CMPE 224
Fall 2021
Programming Homework 1

This assignment is due by 23:55 on Sunday, 31 October 2021.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE224-HW1-OfficeHour1: 17 Oct, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/96794078524
   - CMPE224-HW1-OfficeHour2: 22 Oct, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/96351358164

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

In this part, you must implement your own data structure by taking inspiration from your textbook and use it to help to solve problem. You are not allowed to use any external library or .jar file.

Assume that you want to create a basic online dictionary application. In this application you need to store all distinct words and their number of occurrences. One of the most common way to implement the dictionary is using hash table. In this assignment, you are expected to create hash table to store distinct words in given argument. Your initial table size should be 16, and you should use the hashCode method we talked about in class. For the string "world", the hash code would be 113318802. Since 113318802 % 16 = 2, the string would be added to 2th spot (0-based, of course) of the hash table.

First, begin implementing a hash table class in Java. You may want to start with the Hash class given in the book and modify it according to the homework.

PART I

In this part you should use two different collision resolution strategies while implementing the hash table which are **linear probing** and **separate chaining**.

For linear probing implementation, if your hash table becomes half full, you should expand your table size by doubling. For example, your first table size is 16 and it should be 32 after table is half full and then 64, 128… Also, before the inserting the new word, do not forget relocating all elements in the hash table after expanding your table. This means that you should use the given argument that contains the words and recalculate hash value for new table size, i.e., take mod with new table size.

For separate chaining implementation, if the number of items in the table is above a threshold (N/M >= 8), then you should resize the table as described in class.

After implementing hash table, you should create a Java application that reads the argument containing the words as a command-line argument. Each word is separated with space character. If the argument is not specified, display an error message, and terminate the program.

If you're able to read the argument, read it one word at a time. Keep track of all the words that you encounter and the number of occurrences of each. Remember, the dictionary should only contain distinct words, so you will need to modify the node definition so that it has a field that records the number of times the words have been seen, and only add a new word when you encounter a word that haven't been seen before.

Finally, you should print following output:

- Final table size for hash table with linear probing and separate chaining.
- Top 3 most used words, their indexes in hash table with linear probing,
  their node indexes in separate chaining, see Figure 1, and number of occurrences in hash table.

Figure 1: Hash table with separate chaining example. In this hash table, node index for 50,85 and 92 are 0,1 and 2, respectively.

Here is the `sample argument`:

```
world world world world world world dog cat car car dog cat
world window windows windo window ted cmpe 242 ted ted
accommodate accommodate synonyms accommodating accommodating
accommodating accommodating synonyms accommodation
accommodation synonyms accordionist accordion pleats
accoucheur's hand accoucheuse according as accordionist synonyms
synonyms
```

The output for the above input is as follows. Please check your program with this input as well as the others that you will create. Please note that we may use other input when grading your assignments.

```
% java Test sampleinput1

Final table sizes for linear probing and separate chaining are 64 and
16.

Top 3 most used words, their indexes for linear probing, their node
indexes for separate chaining and their number of occurrences:

world 18 0 7
synonyms 24 0 5
accommodating 44 1 4
```

PART II

In this part, you are expected to compare linear probing and separate chaining performance in terms of time and memory. You should report each method running time and used memory. Also, you should discuss the reason for performance differences between these two methods and report your observations according to different inputs.

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.

- You can test your Java source files on available Moodle VPL environment to ensure your solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input.

- A **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%30): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation, Functionality, Performance Comparison (%40): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section. Also, you should add your performance comparison, part II, here.

Testing (%15): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%10): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?
-

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:55 on Sunday, October 31th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-------------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-----------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
```

```
// Postcondition: The value of the variable is set.
//-------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TAs, Bedrettin Çetinkaya, Deniz Merve Gündüz and İbrahim İleri. Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructors Tolga Çapın and Ulaş Güleç for help.