**Team Project Report 4**

**Team name:** Program Rangers

**Project number:** 1

|  |  |  |
| --- | --- | --- |
| **Area** | **Programmer** | **Status/Comments** |
| **Unit 1: Team Lead** | Justin McKibben | Program mostly complete; all functions integrated and **95%** operational |
| **Unit 2: BST algorithms** | Evan Chen | BST being populated correctly. Tree form display close to complete but still needs tweaking.  **90%** |
| **Unit 3: Hashed array algorithms** | Shahar Marom | Hashing **100%** complete; printEfficiency **100%** operational |
| **Unit 4: User interface algorithms** | Justin McKibben | **100%**; User interface fully functional; properly calls corresponding functions |
| **Unit 5: File I/O algorithms** | David Lindskog | All data populated in data structures properly; hashed array and BST fully functional; some output functions still pending to be implemented; **90%** |
| **Unit 6: Extra structure algorithms** | Shahar Marom | compareDates - fully functional and properly sorts <string> dates  trendingWords - to be implemented  highestRetweet - to be implemented |
| **Unit 7: Rehashing algorithms (team of 5 only)** | N/A | N/A |
| **Test Plan: Options and data so that anyone could use it to demonstrate the project.**  **Presentation Outline: Activity, duration, etc** | Justin McKibben | Test plan **100%** finished; Presentation outline pending |
| **Project Documentation: Introduction, Data structure diagram, Structure Chart, Explanation of collision resolution**  **Program documentation: Source and header files for each programmer, Demonstration test plan** | Introduction - Justin McKibben  Data Structure Diagram - Evan Chen  Structure Chart - David Lindskog  Collision Resolution - Shahar Marom  Program Doc - Justin McKibben  Test Plan - Justin McKibben | Roughly **60%** done with all documentation; Much still needs to be updated |

**Presentation Outline:**

|  |  |  |
| --- | --- | --- |
| **Activity** | Duration 25 minutes max | **Team Member** |
| **Prepare for the presentation** | **2 min.** | All |
| **Introduce team members, brief description of the application** | **3 min** | Justin McKibben |
| **Data structure diagram** | **2 min** | Justin McKibben |
| **Hashing and Collision** | **5 min.** | Shahar Marom |
| **Run the program (using the Demonstration Test Plan)** | **10 min.** | Evan Chen - demo  David Lindskog - explain |
| **Questions/Answers/Conclusion** | **3 min.** | All |

**Test Plan:**

1. displayHash - List in hash sequence, only username
2. displayInOrder - List in key sequence (abc order by username), includes all tweets
3. printEfficiency - Print efficiency of hashed array (after data is imported)
4. trendingWords - Print most trending words (NOT DOING THIS ANYMORE)
5. highestRetweet - Print most popular tweet (according to retweets)
6. displayBST - Print the BST
7. newData - Add new data (to be collided with data A)
8. deleteData - Delete data A
   1. displayOne - Display new data
   2. displayOne - Display data A (unfound)
   3. printEfficiency - Print efficiency (show the numbers have changed) // Should we update the efficiency of the hashedArray everytime the data has changed?
9. Show Undo features
   1. deleteData - Delete data B
   2. deleteData - Delete data C
   3. printEfficiency - Print efficiency
   4. displayOne - Display data B (unfound)
   5. displayOne - Display data C (unfound)
   6. undo - Undo
   7. undo - Display data C
   8. newData - Add data B
   9. displayOne - Display data B // Show that it was added
   10. undo - Undo
   11. displayOne - Display data B (unfound) // Does this mean we need to implement an undo for add?
10. outputToFile - Write data to file (output is like input, including tweets)
    1. Show outputted data in file (open in textpad/notepad)
    2. undo - Undo (print error since stack should be empty)
11. Close app; run again importing the data from the previously output file

Data A (required to collide with data B): *We need to find a user that collides with another user without the same name*

Data B (required to collide with data A): *We need to find a user that collides with another user without the same name*

Data C: *We need to find an interesting tweet to use to entertain the class* ***(yoda)***

**TODO (personal use only)**

* freeData
* update undo to include add
* Update newData/deleteData to update collisions and load factor
  + We need to modify insertHash to take in a third int parameter (verify this logic with Clare)
    - If 0 then add to collisions (for adding to hash)
    - If 1 then subtract from collisions (for deleting from hash)
    - If -1 then do not modify collisions (for searching hash)
* fix bug in displayBST (tree form)
* Check if BST is preventing duplicates
* implement trendingWords
* Implement highestRetweet
* Find particular test data points
  + Data A, Data B, Data C
* Project documentation
  + Program Doc:
    - Program documents such as code, header, etc.
      * Input Data File
      * Output File
    - Test Plan
    - Executable Version of the Project
  + Presentation Doc:
    - Presentation Outline
    - Introduction w/ short description of each person’s assignment
    - Finalize presentation test plan
    - Create presentation outline
      * Include specific data points
    - Mach presentation on Saturday
    - Data Structure Diagram
    - Structure Chart
    - Explanation of Hash Function
    - Explanation of collision resolution
* Finalize, check and verify, clean and comment code
  + Try to find bugs; Implement a thorough test plan for bug testing
    - Test as a group so we can come up with creative test scenarios