# CS 2150 Exam 2

## Name

You MUST write your e-mail ID on **EACH** page and put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will be ripped up and not graded – sorry to have to be strict on this!

There are 6 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points.

**Answers for the short-answer questions should not exceed about 20 words; if your answer is too long (say, more than 30 words), you will get a zero for that question!**

This exam is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

*You step in the stream,*
*But the water has moved on.*
*This page is not here.*

## Page 2: Miscellaneous

1. [3 points] Why do we want hash functions to be fast? What hash table methods would be affected if the hash function were not fast?

2. [3 points] Describe a situation in which you would choose to store values in a tree instead of a hash table

3. [3 points] Explain why one would not typically use AVL trees for buckets in a hash table that uses separate chaining.

4. [3 points] When searching for a node with key $x$ in an $8$-node binary search tree and failing to find it, what is the maximum number of distinct values that must be compared with $x$? Explain your answer briefly.

5. [3 points] Draw a valid AVL tree with 7 nodes and height $4$ (indexing from $1$). Have each node in the tree store a single integer value.

## Page 3: Hash Tables

Consider a hashtable which stores key-value pairs, where each key and each value is an 8-byte integer (e.g., key: 8, value: 25). Assume that the key 0 is never used and so can be used to indicate that a key-value pair is unused rather than having a separate flag in the array. Suppose the hashtable contains 500 key-value pairs, and the size of the underyling array is 1009; additionally

- 400 of the keys map to an index not used by any other key
- 100 of the keys map to an index used by exactly one other key

For these questions, assume that all pointers are 8 bytes, and you may ignore storage required for the internal bookkeeping of `new` or to track the capacity or size of the vector or array.

6. [3 points] If the hashtable uses linear probing to resolve collisions, then how much storage in bytes (this includes unused spots in the table's underlying array) will the hashtable require? You may leave your answer as a formula. Explain your answer briefly.

7. [3 points] If the hashtable uses linear probing to resolve collisions, then what is the maximum number of consecutive array locations that could be occupied in the hashtable?

8. [3 points] If the hashtable has 1009 buckets and uses separate chaining to resolve collisions, then how much storage in bytes (again, including unused spots in the table's underlying array) will the hashtable require? You may leave your answer as a formula. Explain your answer briefly. For this question, assume the underlying array stores pointers to ListNode objects and ListNode objects contain only member variables (1) int (the key), (2) int (the value) and (3) a ListNode pointer.

9. [3 points] If the hashtable uses separate chaining to resolve collisions, what is the **maximum** number of consecutive underlying array indices (i.e. spots in the array) that could be occupied by at least one element in the hashtable?

## Page 4: Assembly

10. [3 points] Consider the x86-64 assembly below and recall that in the Linux x86-64 calling convention:

    - the first argument is in **RDI**, the second argument is in **RSI**, the return value is in **RAX**

    Write C++ code equivalent to the assembly function `foo` above: (A valid prototype is provided for you.)

```
foo:
    mov RAX, [RDI + RSI * 8]
    mov RAX, [RDI + RAX * 8]
    ret
```

```
extern "C"
long foo(long *a, long i) {
        //notice there isn't much space.
        //You can do this in one line of code


}
```

11. [8 points] Fill in the blanks in the assembly code on the right necessary to make it equivalent to the C++ code on the left and obey the calling convention. **It may not be necessary to use all the blanks**. Recall that:

    - the first argument is in **RDI**, the second argument is in **RSI**, the return value is in **RAX**
    - RBX, RBP, and R12-R15 are caller-saved registers.

```
int countSpaces(char *array) {
    int i = 0;
    while (isspace(array[i]) != 0) {
            i += 1;
    }
    return i;
}
```

```
countSpaces:

    --------

    --------
    mov RBX, RDI        // array in RBX
    mov R8, 0           // i in R8
loop:

    mov _____, RBX
    add _____, R8

    -------

    -------

    -------
    call isspace

    -------

    -------
    cmp RAX, 0          // isspace() ?= 0
    je endLoop
    add R8, 1           // i += 1
    jmp loop
endLoop:

    mov ___, ___

    -------

    -------
    ret
```

## Page 5: More Assembly and some IBCM

12. [6 points] Suppose we want to imitate a stack in IBCM, and we decide to use an arbitrary memory address (e.g., 0x000) to store the address of the top of the stack. The stack will start at address 0xfff and grow downward (so address 0x000 will initially store the value 0x0fff and we will assume one item is already on the stack at the beginning of execution). Fill in the method on the right with the missing IBCM commands. Keep the design decisions we've made on the left in mind while writing your code.

- the stack pointer is in address 0x000, but we will just call it **SP** in our psuedo-code
- the caller has already preplaced the value they wish to push into a hardcoded memory address, let's call it **P**
- the accumulator contains the **return address** when the method begins. We need to store this somewhere (let's call this **RET**) and jump back to this address after we are done pushing.

```
push:
          store    RET       ;save the ret address

          load     SP        ;adjust stack pointer

          ---------

          ---------

          load     4000      ;setup store command

          ---------

          ---------

          load     P         ;load param

doit:     0000               ;store param on stack

          load  C000         ;construct command to
                             ;jmp back to caller
          ---------

          ---------

jmpit:    0000               ;jmp back to caller
```

13. [3 points] Briefly, what do we mean when we claim that IBCM is turing-complete?

**Page 6: Unix Honor Pledge**

# Unix Honor Pledge for CS 2150, Spring 2018

On my honor as a student at the University of Virginia, I agree, unless specifically given permission otherwise:

- To not use any integrated development environment (IDE) for the development of C++ or assembly programs for the work for this course. These include, but are not limited to, such development environments such as Eclipse, Netbeans, Xcode, Geany, Visual Studio, Atom, etc. The Sublime editor is allowed, provided it is used without extensions for build automation, debugging, or code completion; it shouldn't be any more powerful than a text editor like Emacs.
  - This applies to IDEs on any platform, such as Microsoft Windows, Mac OS X, Unix, etc.
- To develop my course programming work on Unix or Unix-like system. These systems include Linux (any variant), FreeBSD, Solaris, Unix systems installed on a virtual machine running on another OS (such as Windows or Mac OS X), remote Unix systems, and others as discussed in lecture.
  - IDEs cannot be used in such systems, however, as that defeats the purpose.
  - Mac OS X is allowed as long as the Unix-based features are used, and IDEs are not used.
  - The Windows 10 Bash shell is allowed, but you are on your own on this one – and you still can't use Windows IDEs or Windows-based editors.
- If there is any doubt about the applicability of this pledge, I will ask before assuming.

Failure to abide by this agreement will mean an immediate failure for the course, and the raising of honor charges.

Signature: ______________________________