Using a Heap to Create Sorted Lists of Dates

Out: 1/31
Due: 2/13 by 11:59 PM

Learning Objectives

- Implementing a Binary Heap Data Structure.
- Manipulating an extensible array
- Generating a Sorted List Using a Binary Heap

In this project you will complete the implementation a parametric heap data structure.

Definition 1. Recall that a **heap** is a complete binary tree in which each entry has a key which is less (greater) than or equal to the key of its parent when the heap is a *max-heap* (*min-heap*).

The heap property enables the data structure to be used in generating an ordered list. If a series of insertions is performed on an initially empty heap followed by a series of deletions until the heap is empty, the items are removed from the heap in descending (ascending) order when the heap is a max-heap (min-heap). We can implement a user-defined comparator so that the entries in the heap are inserted and removed in a desired order whether the underlying implementation of the data structure is a max-heap or min-heap. In this project you will generate three ordered lists of dates using a heap data structure.

I have provided starter code for both the *Heap* and *Date* implementations. You will complete the *Heap* implementation. The *Date* is already implemented. See the starter code on Moodle for additional details. Do not modify the code except where indicated.

The DatesHeapifier program

You will write a program called *DatesHeapifier* consisting of only one method (function), the *main*. First, the program reads a list of dates, in the format *mm/dd/yyyy*, from a text file whose name is entered as a command line argument, inserts the dates into an initially empty heap and then performs a series of deletions to empty the heap. The program will then generate a list of the dates in the natural increasing order (*+year+month+day*) by displaying the dates as they are removed from the heap. The primary key is the year, secondary key is the month and tertiary key is the day and the keys are in ascending order for the first list. The list will be formatted as shown in Listing 1.

Listing 1: Dates in +year+month+day Order

```
Dates from <filename> in Ascending Order:
<Day of the week>, <Month name> <day>, <year>
       :           :         :           :    :
       :           :         :           :    :
```

Second, the program again reads the dates from the same text file, inserts them into the empty heap and performs a series of deletions to empty the heap. The program will then generate a second list of the dates in the natural decreasing order (*-year-month-day*) by displaying the dates as they are removed from the heap. The primary key is the year, secondary key is the month and tertiary key is the day and the keys are in descending order for the second list. The list will be formatted as shown in Listing 2.

Listing 2: Dates in -year-month-day Order

```
Dates from <filename> in Descending Order:
<Day of the week>, <Month name> <day>, <year>
:           :         :           :    :
:           :         :           :    :
```

Finally, the program again reads the dates from the same text file, inserts the corresponding 2019 dates for each date into the empty heap and performs a series of deletions to empty the heap. The program will then generate a third list of the dates in the natural increasing order (*+month+day*) by displaying the dates as they are removed from the heap. The primary key is the month and the secondary key is the day and the keys are in ascending order for the third list. Observe that you won't need a tertiary key since each date is in the same year (2019). The list will be formatted as shown in Listing 3.

Listing 3: Dates in +month+day Order

```
Dates from <filename> in Ascending Order and Weekday ↩
   in this Year:
<Day of the week>, <Month name> <day>
:          :        :          :
:          :        :          :
```

I have also provided a file *BundesHolidays.txt* containing a series of dates. Use this file to test your program. You may create additional text files to test your program. (The calendar generator at *http://www.dayoftheweek.org* may be used to verify the correctness of your program.)

Submitting Your Work

1. All source code files that you submit must have header Javadoc comments with the following:

   ```
   /**
    * Describe what the purpose of this file
    * Course: CS3102.01
    * Programming Project #: 1
    * Instructor: Dr. Duncan
    * @author Programmer(s)
    * @since 9999-99-99
    * @see list of files that this file references
    */
   ```

2. Verify that your code has no syntax error and that it is ISO C++ 11 or JDK 8 compliant prior to uploading it to the drop box on Moodle. Be sure to provide missing documentation, where applicable. Also, add your name after the @author tag when you augment the starter code that I have provided. Put the last date modified in the header comments for each source code file.

3. Locate your source files -
   (a) for Java programmers, **HeapAPI.java**, **Heap.java**, **DateAPI.java**, **Date.java** and **DatesHeapifier.java**
   (b) for C++ programmers, **Heap.h**, **Heap.cpp**, **Date.h**, **Date.cpp** and **DateHeapifier.cpp**

   - and enclose them along with the sample data file, **BundesHolidays.txt**, in a zip file, YOURPAWSID_proj01.zip, and submit your programming project for grading using the digital drop box on the course Moodle.