CSCI 340 **Computer Assignment 7** **Spring 2017**
(10 points)

Heaps and Heapsort

For this computer assignment, you are to write a C++ program to sort numbers using the *heapsort* technique. Your program first builds a heap structure for the numbers. Then, it retrieves these numbers from the heap in a certain order and prints them out on `stdout`.

The source file `assignment7.cc` is partially implemented and contains the complete implementation of the `main` function. It is available at `/home/turing/mhou/public/csci340spring2017`. Add and implement the following functions in this source file.

- `void build_heap ( vector < int >& v, int heap_size, bool (*compar)(int, int) )`: This function constructs a heap with `heap_size` elements in the vector `v`. Pay attention that elements start at position 1 (position 0 is wasted and ignored) in the vector. `compar` is a function pointer (predicate) to compare two integers. `build_heap` will invoke `heapify` specified below.

- `void heapify( vector < int >& v, int heap_size, int r, bool (*compar)(int, int) )`: This function "heapifies" a tree at the root position `r`, assuming `r`'s two sub-trees are already heaps. `heap_size` specifies the size of the whole heap contained by the vector (the heap starts at position 1 of the vector). This function uses the function pointer `compar` to compare two elements. This function can be implemented recursively.

- `bool less_than ( int e1, int e2 )`: This function compares two integers and returns `true` if `e1` is less than `e2`. Otherwise it returns `false`. When this function is used as predicate in `build_heap`, a min heap will be constructed.

- `bool greater_than ( int e1, int e2 )`: This function compares two integers and returns `true` if `e1` is greater than `e2`. Otherwise it returns `false`. When this function is used as predicate in `build_heap`, a max heap will be constructed.

- `void heap_sort ( vector < int >& v, int heap_size, bool (*compar)(int, int) )`: This function implement the *heap sort* algorithm. At beginning the vector `v` contains a heap. At the end of this function, vector `v` contains sorted elements. Similar to `build_heap`, there is a predicate in the parameter list to specify how to compare two elements. If `less_than` is passed in as argument here, the results are in ascending order. If `greater_than` is used, the results are in descending order. `heap_sort` will invoke `extract_heap` specified below. You can use the STL algorithm `reverse` if necessary.

- `int extract_heap ( vector < int >& v, int& heap_size, bool (*compar)(int, int) )`: This function extracts the root of the heap recorded in `v`, fills the root with the last element of the current heap, updates `heap_size`, "heapifies" at the root, and returns the old root value. This function will invoke `heapify` specified above.

- `void print_vector ( vector < int >& v, int pos, int size )`: This function displays `size` number of elements contained in vector `v` starting at position `pos`. It shows 8 elements per line. Each item occupies 5 spaces.

Programming Notes:

- Please implement the algorithms to build the heap and sort by heapsort. Do not invoke the STL algorithms `make_heap` or `heap_sort`. If you do, you will get 0 points for this assignment.

- Please pay extra attention that in this assignment vectors' elements start at position 1 (instead of position 0) unless otherwise specified.

- Include any necessary headers and add necessary global constants.

- You are not allowed to use any I/O functions from the C library, such as `scanf` or `printf.` Instead, use the I/O functions from the C++ library, such as `cin` or `cout.`

- In the final version of your assignment, you are not supposed to change existing code, including the main method, provided to you in the original source file `assginment7.cc`.

- To compile the source file, execute "`g++ -Wall assignment7.cc –o assignment7.exe`". This will create the executable file `assignment7.exe.` To test your program, execute "./`assignment7.exe &> assignment7.out`", which will put the output (including any error messages) in file `assignment7.out`. You can find the correct output of this program in file `assignment7.out` in the directory shown in the last page.

- Add documentation to your source file.

- Prepare your Makefile so that the TA only needs to invoke the command "`make`" to compile your source file and produce the executable file `assignment7.exe`. Make sure you use exactly the same file names specified here, i.e. `assignment7.cc` and `assignment7.exe`, in your `Makefile`. Otherwise your submission will get 0 points.

- When your program is ready, submit your source file `assignment7.cc` and `Makefile` to your TA by following the Assignment Submission Instructions.