CSE310 Project 02: Implementing the Heap Data Structure
OUT: Thursday, 03/16/2017
DUE: Thursday, 03/30/2017 (by 11:59pm)

In this programming project, you will be implementing the data structure min-heap. You should use the C++ programming language, not any other programming language. Also, your program should be based on the g++ compiler on general.asu.edu. **All programs will be compiled and graded on general.asu.edu, a Linux based machine**. If you program does not work on that machine, you will receive no credit for this assignment. You will need to submit it electronically at the blackboard, in one zip file, named CSE310-P02-Lname-Fname, where Lname is your last name and Fname is your first name. The zip file should contain a set of files that are absolutely necessary to compile and execute your program. If you program does not compile on general.asu.edu, you will receive 0 on this project.

You need to define the following data types.

- ELEMENT is a data type that contains a field named key, which is of type int. In later assignments, you will have to add on other fields to ELEMENT, without having to change the functions. Note that ELEMENT should not be of type int.
- HEAP is a data type that contains three fields named capacity (of type int), size (of type int), and H (an array of type ELEMENT with index ranging from 0 to capacity).

The functions that you are required to implement are

- Initialize(n) which returns an object of type HEAP with capacity n and size 0.
- BuildHeap(heap, A), where heap is a HEAP object and A is an array of type ELEMENT. This function copies the elements in A into `heap->H` and uses the linear time build heap algorithm to obtain a heap of size size(A).
- Insert(heap, k) which inserts an element with key equal to k into the min-heap heap.
- DeleteMin(heap) which deletes the element with minimum key and returns it to the caller.
- DecreaseKey(heap, element, value) which decreases the key field of element to value, if the latter is not larger than the former. Note that you have make necessary adjustment to make sure that heap order is maintained.
- printHeap(heap) which prints out the heap information, including capacity, size, and the key fields of the elements in the array with index going from 1 to size.

You should implement a main function which takes the following commands from the key-board:

- **S**
- **C n**
- **R**
- **W**
- **I k**
- **D**
- **K i v**

On reading **S**, the program stops.
On reading **C n**, the program creates an empty heap with capacity equal to **n**, and waits for the next command.
On reading **R**, the program reads in the array A from file HEAPinput.txt, calls the linear time build heap algorithm to build the heap based on A, and waits for the next command.
On reading **W**, the program writes the current heap information to the screen, and waits for the next command.
On reading **I k**, the program inserts an element with key equal to **k** into the current heap, and waits for the next command.
On reading **D**, the program deletes the minimum element from the heap and prints the key field of the deleted element on the screen, it waits for the next command.
On reading **K i v**, the program decreases the key of element with index i to the new value v, provided that the new value is not larger than the previous value.

The file HEAPinput.txt is a text file. The first line of the file contains an integer n, which indicates the number of array elements. The next n lines contains n integers, one integer per line. These integers are the key values of the n array elements, from the first element to the nth element.

You should use modular design. At the minimum, you should have

- the main program as `main.cpp` and the corresponding `main.h`;
- the heap functions `heap.cpp` and the corresponding `heap.h`;

- various utility functions `util.cpp` and the corresponding `util.h`.

You should also provide a `Makefile` which compile the files into an executable file named `run`.

Grading policies: (Sample test cases will be posted soon.)

(10 pts) Documentation: You should provide sufficient comment about the variables and algorithms. You also need to provide a README file describing which language you are using. You will also need to provide a Makefile. The executable file should be named `run`.

(10 pts) Data types: You should define the required data types.

(10 pts) Initialize

(10 pts) BuildHeap

(10 pts) Insert

(10 pts) DeleteMin

(10 pts) DecreaseKey

(10 pts) printHeap

(10 pts) modular design

(10 pts) Makefile

Above all, you need to write a working program to correctly parse the commands specified in the project. Without this, your program will not be graded.