University of Newcastle
Discipline of Computing and Information Technology
Semester 2, 2019 - SENG1120/6120

Assignment 3

Due using the Blackboard Assignment submission facility:
11:59PM – November 8th, 2019

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 in charge of inventory management support in a factory. You are required to create binary search tree and hash table data structures to store instances of a class `MechPart`. Both data structures should have functions to add, remove, display, overloaded operators, among others. 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, a `MechPart` class, and a text file with a list of part codes and quantities, and your classes need to interface with them. The binary search tree contents must be printed using an **inorder traversal**. The hash table class must store the instances of `MechPart` in an **array of size 5,000**, 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 to be used is provided below. You can *copy-and-paste* that code to your `HTable` class:

```
template <typename value_type>
int HTable<value_type>::hashfun(const value_type& value)
{
     int position = 0;
     string temp = value.get_code();

     for (int i=0; i<(int)temp.length(); i++)
     {
          position += (i+1) * (i+1) * temp.at(i);
     }
     return position % TABLE_SIZE;
}
```

A FEW POINTS

- As you implement the classes, you will notice that even though `HTable` and `BSTree` are generic class templates, they only work with `MechPart`. That is, some public methods in the `HTable` and `BSTree` classes refer directly (and only make sense) with `MechPart`.
- That was a design decision for this assignment. We could get around that, but it would make the assignment extremely challenging for most students.
- When you implement the classes, start small and then add more and more functionality as you go. Start by implementing `BSTree` and comment out everything in the demo file and in the `makefile` that refers to `HTable`. This way, you can test your code as you go. Once `BSTree` is working, you can move on to `HTable`.
- The solution provided in the next page is the result that you should get if your code is correct. The computational time will differ, as this was run in my personal computer, but the hash table will be faster than the binary search tree for sure.
- *What does the demo code do?* The demo code reads the code and quantity of 100 mechanical parts from a file and populates the binary search tree. Then, it removes and adds some of the elements 100,000 times, and prints some statistics about the process. After that, it does the same for the hash table, and the program finishes.

As demonstrated in the **Week 5 Assignment Skills and Techniques Lecture** (*and the associated sample files and documentation*) – which can be found on **Bb** – begin by analysing **this specification**, and the **demo code files** provided; this should allow you to easily determine the **Class Names**, the **Functions you need to implement** and **Required Operators** that your assignment should use.

SENG6120 STUDENTS (or for bonus marks)

The 100 mechanical parts in the input file provided do not induce any collisions in a hash table with 5,000 positions. If you reduce the size of the array in the hash table to 300 positions, however, there will be 16 collisions. Your task is to create a hash table that eliminates the problem of collisions by having a linked list in each position of the array. This way, if two or more elements map to the same position of the array, they are stored as different nodes in the linked list for that position.

Test you approach by reducing the size of the array to 300 and making sure that all 100 elements are stored correctly, and the demo provided still works.

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 the `.h`, `.cpp` and `.hpp` files related to your **`HTable`**, **`BSTree`** and **`BTNode`** classes only, plus an assessment item coversheet. Also, if necessary, provide a `readme.txt` file containing 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, and appending '**_bonus**' if you have attempted the bonus section. For example, if your student number is **c9876543** and you have implemented the **Bonus Section** (*including 6120 students*), you would name your submission:

`c9876543_bonus.zip`

The same student who has NOT attempted the **Bonus Section** would simply name their submission:

`c9876543.zip`

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

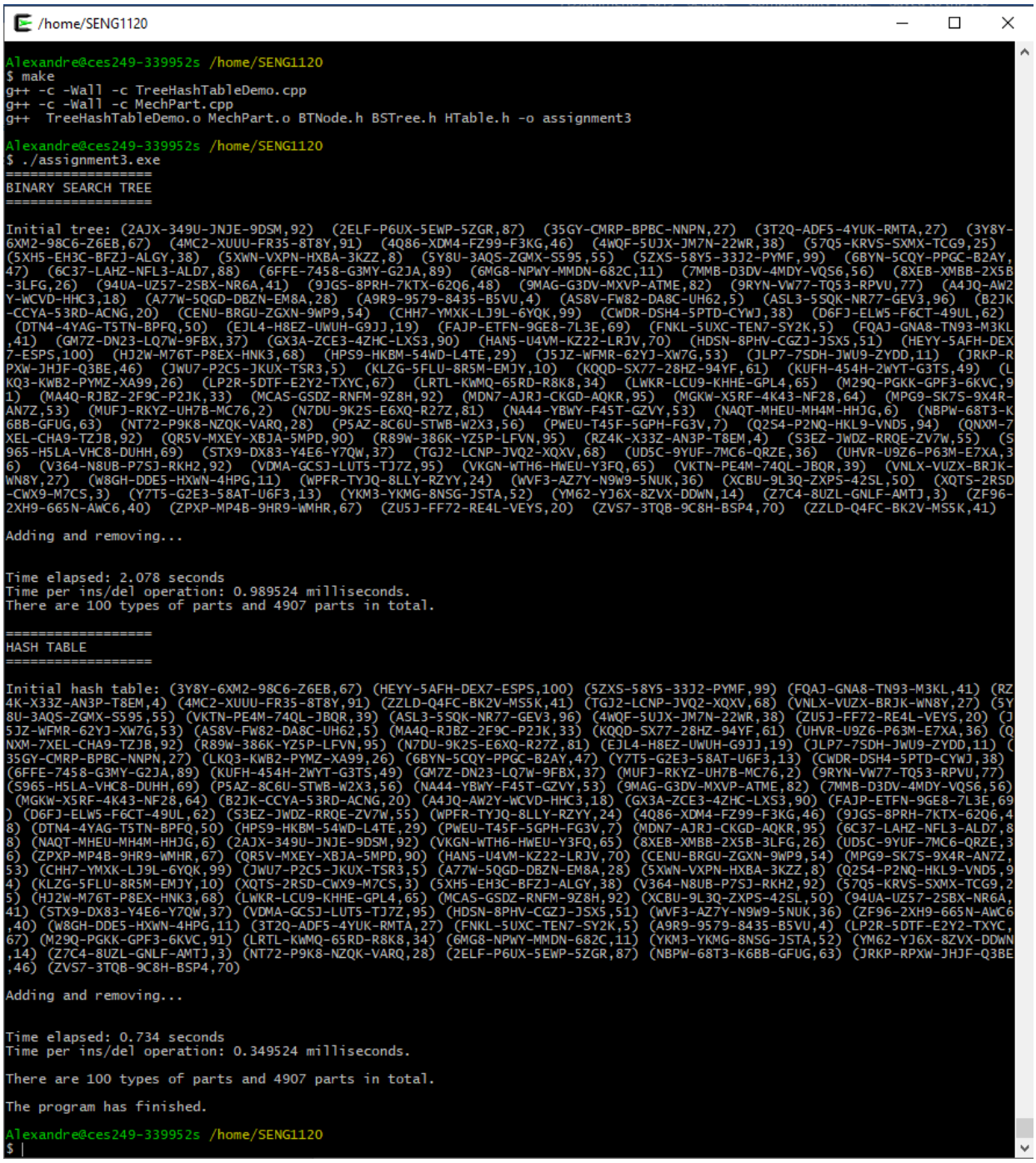
Late submissions are subject to the rules specified in the Course Outline. Finally, a completed Assignment Cover Sheet should accompany your submission.

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

Dan – v1.0 (2019-10-05)

SAMPLE OUTPUT

Using the supplied files, your output should match the output below:



```
Alexandre@ces249-339952s /home/SENG1120
$ make
g++ -c -Wall -c TreeHashTableDemo.cpp
g++ -c -Wall -c MechPart.cpp
g++  TreeHashTableDemo.o MechPart.o BTNode.h BSTree.h HTable.h -o assignment3

Alexandre@ces249-339952s /home/SENG1120
$ ./assignment3.exe
==================
BINARY SEARCH TREE
==================

Initial tree: (2AJX-349U-JNJE-9DSM,92)  (2ELF-P6UX-5EWP-5ZGR,87)  (35GY-CMRP-BPBC-NNPN,27)  (3T2Q-ADF5-
4YUK-RMTA,27)  (3Y8Y-6XM2-98C6-Z6EB,67)  (4MC2-XUUU-FR35-8T8Y,91)  (4Q86-XDM4-FZ99-F3KG,46)  (4WQF-5UJX-
JM7N-22WR,38)  (57Q5-KRVS-SXMX-TCG9,25)  (5XH5-EH3C-BFZJ-ALGY,38)  (5XWN-VXPN-HXBA-3KZZ,8)  (5Y8U-3AQS-
ZGMX-S595,55)  (5ZXS-58Y5-33J2-PYMF,99)  (6BYN-5CQY-PPGC-B2AY,47)  (6C37-LAHZ-NFL3-ALD7,88)  (6FFE-7458-
G3MY-G2JA,89)  (6MG8-NPWY-MMDN-682C,11)  (7MMB-D3DV-4MDY-VQS6,56)  (8XEB-XMBB-2X5B-3LFG,26)  (94UA-UZ57-
2SBX-NR6A,41)  (9JGS-8PRH-7KTX-62Q6,48)  (9MAG-G3DV-MXVP-ATME,82)  (9RYN-VW77-TQ53-RPVU,77)  (A4JQ-AW2Y-
WCVD-HHC3,18)  (A77W-5QGD-DBZN-EM8A,28)  (A9R9-9579-8435-B5VU,4)  (AS8V-FW82-DA8C-UH62,5)  (ASL3-5SQK-NR77-
GEV3,96)  (B2JK-CCYA-53RD-ACNG,20)  (CENU-BRGU-ZGXN-9WP9,54)  (CHH7-YMXK-LJ9L-6YQK,99)  (CWDR-DSH4-5PTD-
CYWJ,38)  (D6FJ-ELW5-F6CT-49UL,62)  (DTN4-4YAG-T5TN-BPFQ,50)  (EJL4-H8EZ-UWUH-G9JJ,19)  (FAJP-ETFN-9GE8-
7L3E,69)  (FNKL-5UXC-TEN7-SY2K,5)  (FQAJ-GNA8-TN93-M3KL,41)  (GM7Z-DN23-LQ7W-9FBX,37)  (GX3A-ZCE3-4ZHC-
LXS3,90)  (HAN5-U4VM-KZ22-LRJV,70)  (HDSN-8PHV-CGZJ-JSX5,51)  (HEYY-5AFH-DEX7-ESPS,100)  (HJ2W-M76T-P8EX-
HNK3,68)  (HPS9-HKBM-54WD-L4TE,29)  (J5JZ-WFMR-62YJ-XW7G,53)  (JLP7-7SDH-JWU9-ZYDD,11)  (JRKP-RPXW-JHJF-
Q3BE,46)  (JWU7-P2C5-JKUX-TSR3,5)  (KLZG-5FLU-8R5M-EMJY,10)  (KQQD-SX77-28HZ-94YF,61)  (KUFH-454H-2WYT-
G3TS,49)  (LKQ3-KWB2-PYMZ-XA99,26)  (LP2R-5DTF-E2Y2-TXYC,67)  (LRTL-KWMQ-65RD-R8K8,34)  (LWKR-LCU9-KHHE-
GPL4,65)  (M29Q-PGKK-GPF3-6KVC,91)  (MA4Q-RJBZ-2F9C-P2JK,33)  (MCAS-GSDZ-RNFM-9Z8H,92)  (MDN7-AJRJ-CKGD-
AQKR,95)  (MGKW-X5RF-4K43-NF28,64)  (MPG9-SK7S-9X4R-AN7Z,53)  (MUFJ-RKYZ-UH7B-MC76,2)  (N7DU-9K2S-E6XQ-
R27Z,81)  (NA44-YBWY-F45T-GZVY,53)  (NAQT-MHEU-MH4M-HHJG,6)  (NBPW-68T3-K6BB-GFUG,63)  (NT72-P9K8-NZQK-
VARQ,28)  (P5AZ-8C6U-STWB-W2X3,56)  (PWEU-T45F-5GPH-FG3V,7)  (Q2S4-P2NQ-HKL9-VND5,94)  (QNXM-7XEL-CHA9-
TZJB,92)  (QR5V-MXEY-XBJA-5MPD,90)  (R89W-386K-YZ5P-LFVN,95)  (RZ4K-X33Z-AN3P-T8EM,4)  (S3EZ-JWDZ-RRQE-
ZV7W,55)  (S965-H5LA-VHC8-DUHH,69)  (STX9-DX83-Y4E6-Y7QW,37)  (TGJ2-LCNP-JVQ2-XQXV,68)  (UD5C-9YUF-7MC6-
QRZE,36)  (UHVR-U9Z6-P63M-E7XA,36)  (V364-N8UB-P7SJ-RKH2,92)  (VDMA-GCSJ-LUT5-TJ7Z,95)  (VKGN-WTH6-HWEU-
Y3FQ,65)  (VKTN-PE4M-74QL-JBQR,39)  (VNLX-VUZX-BRJK-WN8Y,27)  (W8GH-DDE5-HXWN-4HPG,11)  (WPFR-TYJQ-8LLY-
RZYY,24)  (WVF3-AZ7Y-N9W9-5NUK,36)  (XCBU-9L3Q-ZXPS-42SL,50)  (XQTS-2RSD-CWX9-M7CS,3)  (Y7T5-G2E3-58AT-
U6F3,13)  (YKM3-YKMG-8NSG-JSTA,52)  (YM62-YJ6X-8ZVX-DDWN,14)  (Z7C4-8UZL-GNLF-AMTJ,3)  (ZF96-2XH9-665N-
AWC6,40)  (ZPXP-MP4B-9HR9-WMHR,67)  (ZU5J-FF72-RE4L-VEYS,20)  (ZVS7-3TQB-9C8H-BSP4,70)  (ZZLD-Q4FC-BK2V-
MS5K,41)

Adding and removing...


Time elapsed: 2.078 seconds
Time per ins/del operation: 0.989524 milliseconds.
There are 100 types of parts and 4907 parts in total.

==================
HASH TABLE
==================

Initial hash table: (3Y8Y-6XM2-98C6-Z6EB,67) (HEYY-5AFH-DEX7-ESPS,100) (5ZXS-58Y5-33J2-PYMF,99) (FQAJ-GNA8-
TN93-M3KL,41) (RZ4K-X33Z-AN3P-T8EM,4) (4MC2-XUUU-FR35-8T8Y,91) (ZZLD-Q4FC-BK2V-MS5K,41) (TGJ2-LCNP-JVQ2-
XQXV,68) (VNLX-VUZX-BRJK-WN8Y,27) (5Y8U-3AQS-ZGMX-S595,55) (VKTN-PE4M-74QL-JBQR,39) (ASL3-5SQK-NR77-
GEV3,96) (4WQF-5UJX-JM7N-22WR,38) (ZU5J-FF72-RE4L-VEYS,20) (J5JZ-WFMR-62YJ-XW7G,53) (AS8V-FW82-DA8C-UH62,5)
(MA4Q-RJBZ-2F9C-P2JK,33) (KQQD-SX77-28HZ-94YF,61) (UHVR-U9Z6-P63M-E7XA,36) (QNXM-7XEL-CHA9-TZJB,92) (R89W-
386K-YZ5P-LFVN,95) (N7DU-9K2S-E6XQ-R27Z,81) (EJL4-H8EZ-UWUH-G9JJ,19) (JLP7-7SDH-JWU9-ZYDD,11) (35GY-CMRP-
BPBC-NNPN,27) (LKQ3-KWB2-PYMZ-XA99,26) (6BYN-5CQY-PPGC-B2AY,47) (Y7T5-G2E3-58AT-U6F3,13) (CWDR-DSH4-5PTD-
CYWJ,38) (6FFE-7458-G3MY-G2JA,89) (KUFH-454H-2WYT-G3TS,49) (GM7Z-DN23-LQ7W-9FBX,37) (MUFJ-RKYZ-UH7B-MC76,2)
(9RYN-VW77-TQ53-RPVU,77) (S965-H5LA-VHC8-DUHH,69) (P5AZ-8C6U-STWB-W2X3,56) (NA44-YBWY-F45T-GZVY,53) (9MAG-
G3DV-MXVP-ATME,82) (7MMB-D3DV-4MDY-VQS6,56) (MGKW-X5RF-4K43-NF28,64) (B2JK-CCYA-53RD-ACNG,20) (A4JQ-AW2Y-
WCVD-HHC3,18) (GX3A-ZCE3-4ZHC-LXS3,90) (FAJP-ETFN-9GE8-7L3E,69) (D6FJ-ELW5-F6CT-49UL,62) (S3EZ-JWDZ-RRQE-
ZV7W,55) (WPFR-TYJQ-8LLY-RZYY,24) (4Q86-XDM4-FZ99-F3KG,46) (9JGS-8PRH-7KTX-62Q6,48) (DTN4-4YAG-T5TN-
BPFQ,50) (HPS9-HKBM-54WD-L4TE,29) (PWEU-T45F-5GPH-FG3V,7) (MDN7-AJRJ-CKGD-AQKR,95) (6C37-LAHZ-NFL3-ALD7,88)
(NAQT-MHEU-MH4M-HHJG,6) (2AJX-349U-JNJE-9DSM,92) (VKGN-WTH6-HWEU-Y3FQ,65) (8XEB-XMBB-2X5B-3LFG,26) (UD5C-
9YUF-7MC6-QRZE,36) (ZPXP-MP4B-9HR9-WMHR,67) (QR5V-MXEY-XBJA-5MPD,90) (HAN5-U4VM-KZ22-LRJV,70) (CENU-BRGU-
ZGXN-9WP9,54) (MPG9-SK7S-9X4R-AN7Z,53) (CHH7-YMXK-LJ9L-6YQK,99) (JWU7-P2C5-JKUX-TSR3,5) (A77W-5QGD-DBZN-
EM8A,28) (5XWN-VXPN-HXBA-3KZZ,8) (Q2S4-P2NQ-HKL9-VND5,94) (KLZG-5FLU-8R5M-EMJY,10) (XQTS-2RSD-CWX9-M7CS,3)
(5XH5-EH3C-BFZJ-ALGY,38) (V364-N8UB-P7SJ-RKH2,92) (57Q5-KRVS-SXMX-TCG9,25) (HJ2W-M76T-P8EX-HNK3,68) (LWKR-
LCU9-KHHE-GPL4,65) (MCAS-GSDZ-RNFM-9Z8H,92) (XCBU-9L3Q-ZXPS-42SL,50) (94UA-UZ57-2SBX-NR6A,41) (STX9-DX83-
Y4E6-Y7QW,37) (VDMA-GCSJ-LUT5-TJ7Z,95) (HDSN-8PHV-CGZJ-JSX5,51) (WVF3-AZ7Y-N9W9-5NUK,36) (ZF96-2XH9-665N-
AWC6,40) (W8GH-DDE5-HXWN-4HPG,11) (3T2Q-ADF5-4YUK-RMTA,27) (FNKL-5UXC-TEN7-SY2K,5) (A9R9-9579-8435-B5VU,4)
(LP2R-5DTF-E2Y2-TXYC,67) (M29Q-PGKK-GPF3-6KVC,91) (LRTL-KWMQ-65RD-R8K8,34) (6MG8-NPWY-MMDN-682C,11) (YKM3-
YKMG-8NSG-JSTA,52) (YM62-YJ6X-8ZVX-DDWN,14) (Z7C4-8UZL-GNLF-AMTJ,3) (NT72-P9K8-NZQK-VARQ,28) (2ELF-P6UX-
5EWP-5ZGR,87) (NBPW-68T3-K6BB-GFUG,63) (JRKP-RPXW-JHJF-Q3BE,46) (ZVS7-3TQB-9C8H-BSP4,70)

Adding and removing...


Time elapsed: 0.734 seconds
Time per ins/del operation: 0.349524 milliseconds.

There are 100 types of parts and 4907 parts in total.

The program has finished.

Alexandre@ces249-339952s /home/SENG1120
$
```