CS 3353 Data Structure and Algorithm Analysis I
Assignment 04: Full Marks 100
Due Date: 11/06/2022, 11:59 PM CT **End Date: 11/13/2022, 11:59 PM CT**

Read the following instructions before you proceed with the assignment:

- The assignment needs to be completed individually in Java.
- Make sure the program runs in CSX machine.
- Please go through the grading rubrics and submission instructions carefully to avoid losing points. There have been some changes (given in blue font) made in the submission method.

In this assignment, you will use implement heap using array. You need to use your own priority queue ADT and not use Priority Queue class from Collection frameworks.

This assignment has a bonus part. Please read the complete assignment, including the bonus part, before you start the assignment.

Background:

The standard queue works on the principle of *first-in*, *first-out* (*FIFO*) strategy i.e. the first element added to the collection will always be the first element to be removed from the collection. However, this strategy is often too simplistic and is not be applicable in many real-life activities. For example, in the medical field, you need to schedule patients based on some priority. The patient with more severe health issues should preempt other patients, even if they are waiting for an extended period. This is where priority queue comes into picture. In priority queue, elements are added to the collection in an arbitrary order, but the element with the highest priority will be the first to get removed. In this assignment, you will use priority queue to model the heart transplant waiting list.

Input File Description:

In this assignment, you will use the provided input file *inputFile.txt* to populate your data structure. The input file is a list of patients waiting for the heart transplant. The file consists of the following columns, separated by semicolon:

1. *first_name*: first name of the patient in the heart transplant waiting list
2. *last_name*: last name of the patient in the heart transplant waiting list
3. *address*: current geographic location of the patient's residence
4. *city*: city of the patient's residence
5. *county*: county location of the patient's residence
6. *state*: state of the patient's residence
7. *zip*: zip-code of the patient's residence
8. *phone1*: first preference contact number of the patient
9. *phone2*: second preference contact number of the patient
10. *email*: email address of the patient
11. *date_listed*: data the patient is listed on the waiting list at the given *UNOS_status*
12. *UNOS_status*: United Network of Organ Sharing (UNOS) status classification based on the patient condition.
13. *date_of_birth*: date of birth of the patient

Each line of the input file represents one patient's information and will be called *record* hereafter.
Disclaimer: All data appearing in this text file are fictitious and created randomly using Excel. Any resemblance to any real person is purely coincidental.

Implementation Requirement:

You need to implement priority queue in terms of binary heap using an array. A heap is a binary tree with two properties:

1. it is a complete binary tree i.e. it is full in all levels.
2. heap-order property: each node has a priority higher than or equal to its child's value. Each node has two priorities: *primary priority* (which is *UNOS_status*) and *secondary priority* (which is patient's age obtained from *date_of_birth*). You will always use *primary priority* first. Only if there is a tie between the record, you will use *secondary priority*.

How is the priority ranked?

Primary priority is obtained from *UNOS_status* column. There are four UNOS status classifications based on the condition*:

- *Status 1A* or *urgent need*: Requires intensive care hospitalization, life-support measures, certain cardiac supporting intravenous medications with a Swan-Ganz catheter, or mechanical-assist device(s).
- *Status 1B*: Dependent on intravenous medications or a mechanical-assist device – in the hospital or at home.
- *Status 2*: Stable on oral medications and able to wait at home.
- *Status 7* or *inactive list*: Inactive due to a change in condition – patients do not lose time they have already accrued.

*Source:https://www.pennmedicine.org/for-patients-and-visitors/find-a-program-or-service/transplant-institute/heart-transplant/heart-transplant-process/heart-waiting-list

The priority for *UNOS_status* is as follows: status 1A has the highest priority, which is then followed by Status 1B, then followed by Status 2 and finally Status 7. If there are patients with the same UNOS_status, then you need to use age to order the patient.

Secondary priority is patient's age which can be obtained from patient's *date_of_birth*. Patient with the lower age gets higher priority than the patient with the higher age.

Task:

You need to write a menu driven program with the following options:

1. insert
2. peek
3. nextPatient
4. removePatient
5. size
6. exit

Each of these options are described as the following five tasks:

1. insert:
   This method inserts *record* from the given input file into the priority queue. The insert method consists of two steps:
   i) store the new *record* at the next available location in the array.
   A new *record* is added at the very bottom of the heap. After the insertion of a new *record*, the heap-order property may be violated. To fulfill the heap-order requirement, you execute the next step.

   ii) restore the heap-order property.
   To fulfill the heap-order property, you need to move the *record* in its proper place in the binary tree.

   Note: If any record does not include value in the *UNOS_status* column, then your program should take a default value of *Status 7*.

   The sample output after this option should be as shown below:

   ```
   Input file is read successfully..
   ```

   Your insert option should only read data from the file once. First insert operation should read the data from the file. After that if the user chooses option 1, then it should get data from the console.

2. peek:
   This method returns (without removing) the *record* which has the highest priority. Given the heap ordering, the highest priority patient resides at the root of the binary tree.

   The sample output after this option should be as shown below:

   ```
   The patient detail with the highest priority is as follows:
   Patient's first name: Cammy
   Patient's last name: Albares
   Data of birth of the patient: 10/30/1973
   Address: 56 E Morehead St
   City: Laredo
   County: Webb
   State: TX
   Zip code: 78045
   Phone Number (1st Preference): 956-537-6195
   Phone Number (2nd Preference): 956-841-7216
   Email address: calbares@gmail.com
   UNOS Status: Status 1B
   Data listed on Status 1B: 12/7/2016
   ```

   Notice the red font text in the sample output.

3. nextPatient:
   This method removes the highest priority *record* from the priority queue. This method needs to follow the following steps:
   i) replace the root node with the last node (call it a node *w*) in the binary tree.
   ii) remove the node *w*.
   iii) restore the heap-order property.

   The sample output after this option should be as shown below:

```
The patient removed from the heap is as follows:
Patient's first name: Cammy
Patient's last name: Albares
Data of birth of the patient: 10/30/1973
Address: 56 E Morehead St
City: Laredo
County: Webb
State: TX
Zip code: 78045
Phone Number (1st Preference): 956-537-6195
Phone Number (2nd Preference): 956-841-7216
Email address: calbares@gmail.com
UNOS Status: Status 1B
Data listed on Status 1B: 12/7/2016
```

   Notice the red font text in the sample output.

4. removePatient:
   This method allows users to remove *record* from anywhere in the priority queue. While the nextPatient from option (3) only removes record at the root node, this method should remove record from anywhere in the queue.

   Only if the record is found, the output after this option should be as shown below:

```
Please enter the patient information to remove from the queue:
Please enter patient's first name: Cammy
Please enter patient's last name: Albares
Please enter patient's data of birth: 10/30/1973
Please enter patient's address: 56 E Morehead St
Please enter patient's city: Laredo
Please enter patient's county: Webb
Please enter patient's state: TX
Please enter patient's zip code: 78045
Please enter patient's phone number (1st Preference): 956-537-6195
Please enter patient's phone number (2nd Preference): 956-841-7216
Please enter patient's email address: calbares@gmail.com
Please update the UNOS Status: Status 1A

The requested patient's record has been removed from the queue.
```

All the user inputs are shown in blue font.

If the record is not found, then you need to display the output as follows:

```
Please enter the patient information to remove from the queue:
Please enter patient's first name: Cammy
Please enter patient's last name: Albares
Please enter patient's data of birth: 10/30/1973
Please enter patient's address: 56 E Morehead St
Please enter patient's city: Laredo
Please enter patient's county: Webb
Please enter patient's state: TX
Please enter patient's zip code: 78045
Please enter patient's phone number (1st Preference): 956-537-6195
Please enter patient's phone number (2nd Preference): 956-841-7216
Please enter patient's email address: calbares@gmail.com
Please update the UNOS Status: Status 1A

The requested patient's record is not found.
```

All the user inputs are shown in blue font.

5. size:
   This method returns the number of records in the queue i.e. the number of nodes in the binary tree.

   The sample output after this option should be as shown below:

```
Number of records in the database: 500
```

6. exit:
   Only this method should allow user to exit from the program. This implies that after completing all other options, the program should again display the option menu.

Bonus Task:

The marks you obtained from completing the bonus point will only be added to the assignment portion in the final grading. For example, if a student has a total score of 500 points (out of 500 points from all 5 assignments), then any additional point obtained from the bonus will not be added to the grade book. The bonus points from the assignment cannot be added to the exams or discussions.

Part A: You need to add an additional menu on this part:

6. updatePriority
   This method allows updating patient's *UNOS_status*. Depending upon the medical condition, the patient's *UNOS_status* can be changed to any of the four specified *UNOS_status*. After changing the *UNOS_status*, the heap-property may get violated. You need to restore it.

The output after this option should be as shown below:

```
Please enter the patient information to change UNOS status:
Please enter patient's first name: Cammy
Please enter patient's last name: Albares
Please enter patient's data of birth: 10/30/1973
Please enter patient's address: 56 E Morehead St
Please enter patient's city: Laredo
Please enter patient's county: Webb
Please enter patient's state: TX
Please enter patient's zip code: 78045
Please enter patient's phone number (1st Preference): 956-537-6195
Please enter patient's phone number (2nd Preference): 956-841-7216
Please enter patient's email address: calbares@gmail.com
Please update the UNOS Status: Status 1A

The following patient detail has been updated:
Patient's first name: Cammy
Patient's last name: Albares
Data of birth of the patient: 10/30/1973
Address: 56 E Morehead St
City: Laredo
County: Webb
State: TX
Zip code: 78045
Phone Number (1st Preference): 956-537-6195
Phone Number (2nd Preference): 956-841-7216
Email address: calbares@gmail.com
UNOS Status: Status 1A
Data listed on Status 1A: 12/7/2016
```

Notice the red font text in the sample output. All the user inputs are shown in blue font. Also notice that the date is automatically updated to the current date.

Please note that, this method is given option (6). If you intend to do bonus task, please change exit to option (7).

Part B: You need to keep record of all the changes made through option (6) i.e. updatePriority i.e. if you have changed the UNOS_status of a patient, then you need to keep record of all the past UNOS_status and the date_listed for medical history. Consider for example, the patient named *James* is added to the priority queue under the UNOS_status of *status 7* on 9/4/2015 and his record have been updated as follows:

- On 10/12/2015, UNOS_status changed to *status 2*.
- On 12/23/2015, UNOS_status changed to *status 1B*.
- On 01/30/2016, UNOS_status changed to *status 1A*.

then all these changes need to be included in his record and not just the final UNOS_status and the corresponding adjusted date. Please note that the UNOS_status can be changed from any level to any other level. It is left to the student how they will implement this additional requirement. But the solution must be space efficient. Please clearly specify your design approach for this part in a separate pdf file.
Note: The inclusion of option (4) i.e. removePatient and option (6) i.e. updatePriority makes your priority queue adaptive.

Grading Rubric:

Your assignment will be graded based on the following grading Rubrics:

• Design of the priority queue using binary heap as specified in the assignment		**[30 Points]**
• Successful completion of the tasks and the display of appropriate results		**[70 Points]**
○ Option (1): insert ○ Option (2): peek ○ Option (3): nextPatient ○ Option (4): removePatient ○ Option (5): size ○ Option (6): exit	[18 Points] [7 Points] [15 Points] [18 Points] [7 Points] [5 Points]	

Grading for Bonus part:

• Successful implementation of Part A	**[20 Points]**
• Successful and efficient implementation of Part B ○ Make sure you include your design approach in a separate pdf file	**[10 Points]**

Tips to avoid any point deduction:

- Failure to follow standard programming practices will lead to points deduction, even if the program is running correctly, like
  - No comments on code
  - Not writing meaningful identifiers
- If the program does not compile successfully on the CSX machine, then 20 points will be automatically deducted.

Submission Guidelines:

- All the files should be submitted through Canvas. Assignment submitted through email will not be accepted. CSX machine is only used to run the program and not for assignment submission.

- Your assignment will be checked in the CSX machine. You need to include readMe.txt file that shows:
  - how to run your program.
  - which csx server has been tested on
  - any assumption you made on the assignment.

- If you decide to do the bonus tasks, then please include your model in a separate pdf file. It needs to be computer typed and not handwritten. Name this file as:

  assignment04_lastName_firstName_bonus.pdf

- All the java files need to have the following format:

  assignment04_lastName_firstName_<appropriate_file_name>.java

  where, use meaningful file name in the place of appropriate_file_name.

- In addition to .java file, all the codes should be submitted in .pdf format as well. Please copy the codes in the text form. Do not screenshot or include any image of the code in the pdf. This is used to check for plagiarism.

- Please include .java and .txt files inside a zip file. Please do not include pdf files inside the zip. They need to be submitted as separate files on the same submission.

- If you have multiple submissions, then make sure you submit all the files. Canvas displays the files associated with the latest submission and not from the previous submissions. So, we will not be referring to your previous submissions to get other files. So, your latest submission needs to include all the necessary files.

- The main driver file needs to have the following header information:
  - Author Name:
  - Email: <Use only official email address>
  - Date:
  - Program Description:

- Use comments throughout your program.
  - Each class and the method should be properly commented:
    - description of arguments and return data
    - description of method