Programming II | **Heap Sort**
CMSC 2613 | **Project p10**

Assignment: Exercise a heap Abstract Data Type (ADT) by inserting integers read from file **i10.dat** into the heap. Remove and print the minimum element in the heap, repeatedly, until the heap is empty. Print results in file **o10.dat** as shown in Figure 2.

Prohibition: Use of the C++ Standard Template Library is prohibited in the implementation of this project.

Program Files: Project **10** consists of files **p10.cpp**, **Heap10.h**, **Heap10.cpp**, and **p10make.** Project file names are exactly as given. Failure to employ the foregoing names will result in a score of ***zero*** (***0***) for this project

Project files must be stored in the **root directory of your student account**. Failure to store project files in the root directory of your student account will result in a score of ***zero*** (***0***) for this project.

File	**Description**
p10.cpp	File **p10.cpp** contains functions which process command line arguments and direct heap operations.
Heap10.h	File **Heap10.h** contains the definition of class *Heap*. Class *Heap* implements a heap containing integers. The heap is stored in a dynamically allocated array. Element zero contains a sentinel.
Heap10.cpp	File **Heap10.cpp** contains the implementation of member functions of class *Heap*.
p10make	File **p10make** contains instructions for creating executable file **p10**. Instructions in file **p10make** are accepted by the UNIX utility ***make*.**

Command Line: Project **10** can be invoked with zero, one, or two program parameters. The first program parameter is the input file name. The second parameter is the output file name. Sample command lines together with corresponding actions by program **p10** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

```
$ p10
Enter the input file name: i10.dat
Enter the output file name: o10.dat

$ p10 i10.dat
Enter the output file name: o10.dat

$ p10 i10.dat o10.dat
```

Input files: The input file contains a list of integers as shown in Figure 1.

Output files: Print the heap after all integers have been inserted into the heap. Print the heap graphically. Perform an inorder traversal on the elements in the heap printing each element according to its depth into the heap. Each level of the heap is indented five spaces from the higher level.

Print the integers in ascending order.

Figure 1.
Input file format:

```
14 29 -3 64 13 56 0 72
41 92 29 46 31 65 10
```

```
Graphical Representation
                                                            72
                                        41
                                                            64
                    13
                                                            92
                                        29
                                                            29
-3
                                                            56
                                        31
                                                            46
                    0
                                                            65
                                        10
                                                            14
Sorted
-3 0 10 13 14 29 29 31 41 46 56 64 65 72 92
```

Figure 2. Output file format: