# INFS-519 - Assignment 3: Hashing Analysis

Assignment Given: 11/09/2017
Assignment Due: 11/16/2017, 7:30 pm

In this assignment you are to create Java code to build a hash table using information about students as input. The input will consist of a stream of up to 95 transactions and data records, each having the following structure:

| Field Description | Type | Size |
|---|---|---|
| transaction code | alphabetic | 1 character<br>"I" = insert<br>"F" = find<br>"R" = remove/delete |
| <placeholder blank> | alphabetic | 1 character blank |
| surname | alphabetic | 15 characters |
| given name | alphabetic | 15 characters |
| year of birth | numeric | 4 digits |
| gender | Alphabetic | 1 character |
| major | alphabetic | 6 characters |

The hash table to be built consists of 100 entries, with keys ranging from 1 to 100. To build the table you will have to create a hash function to generate the keys using the information in the records. The function will generate a record key for the data and insert it and the data into the table. While the table is larger than the number of input records, collisions are likely and will have to be dealt with. A hash function should provide a good abstract representation of the data and minimize collisions.

In creating and coding the program:

1. The input is a single transaction text file, with up to 95 records as described above. A test file will be made available on Blackboard.
2. The transaction code describes what to do with the record – insert, find, or remove
3. Insert means hash the record and place it in the table
4. Find means hash the record and search the table
5. Remove means hash the record and delete the record from the table
6. Choose your own hash function. Using one presented in Weiss or elsewhere is permitted. Provide attribution if appropriate.
7. The hash table should be implemented as a 100 element array consisting of the generated key (1 to 100) and the associated data.
8. Collisions are to be reconciled using one of the methods discussed in class
9. As each transaction is processed, print a copy of the transaction to output, along with what was accomplished (record inserted, found, etc.), and note whether a collision was encountered
10. Print a copy of the hash table to output at the end of execution

**Measuring and Analysis Report**

You are to submit a short report (1-2 pages max) that describes your hash function and how you handle collisions. Also include how many collisions were encountered while processing the transaction stream, how many extra table slots needed to be examined when a collision occurred, and the overall load factor of the table at various stages of the run, including the end.

**What and where to submit:**

All items are to be submitted through Blackboard in a single zipped directory named "PA4-<username>HT.zip".

Example: PA4-jdoeHT.zip. Name the code file "ExHash" (without the quotes). Do not make your code part of a package. The unzip must unzip to a single directory that, in turn, has your code. Include:

1. The Java code used, with a comment block that includes your name, date, "INFS-519 Assignment 4", appropriate code attributions (if any), and a short description of the code's input, processing, and outputs. All code needed to execute from the command line must be included.
2. A pdf file that includes a log or screen shot of the output.
3. A pdf file of the 1-2 page report

**How the assignment will be assessed**

The Java code will be visually inspected and executed from command line using a set of test transactions from a .txt file. An example file for you to use in debugging will be provided on Blackboard. That file does not necessarily contain all possible combinations of expressions or errors. Your program must read the input file as an argument and open it for processing. The GTA will not edit your code to hard code a file name. Programs correctly echoing the input and processing the transaction stream without premature termination will be given full credit.

| Item | Assessment Description | Max Value |
|---|---|---|
| Java Code | All code submitted | 45 |
| | Code is readable, commented, attributed when necessary | 5 |
| Execution Output | Code executes against the transaction file error-free, produces the transaction echo, and displays the hash table at the end of the run. GTA assigns partial credit for less than 100% correct output, using his judgment. | 25 |
| Written report | A 1-2 page report describing the hash function and collision handling strategy is included. Including a few examples of hash input and output is encouraged. The report should be brief and readable, and address the particular points requested. GTA may assign partial credit. | 25 |
| **Total** | | **100** |

**Extra Credit:** As with the other assignments, for the class being held the night the assignment is due (11/16), I would like a few students to volunteer to present their code and results to the class. This would be at most a 10 minute presentation that includes (1) a code walk-through, (2) discussing results, and (3) Q&A from the class. For this presentation, concentrate on what your particular hash function does and how it handles collisions. Show results from running against the provided transaction file. Contact me separately if interested. Priority will be given to those who haven't presented before. The number of presentations is limited to five. Extra credit of +5% will be added to your cumulative assignment score (i.e. if the collective grade for all your assignments is 85% for the semester, this increases it to 90%).