CS 2150 Final Exam, fall 2021

Name _____

You MUST *clearly* write your name above. You must also write your e-mail ID on **EACH** page.

If you are still writing when "pens down" is called, your exam will not be graded – even if you are still writing to fill your name and userid. So please do that first. 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 about 30 words), you will get a zero for that question! This is not a hard maximum, but you should try your best to answer the question as concisely as you can.

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.

_____

_____

_____

_____

Serious error.
All shortcuts have disappeared.
Screen. Mind. Both are blank.

Page 2: Exam 1 material

1. [3 points] Convert -237 to a 32-bit two's complement integer, and leave your answer in big-Endian hexadecimal. Note that $237 = 128 + 64 + 32 + 8 + 4 + 1$

2. [3 points] Convert the big-Endian value 0x12345678 to little-Endian

3. [6 points] Fill in the **worst-case** running times for the three primary operations for the listed data structures. Note that they should be interpreted appropriately for each data structure, so insert/remove/find for a stack are `push()`/`pop()`/`top()`, find for a queue is `enqueue()`/`dequeue()`/`front()`. Write the running time as just n, not $\Theta(n)$ and not "linear" (and appropriately different if it's not linear). For the lists (last four rows): Any removal is from the middle of the list, and assumes there is *already* an iterator or index to the element to be removed, if appropriate to that data structure. Insert assumes you are inserting to the end of the list that is most efficient. *Note that find() for the last four rows is find an element, not find at index k.*

Data structure	find time	insert time	remove time
Queue, linked-list based			
Stack, linked-list based			
Queue, vector-based			
Stack, vector-based			
Vector-based list			
Array-based list			
Doubly linked list			
Singly linked list			

Page 3: Exam 2 material

4. [6 points] Look at the x86 function below and answers the questions underneath it.

```
someFunc:
  xor rcx, rcx      ; rcx = i = 0

loop:
  cmp rcx, rsi      ; compare i and n
  jge done
  mov [rdi+4*rcx], rdx
  inc rcx
  jmp loop

done:
  ret
```

4.1 Does this function have parameters? If so, how many? _____

4.2 Does this function return anything? If so, what does it return? _____

4.3 In one sentence, what does this function do? _____

5. [6 points] We've decided to make an update to IBCM to give it more registers. Specifically, instead of only having 1 accumulator (which is implicitly used), we want to increase to 12 total registers. The catch is that we still want each instruction to only be 16 bits. We've decided to use the following scheme:

   We will still use four bits for the opcode, but we will need to TAKE some bits away from the 12 that we normally use to address memory and use it to address the registers instead (Combinations of these bits will refer to using specific registers). Answer the following questions about this new scheme:

   5.1 How many bits do we need, at minimum, to address the 12 registers? _____

   5.2 How many of the values from the previous question are "wasted" and don't actually reference any of the 12 registers? _____

   5.3 How many bits does this leave to address memory? _____

   5.4 How many unique memory addresses can we now address with this new scheme? _____

Page 4: Heaps and Huffman

6. [3 points] Suppose I insert five unique integers (e.g.: 1,2,3,4,5) into a standard min-heap. How many total times will these elements percolate up across all five insertions if the items are inserted in increasing sorted order? How many times if inserted in decreasing sorted order?

7. [3 points] In class we saw how you could use a min-heap to sort a list of numbers. Describe that (high-level) algorithm briefly.

8. [3 points] When will Huffman coding *not* work well as a compression algorithm?

9. [3 points] Draw the resulting structure (in tree form) if the following values were placed into a min-heap. The values, in order of insertion, are: 3, 6, 8, 2, 1, 7, 5

Page 5: Graphs

10. [6 points] In class, we saw Breadth-First Search, in which each node stores a distance and a *pointer to the predecessor node along the path* that leads to this distance. Look at the code below that prints the actual shortest path to a node in the Graph and fill in the missing blank lines:

```
/* Given a node n, print the shortest path to n */
/* Nodes contains a name (string), dist (int) and  */
/* path (Node pointer) fields  */
/* The start node will have a path = "NULL" */
/* Assume BFS has already been run on this Graph */
void printPath(Node* n) {

  if(_____) return;

  _____ //Can leave blank if needed

  printPath(_____);

  _____ //Can leave blank if needed
}
```

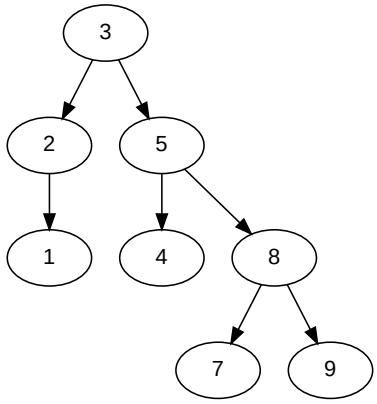
11. [6 points] Your friend tried to implement Dijkstra's algorithm, but it doesn't seem to be working correctly. You look at their code and discover they are missing two essential things. What are they?

```
void Graph::dijkstra(Vertex start) {
  Vertex v,w;
  start.dist = 0;

  while (there exist unknown vertices) {
    v = unknown v with the smallest distance;


    for each w adjacent to v {
      w.dist = v.dist + Cost_VW;
      w.cameFrom = v;
    }
}
```

1. Missing thing one: _____
2. Missing thing one: _____

Page 6: Data Structures

12. [6 points] Consider the following AVL tree. Draw the resulting tree when we insert 6.



13. [3 points] What is the big-Theta run-time of insertion into a hash table that uses separate chaining? *Assume that we are not concerned about duplicate values in our hash table.*

14. [3 points] Briefly explain the difference between *shared* and *replicated* multiple inheritance.