struct not the actual value.

# Part 2: Heap Operations

* You will need to implement the following functions in your heap.c file.
* Push:
  + Insert a new element while maintaining the heap order based on the priority values.
  + void push (Heap \* heap, Data value)
* Pop:
  + Removes and returns the Min element (minimum priority) while maintaining the heap order.
  + Data pop (Heap \* heap)
* Peek:
  + Retrieves the value of the Min element (minimum priority).
  + Data peek (Heap \* heap)
* You will also need the following helper functions for your heap:
  + siftDown
  + heapify

# Part 3: Testing Your Heap

# Have a loop that create an array of 100 Data objects with random priorities between 1-100 and random string values. Once the data Array is created, you should use it to create a heap using your initHeap() function.

* Have a second loop that pushes 1000 new Data objects with random priorities between 1-1000 and random string values.
* Have a third loop that first peeks and then pops all elements of the heap by continuously removing the priority value at each iteration.
  + Use the [assert](http://man7.org/linux/man-pages/man3/assert.3.html) function to test your peek and pop functions to make sure the priority value you remove is greater than or equal to the previous priority value that was removed in the previous iteration.

Part 4 - Submission

* Required code organization:
  + program7.c:
    - Contains the driver code with three loops which test all the functionality of your program (Part 3).
  + data.h:
    - Data struct:
      * int priority
      * char [] value
  + heap.c/h:
    - Heap struct
      * data (Data \*)
      * current\_size (int)
      * maximum\_capacity (int)
    - Heap \* initHeap(Data \* d);
    - void siftDown(Heap \* heap, int index);
    - void \* heapify(Heap \* heap);
    - Data pop (Heap \* heap);
    - void push (Heap \* heap, Data value)
    - Data peek (Heap \* heap)
    - makefile
      * *You must have the following labels in your makefile:*
        + all - to compile all your code to an executable called ‘**program7**’ (no extension). **Do not run**.
        + run - to compile **if necessary** and run.
        + checkmem - to compile and run with valgrind
        + clean - to remove all executables and object files
* Use below command to tar.gz your assignment folder.
  + tar -czvf <yourid>\_program7.tar.gz <assignment folder>

**Do not include your object files or executables**

* Upload the archive to Blackboard under Program 7.

# Grading Guidelines

Total: 25 points

## Part 1,2,3:

* + To get credit, your Heap, and your test loops must all work as expected.

## Style Guidelines and Memory Leaks

* + You will lose significant points for the following:
    - Makefile does not have requested format and labels (-5)
    - Does not pass Valgrind Tests (-10)
    - Does not follow requested program structure and submission format (-5)
    - Does not follow [formatting guidelines](https://drive.google.com/open?id=1PyRZpBay-PZ8CzQRDAaz2Oke0K0T1rIBHMKNzZWjgbI) (-5)