UAH Electrical and Computer Engineering Department
Project05 (4 points) Version 1.0

Submit Your Solution Using Canvas by 5:00 PM, Friday April 6, 2018

<Project05 Description>

For this project, you will write a ***multi-file*** C++ program that implements and tests a **Smart Heap** ***class template***. In this version of the **Heap** datatype you will store members of the heap in a one-dimensional array that is dynamically allocated by the class constructor. The first two input files allow you to test your class template using integers. The last two input files allow you to test your class template using objects of type **Task**.

Concepts covered by this assignment: class templates, operator overloading, friend functions

➔ ***This project requires you to write two files and submit both files for grading*** ←

List of files that must be submitted: (see CANVAS for detailed submission instructions)

heap.h **(provided – do not submit)**

Specification file for the **Heap** class as supplied by the instructor

heap.cpp **(➔ you must write and submit this file ←)**

This file will implement ***all member functions*** of the **Heap** class and any supporting functions required.

task.h **(provided – do not submit)**

Specification file for the **Task** class as supplied by the instructor. At end of this file, I have the preprocessor directive **#include "heap.cpp"**

task.cpp **(➔ you must write and submit this file ←)**

This file will implement ***member functions*** of the **Task** class and any supporting functions required. Objects of type **Task** are placed in the heap based on the integer attribute value called **taskPriority**. Each task also has a string attribute representing the **taskName**.

main.cpp **(provided – do not submit)**

Contain your ***main*** function that serves as a test driver for the **Heap** class.

This file includes preprocessor directive **#include "heap.h"**

makefile **(provided – do not submit)**
this file provides all required dependencies and commands for the Linux ***make*** utility to compile and link your program.

➔ ***Detailed submission instructions appear on the Project05 dropbox on CANVAS*** ←

➔ ***Failure to follow submission directions will result in zero (0) credit on this assignment*** ←

<Project05 Directions for Executing the Sample Solution>

You may run the sample solution by typing the following at a terminal window command prompt:

/home/work/cpe212/project05/p05 inputfile

Note: inputfile is called a Command-Line Argument

inputfile must be the name of a text file in your current working directory

All output is written to the standard output device (monitor).

The grade preview script for Project05 is available at

/home/work/cpe212data/project05/preview05.bash

<Project05 Constraints>

On Project05, you may only use the following include files:

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <cmath>
#include <new>
#include <cstddef>
#include "heap.h"
```

You are not allowed to utilize Global Variables!!

➔ ***Recursive Function Calls are permitted!!*** ➔

Failure to follow these directions will result in zero (0) credit on this assignment.

Once you are satisfied with your solution, submit your program files via CANVAS.

NOTE: make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.

<Project05 heap.cpp Description>

Your goal is to EXACTLY match the output of your program to that of the sample solution.

'c' ➔ Constructor dynamically allocates a Heap object

'+' ➔ Insert the data value that appears after the '+' onto the Heap

'-' ➔ DeleteMax deletes the maximum value from the Heap

'p' ➔ Print writes the contents of Heap to stdout

's' ➔ Size returns the number of elements currently stored in the Heap object

'e' ➔ MakeEmpty empties the heap leaving it ready to accept new data

'm' ➔ Capacity returns the maximum number of elements currently allocated to the Heap object

'd' ➔ Destructor deallocates the Heap object

<Project05 Heap Class Description>

The **Heap** class uses a ***dynamically allocated*** one dimensional array to store heap values. When this array is filled to capacity, the next insert causes creation of a new array with twice the capacity. Values from the smaller array are copied from the small array into the new larger array, and then the smaller array is deallocated to prevent a memory leak.

The data type of the value stored in each node is determined by the client code **main.cpp**.

➔ **Inline functions are not allowed in this course and will result in no credit (0)** ➔

[Note: An inline function is a function whose definition appears within the class declaration]

The **Heap** class contains three **private** attributes, one **private** function, and twelve **public** member functions

Attribute Name	Data Type	Purpose
ptr	**ItemType***	Stores **address** of smart heap integer array
maxsize	**int**	Current maximum heap size (maximum number of values that can currently be stored in the existing heap array)
num	**int**	Number of values currently stored in heap

Function Name	Return Type	Purpose
Heap	**N/A**	Default constructor that creates a **Heap** object with default MAXSIZE elements
Insert	**void**	Adds a new integer in the **Heap** assuming that it is not already full. If **Heap** capacity has been reached, it attempts to double the current capacity. If capacity cannot be doubled, it throws FullHeap
ReheapUp	**void**	Restores heap order property leaf to root
ReheapDown	**void**	Restores heap order property root to leaf
DeleteMax	**ItemType**	Removes and returns the maximum value from **Heap.** If heap is empty, it throws EmptyHeap
IsEmpty	**bool**	Returns **true** if Heap is empty, **false** otherwise
IsFull	**bool**	Returns **true** if Heap is full to capacity, **false** otherwise
MakeEmpty	**void**	Empties Heap array
Size	**int**	Returns total number of data values stored in Heap
Capacity	**int**	Returns current number of elements allocated to Heap array
Print	**void**	Writes **Heap** array contents to **stdout**
~Heap	**N/A**	Deallocates **Heap** array

➔ **See the file heap.h for additional details regarding the function interfaces** ➔

<Project05 Hints>

Sample codes for the array implementation of the Heap container appear in Chapter 9 of the official course textbook,

C++ Plus Data Structures, Sixth edition by Nell Dale, Chip Weems, and Tim Richards