**Small Input File Demonstration:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test #** | **Menu Option** | **Input** | **Output** | **Remarks** | **Team Member** |
|  | A - Add | Key  SP105 | Card Added | User will have to enter the additional fields as well. | Efrain |
|  | A - Add | Key  SP105 | Card Not Added - SP105 - already exists | Can't have duplicate keys. | Efrain |
|  | A - Add | Key  745 | Two letters and Three numbers | Format is LLNNN  (L = Letter and N = Numbers) | Efrain |
|  | A - Add | Key  ALR20 | Two letters and Three numbers | Format is LLNNN  (L = Letter and N = Numbers) | Efrain |
|  | A - Add | Name  "Timmy the Wizard"  ZZ123 | Card Added | User will have to enter the additional fields as well. Key will be checked similarly to Cases #4-6 | Efrain |
|  | A - Add | Name  "Timmy the Wizard"  ZZ124 | Card Added (as long as key entered is unique) | User will have to enter the additional fields as well. Key will be checked similarly to Cases #4-6 | Efrain |
|  | D - Delete | Key  745 | Two letters and Three numbers | Format is LLNNN  (L = Letter and N = Numbers) | Efrain |
|  | D - Delete | Key  ALR20 | Two letters and Three numbers | Format is LLNNN  (L = Letter and N = Numbers) | Efrain |
|  | D - Delete | Key  SP105 | Card Deleted | Needs to check for valid key | Efrain |
|  | D - Delete | Key  SP105 | Card Does Not Exit -  Not Delete | Card was just deleted in above case. | Efrain |
|  | D - Delete | Name  "Timmy the Wizard"  ZZ123 | Card Deleted | User will be given a list of cards with key numbers and will be asked which one they want to delete. | Efrain |
|  | D - Delete | Name  "Timmy the Wizard"  ZZ124 | Card Deleted | User will be given a list of cards with key numbers and will be asked which one they want to delete. | Efrain |
|  | D - Delete | Name  "Timmy the Wizard" | Card Doesn't Exist  Can Not Be Deleted | User will be given a list of cards with key numbers and will be asked which one they want to delete. | Efrain |
|  | S - Print Sorted List | Key | Prints Key Tree | Prints BST - InOrder Traversal by Key | Edward |
|  | S - Print Sorted List | Key | Prints Empty Message | If BST is empty | Edward |
|  | I - Print Indented List | Key | Prints Indented List | Prints Indented List | Edward |
|  | I - Print Indented List | Key | Prints Empty Message | If Tree is Empty | Edward |
|  | F - Find | Find Root by Key | Root is Found | Root is Displayed | Edward |
|  | D - Delete | Key  ???????? | Root is Deleted | Deletes the Root of the BST | Edward |
|  | I - Print Indented List | Key | Prints Indented List | Prints Indented List With New Root | Edward |
|  | A - Add | Key  ???????? | Add New Leaf | Adds Leaf to end of Tree | Edward |
|  | I - Print Indented List | Key | Prints Indented List | Prints Indented List With New Leaf | Edward |
|  | S - Print Sorted List | Name | Prints Name Tree | Prints AVL Tree - InOrder Traversal by Name | Steven |
|  | S - Print Sorted List | Name | Prints Empty Message | If AVL Tree is Empty | Steven |
|  | I - Print Indented List | Name | Prints Indented List | Prints Indented List | Steven |
|  | I - Print Indented List | Name | Prints Empty Message | If Tree is Empty | Steven |
|  | F - Find | Find Root by Name | Root is Found | Root is Displayed | Steven |
|  | D - Delete | Name  ???????? | Root is Deleted | Deletes Root | Steven |
|  | I - Print Indented List | Name | Prints Indented List | Prints Indented List With New Root | Steven |
|  | A - Add | Name  ???????? | Add New Leaf | Adds Leaf to end of Tree | Steven |
|  | I - Print Indented List | Name | Prints Indented List | Prints Indented List With New Leaf  Also shows Balancing? | Steven |
|  | H - Print Hash | PrintTable | Prints Hash Table | Prints Hash Table in Hash Table Sequence | Nick |
|  | H - Print Hash | DisplayTable | Prints Hash Table | Prints with Index Notation and Synonyms | Nick |
|  | T - Show Hash Stats | None | Displays Hash Stats | Displays Hash Stats | Nick |
|  | F - Find | Synonym by Key  ?????????? | Synonym is Found | Displays Synonym | Nick |
|  | D - Delete | Delete Synonym  ?????????? | Synonym is Deleted | Deletes Synonym | Nick |
|  | H - Print Hash | DisplayTable | Prints Hash Table | Prints List with Synonym Gone | Nick |
|  | A - Add | Add Entry  ?????????? | Adds Entry to Empy | Adding Entry that Hashes to Empty Index | Nick |
|  | H - Print Hash | DisplayTable | Prints Hash Table | Prints List with New Entry in Empty Index | Nick |
|  | Undo Delete | None | Successful! | Last Deleted Item Is Reinserted into the Stack | Jordan |
|  | Undo Delete | None | Unsuccessful! | What happens when multiple unique keys are on the stack????? | Jordan |
|  | V - Save Data Base | None | Saves Data to File | Show Saved Output File? | Jordan |
|  | Undo Delete | None | Prints Empty Message | Delete Stack is cleared when Save is Called | Jordan |

**Large Input File Demonstration:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test #** | **Menu Option** | **Input** | **Output** | **Remarks** | **Team Member** |
|  | PreMenu – Input File1 | None | HashTable LoadFactor  NOT over 75% |  | Jordan |
|  | PreMenu - Input File 2 | None | HashTable LoadFactor  over 75% | ReHash Table | Jordan |
|  | PreMenu - Input File 2 | None | HashTable LoadFactor  NOT over 75% |  | Jordan |
|  | I - Print Indented List | Key | Prints Indented List | Prints Indented List | Jordan |
|  | I - Print Indented List | Name | Prints Indented List | Prints Indented List | Jordan |
|  | H - Print Hash | DisplayTable | Prints Hash Table | Prints with Index Notation and Synonyms | Jordan |
|  | T - Show Hash Stats | None | Prints Hash Statistics |  | Jordan |
|  | Q - Quit | None | No Leak! |  | Jordan |

Key is the unique searchable member key. Name is a non unique searchable member. Collision resolution method: chained lists.