# Final Exam/Foundation Exam Review

Monday, July 27, 2020 4:09 PM

Your final exam will actually follow the Foundation Exam Format.

Why?
---
1. We have 2 hours not 3 hours
2. We are giving the Foundation Exam online, and I want some practice!
3. Good practice for those of you who end up taking exam!

Date: July 29th, 2020, Wednesday
Time: 4 - 6 pm (please get online by 3:50 pm, and I will extend turn in time through 6:10 pm)
Method: Four sections called FE-A, FE-B, FE-C and FE-D, which will mirror the sections on the Foundation exam that are typically called "Section 1A", "Section 1B", "Section 1C" and "Section 1D". Each section will have 3 questions, two of them will be worth 10 points and the other one will be worth 5 pts. Total = 100 pts, with each section equal 25 pts.
FE-A will be released on Webcourses at a couples minutes before 4 pm. FE-A's deadline will be 4:30 pm for submission, with a 10 minute late window, cutting off submissions at 4:40 pm. (No late penalty.)
FE-B will be released a couple minutes before 4:30 pm, due at 5:00 pm with a cutoff window of 5:10 pm.
FE-C will be released a couple minutes before 5:00 pm, due at 5:30 pm with a cutoff window of 5:40 pm.
FE-D will be released a couple minutes before 5:30 pm, due at 6:00 pm with a cutoff window of 6:10 pm.

AIDS - printed materials you have in front of you. Calculator but show your steps as if you didn't have one.

**I DON'T KNOW RULE - If you don't know how to approach a question, if you ONLY write "I DON'T KNOW" to the answer of a question, you'll get 30% on the 10 pts questions and 20% on the 5 pts questions.**

**File types allowed for submission: .txt, .doc, .docx, and .pdf.**

**Preferred - One file per each part, worst case if there are 2, we will manage.**

**Content**

**Exam Structure Link:**

http://www.cs.ucf.edu/wp-content/uploads/2019/08/FE-ExamOutline.pdf

**Part A/Section 1A**

**----------**

1) **Dynamic Memory Allocation Question**
   a. **Usually some code**
   b. **Either allocate some memory**
   c. **Or free some memory**
   d. **malloc, calloc, realloc, free**
2) **Linked Lists**
   a. **Usually some code**
   b. **Occasionally Tracing**
   c. **Know how to iterate through a linked list**
   d. **Know how to make small structural changes**
   e. **Understand when it's necessary to return a pointer to the potentially new front of the list.**
3) **Stacks/Queues**
   a. **Evaluate Postfix**
   b. **Infix --> Postfix**
   c. **Ask qualitative questions about stacks/queues.**
   d. **Rare but could be asked to code a portion of an implementation (array stack, array queue, LL stack, LL queue)**

**Example Infix --> Postfix**

```
                        a                   b         c
( ( 3 + 5 ) * 2 / ( 8 / 4 ) ) + 6


          /
         (
```

```
*     /
(     (     +
a     b     c
```

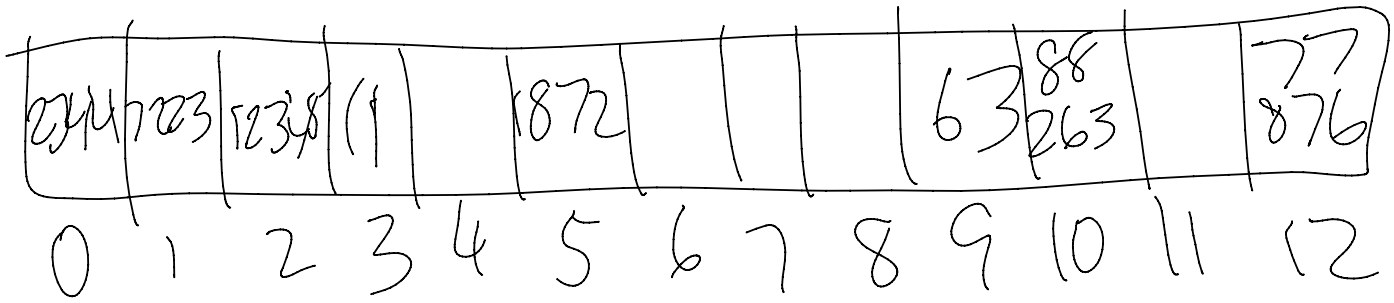
**3 5 + 2 * 8 4 / / 6 +**

**Part B/Section 1 B**

---

1) **Binary Trees**
   a. **Usually Coding**
   b. **Usually Recursive**
   c. **So, think about do I want to call my function recursively to both sides or just one side.**
   d. **What other work do I need to do once I get the answers from the recursive calls.**
   e. **Make sure you know your traversals.**
2) **Hash Tables or Binary Heaps**
   a. **Almost always tracing of some sort**
   b. **Hash Tables - linear probing, quadratic probing, linear chaining hashing**
   c. **Heaps - tracing inserts, deleteMin**

**Spr 19 Exam Posted online Q16**




63, 1872, 12345, 11, 7223, 88263

2344, 77876

9, 10, 0, 5, 12

3) AVL Tree/Trie

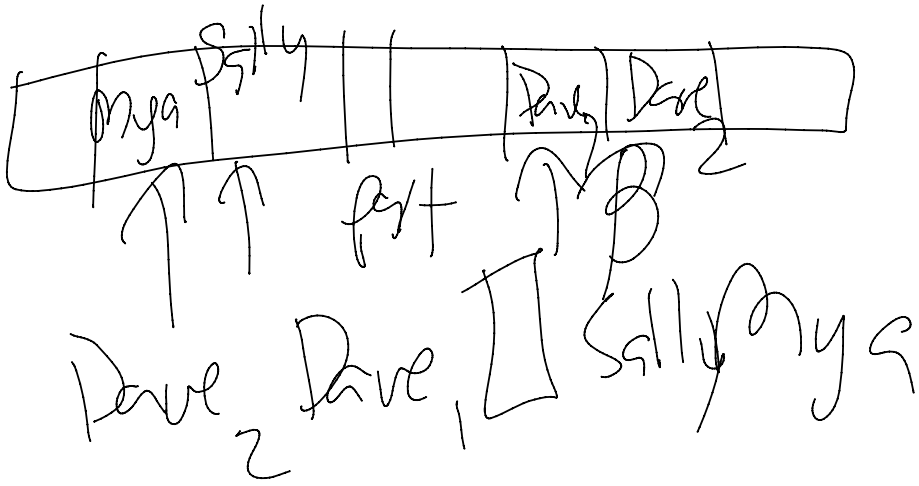
a. AVL - usually tracing
b. Trie - usually coding

Part C/Section 2A
---

1) Algorithm Analysis Question
   a. Analyze a new problem
   b. List the run-times of several algorithms/data structures from class
2) Timing Questions
   a. Set up equation based on given info
   b. Solve for c
   c. Use c to then answer question
3) Summation/Recurrence Relation

Part D/Section 2B
---

1) Recursive Coding
   a. always forced to do recursively
   b. base case (2-3 pts)
   c. think about breaking the problem down into smaller pieces
   d. think about how to solve your instance of the problem with the solution(s) to the smaller piece(s).
2) Sorting
   a. Usually tracing
   b. If coding, probably one of the n squared sorts
   c. Insertion, Bubble, Selection
   d. Merge Sort
   e. Quick Sort
   f. Process questions about how the sorts work



3. Bitwise operators on Backtracking
    a. Bitwise ops - trace
    b. Bitwise ops - ask a function to write
    c. Backtracking - usually filling in a framework of code

In general (1<<n)-1 in binary is n 1s in a row.

1<<n  1000000000 (1 followed by n 0s)
if we subtract 1, we carry every time and get
111111111

P[0] = 000001110
P[1] = 001101001
P[2] = 101011100
P[3] = 010000001
P[4] = 000111111
P[5] = 111110000

res = 000001110 (with P[0])
= 001101111 (with P[1])
= 101111111 (with P[2])
= 111111111 (with P[3])
answer is 4