**CSC 2200 – Computer Science II**

**Lab #10**

**Take-Home**

**60 Points**

1. Save a copy of this document with your name and the assignment number somewhere in the file name. For example, the file name *“Jane\_Doe\_CSC2200\_Lab1.docx”*
2. Copy-and-paste your answers (e.g., C++ source code) into the document.
3. Copy-and-paste the program output window.
4. Submit the following files separately (do not compress the files) to the Canvas in one submission:

1) This document as a word document (i.e., with the extension ***.doc*** or ***.docx***).

2) All C++ source code solution file(s) (only the ***.cpp*** and ***.h*** files) to the Canvas item associated with this assignment/lab solution. ***\*\*If you modified it, submit it***

\*Submit entire Visual Studio solution, if possible, otherwise only the required files.

Questions (60 points):

Implement functions: isEmpty, remove, retrieve. (30 points)

Exercise 3: Implement standardDeviation function. (30 points)

**Note**: *If you do not provide a screen shot of the tested functions, I will take 10 pts*

*off.*

\* **Copying-and-pasting your *C++ program code* to a Word document**

1) From within the Visual Studio program, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

#include "HashTable.h"

//Constructor

template <typename DataType, typename KeyType>

HashTable<DataType, KeyType>::HashTable(int initTableSize)

{

tableSize = initTableSize;

dataTable = new BSTree<DataType, KeyType>[tableSize];

}

//Copy constructor

template <typename DataType, typename KeyType>

HashTable<DataType, KeyType>::HashTable(const HashTable& other)

{

copyTable(other);

}

//Assignment operator

template <typename DataType, typename KeyType>

HashTable<DataType, KeyType>& HashTable<DataType, KeyType>::operator=(const HashTable& other)

{

if (this != &other) {

clear();

copyTable(other);

}

return \*this;

}

//Destructor

template <typename DataType, typename KeyType>

HashTable<DataType, KeyType>::~HashTable()

{

clear();

delete[] dataTable;

}

//Insert

template <typename DataType, typename KeyType>

void HashTable<DataType, KeyType>::insert(const DataType& newDataItem)

{

int hashValue = hashFunction(newDataItem.getKey());

dataTable[hashValue].insert(newDataItem);

}

//Remove

template <typename DataType, typename KeyType>

bool HashTable<DataType, KeyType>::remove(const KeyType& deleteKey)

{

int hashValue = hashFunction(deleteKey);

return dataTable[hashValue].remove(deleteKey);

}

//Retrieve

template <typename DataType, typename KeyType>

bool HashTable<DataType, KeyType>::retrieve(const KeyType& searchKey, DataType& returnItem) const

{

int hashValue = hashFunction(searchKey);

bool found = dataTable[hashValue].retrieve(searchKey, returnItem);

if (!found) {

returnItem = DataType(); //Set returnItem to an undefined state

}

return found;

}

//Clear

template <typename DataType, typename KeyType>

void HashTable<DataType, KeyType>::clear()

{

//Goes through each BSTree in the HashTable, calls clear() function on every BSTree

for (int i = 0; i < tableSize; ++i) {

dataTable[i].clear();

}

}

//isEmpty

template <typename DataType, typename KeyType>

bool HashTable<DataType, KeyType>::isEmpty() const

{

for (int i = 0; i < tableSize; ++i) {

if (!dataTable[i].isEmpty()) {

return false;

}

}

return true;

}

//show10 file

#include "show10.cpp"

//Standard deviation

template <typename DataType, typename KeyType>

double HashTable<DataType, KeyType>::standardDeviation() const

{

//Calculate the mean (average number of keys per bucket)

double mean = 0;

for (int i = 0; i < tableSize; ++i) {

mean += dataTable[i].getCount();

}

mean /= tableSize;

//Calculate variance

double variance = 0;

for (int i = 0; i < tableSize; ++i) {

double deviation = dataTable[i].getCount() - mean;

variance += deviation \* deviation;

}

variance /= tableSize;

//Calculate standard deviation (square root of variance)

double stdDeviation = sqrt(variance);

return stdDeviation;

}

template <typename DataType, typename KeyType>

void HashTable<DataType, KeyType>::copyTable(const HashTable& source)

{

}

\*\* **Copying-and-pasting a C++ “*output window”* to a Word document**

1) From the Visual Studio output window, press **ALT-PrintScreen**.

2) From within the Word document, press **CTRL-V**.

