Programming Assignment 3

CS 202 – Data Structures

Deadline: 24th March 2024, 11:55 PM

Lead TAs: Safa Salam, Maira Kamal, Muhammad Muneeb Ahmad

Plagiarism Policy:

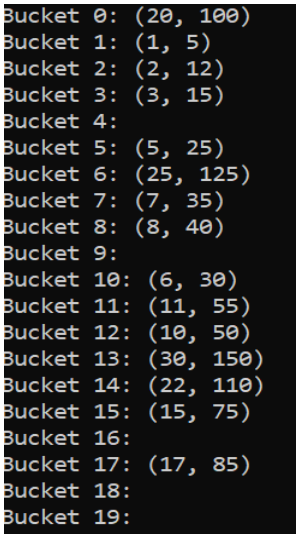
1. Students must not share the actual program code with other students.
2. Students must be prepared to explain any program code they submit.
3. Students cannot copy code from the Internet.
4. Students must indicate any assistance they received.
5. All submissions are subject to automated plagiarism detection.
6. Students are strongly advised that any act of plagiarism will be reported to the Disciplinary Committee

Task-1: Double Hashing (30 Marks)

In this task you will be implementing Double Hashing. For the HashTable class, a constructor and HashFunctions are already defined and implemented for you. The functions that you need to implement are:

1) **void insert(int key, int value)**
   Hash the key and insert it into HashTable. Continuously probe the hash table until an empty slot is found or traverse the entire table. Insert the key-value pair into the table once a suitable position is identified.
2) **int search(int key)**
   Search for a key in the Hashtable. If the key is found then return its corresponding value, otherwise **return -1.**
3) **void remove(int key)**
   Remove a key from the hashtable. Continuously probe the hash table until an empty slot is found or traverse the entire table. Remove the key-value pair from the table once the key is identified.
4) **void printTable()**
   Print the contents of the HashTable. You can use this function to check the state of your HashTable at any time. (Consider it very useful when debugging your code.)
   Note: This function is not tested in the test file, you can refer to the sample output to verify if your implementation is correct.

Sample Output for printTable function:



Note: In the test file your insertion test may pass in the case of incorrect insertions as well. While the test file will register that the test has passed, the accuracy of the insertion process will be validated through a detailed examination of the hash table's state using the `printTable` function.

Don’t hard code. Hidden test cases will be used for the final testing.

Task-2: B+ Trees (70 Marks)

Part A: (60 marks)

In this part you will be implementing a B+ tree. You have been given a BPlusTree.h file and a BPlusTree.cpp file with the starter code.

You have been given a struct BPlusNode which represents one node of the B+ tree. Each node has a vector for keys, a vector for values and a pointer to the children of that node.
The constructors and destructors code is already given to you. The functions you need to implement are:

1) **void BPlusTree::insert(int key, int value)**
   Insert a key-value pair in the B+ Tree.
2) **void BPlusTree::remove(int key)**
   Remove a key from the B+ Tree.
3) **void BPlusTree::printTree()**
   Print the entire B+ Tree in this function. This function will help you debug the code easily.
4) **vector<int> BPlusTree::inOrderTraversal()**
   Implement in-order traversal of the B+ Tree and return in the form of a vector.
5) **int BPlusTree::findMinKey()**
   Return the minimum key in the B+ Tree
6) **int BPlusTree::findMaxKey()**
   Return the maximum key in the B+ Tree
7) **int findKey(int key)**
   Return the given key if found. Otherwise **return -1.**
8) **void BPlusTree::clearTree()**
   Clear the entire tree in this function.

Note: Other helper functions have been defined for you as well. You may or may not use them. You may also declare more helper functions if you want.

Part B: (10 marks)

In this part you will be using the implementation of the previous part and work with actual data. The constructor has already been made for you. Functions you will need to implement are:

1. **void RecordManager::loadRecordsFromFile()**
   Loads records from the file into the B+ tree.
2. **void RecordManager::saveRecordsToFile()**
   Saves records from the B+ tree to the file.
3. **void RecordManager::addRecord()**
   Adds a record to the B+ tree.
4. **void RecordManager::deleteRecord()**
   Deletes a record from the B+ tree.
5. **void RecordManager::printRecords()**
   Prints all records in the B+ tree.
6. **void RecordManager::printRangeQuery()**
   Prints records within a specified range.
7. **int RecordManager::findMinKey()**
   Finds the minimum key in the B+ tree.
8. **int RecordManager::findMaxKey()**
   Finds the maximum key in the B+ tree.
9. **void RecordManager::clearAllRecords()**
   Clears all records from the B+ tree.

The text file for this task has been given to you. Please note that the output from the test file will be checked along with your code so do not hard code.

The sample output will be shared later.

TESTING:

To test the implementation of your code, compile and run the test files using the following commands in WSL:

PART 1:

```
g++ DHTestFile.cpp -pthread -std=c++11
./a.out
```

PART 2:

```
g++ TestBPlusTree.cpp -pthread -std=c++11
./a.out
```

```
g++ TestRecordManager.cpp -pthread -std=c++11
./a.out
```

Also make sure you are in the correct folder as each task has its own folder.

Submission guidelines:

You only have to submit .cpp files for both tasks. If you add any helper function in the header files then you may submit those as well.

Zip the complete folder and use the following naming convention:

PA3_<roll number>.zip

For example, if your roll number is 25100018 then your zip file name should:

PA3_25100018.zip

All submissions must be uploaded on LMS before the deadline.
You are allowed 5 "free" late days during the semester (that can be applied to one or more assignments; the final assignment will be due tentatively on the final day of classes, i.e., before the dead week and cannot be turned in late. The last day to do any late submission is also the final day of classes, even if you have free late days remaining).

If you submit your work late for any assignment once your 5 “free” late days are used, the following penalty will be applied:

- 10% for work submitted up to 24 hours late
- 20% for work submitted up to 2 days late
- 30% for work submitted up to 3 days late
- 100% for work submitted after 3 days (i.e., you cannot submit assignments more than 3 days late after you have used your 5 free late days.