University of Newcastle
Discipline of Computing and Information Technology
Semester 1, 2020 - SENG1120/6120

Assignment 3

Due using the Blackboard Assignment submission facility:
11:59PM – Friday June 12th, 2020 (Week 13)

NOTE: *The important information about submission and code specifics at the end of this assignment specification***.**

INTRODUCTION

In lectures, we have discussed the benefits of using binary search trees and hash tables to store information. In this assignment you will implement both and compare their performances in terms of speed of access.

ASSIGNMENT TASK

You are required to create a binary search tree, and a hash table data structures to store strings, that will function with the supplied `TreeHashTableDemo.cpp` and `makefile`. Both structures should have functions to add and remove elements. The classes MUST be implemented as class templates. The binary search tree class must be called `BSTree` and will use as nodes instances of `BTNode`. The hash table class must be named `HTable`.

You will be provided a demo file (`TreeHashTableDemo.cpp`), C++ makefile, and your classes need to interface with it. The binary search tree contents must be printed using an **inorder traversal**. The hash table class must store the numbers in an array of integers with `size 150`, and the contents can be printed from position `0` to `n-1`, but only for those positions that contain a valid entry. The hash function used must sum up the ASCII values of each character in the string and return the result of that sum mod (%) 150:

```
int hashfun(string value)
{
    int addResult = 0;
    // put you code here to add up the ASCII codes of all
    // characters in the parameter value and store in the
    // integer variable addResult
    return addResult % 150;
}
```

SUBMISSION

Make sure your code works with the files supplied, and DO NOT change them. For marking, we will add the main file to the project and compile it using the makefile, together with your own files. If it does not compile or run, your mark will be zero.

Your submission should be made using the Assignments section of the course Blackboard site. **Incorrectly submitted assignments will not be marked.** You should provide all your files. Also, if necessary, provide a `readme.txt` file containing any instructions for the marker. Each program file should have a proper header section including your name, course and student number, and your code should be properly documented.

Remember that your code should compile and run correctly using Cygwin. There should be no segmentation faults or memory leaks during or after the execution of the program.

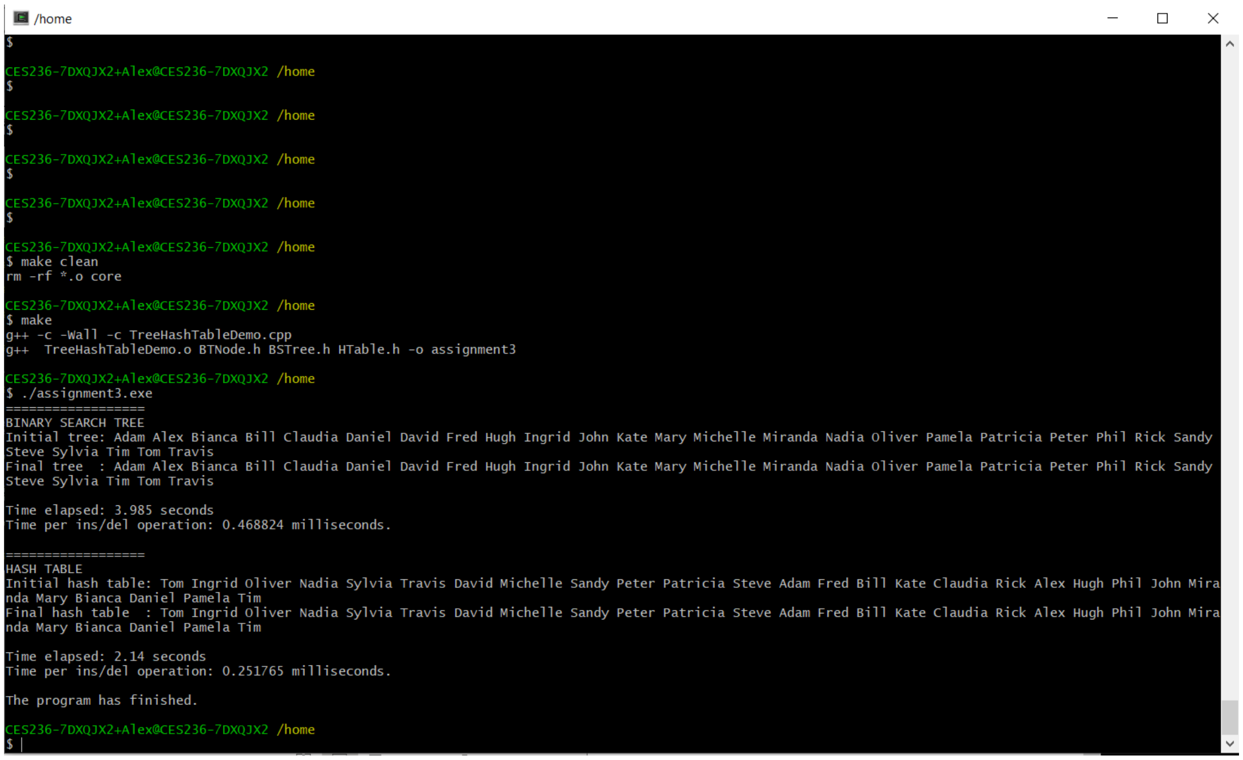
Compress all your files into a *single .zip file*, using your student number as the filename. For example, if your student number is **c9876543**, you would name your submission:

`c9876543.zip`

Submit by clicking in the link that will be created in the Assignments section on Blackboard.

Late submissions are subject to the rules specified in the Course Outline. Finally, a completed Assignment Cover Sheet should accompany your submission.

This assignment is worth 15 marks of your final result for the course.



```
/home
$

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ make clean
rm -rf *.o core

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ make
g++ -c -Wall -c TreeHashTableDemo.cpp
g++  TreeHashTableDemo.o BTNode.h BSTree.h HTable.h -o assignment3

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ ./assignment3.exe
==================
BINARY SEARCH TREE
Initial tree: Adam Alex Bianca Bill Claudia Daniel David Fred Hugh Ingrid John Kate Mary Michelle Miranda Nadia Oliver Pamela Patricia Peter Phil Rick Sandy
Steve Sylvia Tim Tom Travis
Final tree  : Adam Alex Bianca Bill Claudia Daniel David Fred Hugh Ingrid John Kate Mary Michelle Miranda Nadia Oliver Pamela Patricia Peter Phil Rick Sandy
Steve Sylvia Tim Tom Travis

Time elapsed: 3.985 seconds
Time per ins/del operation: 0.468824 milliseconds.

==================
HASH TABLE
Initial hash table: Tom Ingrid Oliver Nadia Sylvia Travis David Michelle Sandy Peter Patricia Steve Adam Fred Bill Kate Claudia Rick Alex Hugh Phil John Mira
nda Mary Bianca Daniel Pamela Tim
Final hash table  : Tom Ingrid Oliver Nadia Sylvia Travis David Michelle Sandy Peter Patricia Steve Adam Fred Bill Kate Claudia Rick Alex Hugh Phil John Mira
nda Mary Bianca Daniel Pamela Tim

Time elapsed: 2.14 seconds
Time per ins/del operation: 0.251765 milliseconds.

The program has finished.

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ |
CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ make clean
rm -rf *.o core

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ make
g++ -c -Wall -c TreeHashTableDemo.cpp
g++  TreeHashTableDemo.o BTNode.h BSTree.h HTable.h -o assignment3

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$ ./assignment3.exe
==================
BINARY SEARCH TREE
Initial tree: Adam Alex Bianca Bill Claudia Daniel David Fred Hugh Ingrid John Kate
Mary Michelle Miranda Nadia Oliver Pamela Patricia Peter Phil Rick Sandy Steve Sylvia
Tim Tom Travis
Final tree  : Adam Alex Bianca Bill Claudia Daniel David Fred Hugh Ingrid John Kate
Mary Michelle Miranda Nadia Oliver Pamela Patricia Peter Phil Rick Sandy Steve Sylvia
Tim Tom Travis

Time elapsed: 3.985 seconds
Time per ins/del operation: 0.468824 milliseconds.

==================
HASH TABLE
Initial hash table: Tom Ingrid Oliver Nadia Sylvia Travis David Michelle Sandy Peter
Patricia Steve Adam Fred Bill Kate Claudia Rick Alex Hugh Phil John Miranda Mary
Bianca Daniel Pamela Tim
Final hash table  : Tom Ingrid Oliver Nadia Sylvia Travis David Michelle Sandy Peter
Patricia Steve Adam Fred Bill Kate Claudia Rick Alex Hugh Phil John Miranda Mary
Bianca Daniel Pamela Tim

Time elapsed: 2.14 seconds
Time per ins/del operation: 0.251765 milliseconds.

The program has finished.

CES236-7DXQJX2+Alex@CES236-7DXQJX2 /home
$
```

Obs: the computational time for each data structure will depend on the computer, but it is expected that the hash table will be faster than the binary search tree.