# EECS 268 2020 Spring Exam 2

Due Date: Monday April 13th 11:59pm

Submit via email to your Lab TA

Place an X in the box that applies to you:

|  |  |
| --- | --- |
| I am in the MWF section (officially) |  |
| I am in the TR section (officially) |  |

# Next, read and sign the following page.

# Rules (Read These!)

* DO NOT alter the formatting of the test. Any changes to formatting could result in grading errors
* Only mark your answers within designated answer boxes
* Read and sign below

Hi EECS 268 students. I know you just got this take home exam off the internet, but can you promise me, or more importantly, promise yourself that you'll take this exam without any unauthorized aid?

* Unauthorized aid: google searches, the materials or help of other students, past exams, help from the undergrad staff or GTAs, any compilers (e.g. don't just put the code tracing problems in a compiler and run them)

But since we're in such a crazy world and our class didn't go as planned, I am fine with you using the following authorized aid:

* Authorized resources: your notes, your labs (the code, not the compiler), the online lectures I made for you, your amazing brain

I'd like this exam to still be an assessment of your skill and understanding.

If you agree to this, then please type your name in the box below. If you can't agree to it, then I'm afraid I won't grade your exam. Afterall, I'm not here to test how well you can google things. That's what your job will do.

|  |  |
| --- | --- |
| Name |  |
| KUID |  |
| Date |  |

# [20pts] Complexity

Please write the big-O complexity of the following actions. Your answer should come in form of, for example, O(n) or O(n\*log2n) or other values for big-O. Any questions involving linked list, assume you only have a front pointer. Stacks only have a top. Queues have a front and a back. Lists, Stacks, and Queues all have a Node based implementation. Assume the data structures contain n elements unless otherwise stated.

|  |  |
| --- | --- |
| **Action** | **Big-O Complexity** |
| Pop off a stack |  |
| Adding to the back of a List |  |
| Destructor of a Queue |  |
| Enqueue on a Queue |  |
| Swapping the first and last value in a linked list. |  |
| Push n values onto a Stack |  |
| Add n values to the front of a LinkedList |  |
| Print all values in a linked list using calls to getEntry |  |
| Swapping the first and last values of an array. |  |
| Exhaustive search for a value in a 2D array of dimensions nXn. |  |

# [15pts] Sorts: Who am I?

Based on the descriptions provided, identify the sort. Some descriptions may not describe the entire algorithm of the sort. Please write your answers in the provided boxes.

1. I am an iterative sort that does a comparison of two neighboring values and swaps them if they are out of order, then moves on to the next two neighbors and so on until I reach the end of the array.

|  |  |
| --- | --- |
| Sort name |  |

1. I am a sort that recursively “breaks” an array in half until I get single element arrays. I can guarantee that each recursive step breaks the array exactly in half, give or take 1 element.

|  |  |
| --- | --- |
| Sort name |  |

1. I am a sort that divides the array up into a sorted and unsorted sections.

|  |  |
| --- | --- |
| Sort name |  |

1. I am a sort that partitions an array over and over.

|  |  |
| --- | --- |
| Sort name |  |

1. I am a sort that loves to find the index of the minimum value (assuming ascending) in the remaining portion of the array over and over.

|  |  |
| --- | --- |
| Sort name |  |

# [10pts] Coding: Recursion

Write a recursive solution to calculate the ith Gibbonacci number. The Gibbonacci sequence, though much less famous than its counterpart, is defined as:

gib1 ⇒ 3

gib2 ⇒ 10

gib3 ⇒ 83

gibi ⇒ gib(i-1) + gib(i-2) + gib(i-3) where i is greater than 3.

Example calls to your function:

gib( 1 ); //returns 3

gib( 2 ); //returns 10

gib( 3 ); //returns 83

gib( 4 ); //returns 96

Write your code in the designated box on the following page

|  |
| --- |
| //write your gibbonacci function below |

# [20pts] Coding: Recursion

Write the definition for a function that takes an array of doubles and its size. The function will reverse the order of values in the array. You may not change the parameter list. Vague hint: arr[3] is the same as \*(arr + 3). OPTIONAL: You may alter the parameter list to whatever you like for a 5pt penalty. (NO LOOPS ALLOWED in any solution)

|  |
| --- |
| void reverseArr(double arr[ ], int size)  {  } |

# [10pts] Recursion: Tracing

Given the code below, indicate what is printed to the terminal.

void recFunc(int n)

{

if(n > 10)

{

std::cout << n << std::endl;

recFunc( n-3 );

}

else if (n > 5)

{

recFunc( n-2 );

std::cout << n << std::endl;

}

else

{

std::cout << n << std::endl;

}

}

int main()

{

recFunc(18);

}

|  |
| --- |
| Output to terminal |
|  |

# 

# [10pts] Maze Walking

Given the following maze and starting coordinates, what is the exhaustive order (no unmarking after deadends) that we will traverse the maze.

Rules:

P - passage

W - Walls

E - Exit (as soon as exit is found, stop)

Move order check:

Up, Right, Down, Left

Starting position: 0, 0 (top left corner of maze)

Maze

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | P | P | P | P | P |
| W | W | P | W | W | P |
| P | P | P | W | P | P |
| P | W | W | W | W | P |
| P | P | P | E | W | P |

Edit this table to show the order of traversal (remember, don't just show the final path, but the order we'd visit all possible passageways until an exit is found. I've done the first 3 steps for you.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

# [15pts] Conceptual

1. [3pts] If Jill's bubble sort function takes 10 seconds to sort 1000 numbers. How long would you anticipate it to sort 2000 numbers?

|  |  |
| --- | --- |
| Answer |  |

1. [3pts] We saw in class that if a recursive function does not have a base case it will keep making recursive calls, but it can't make recursive calls forever. What will stop the recursion (and the program) if even a base case is missing? (This is referring to a bad thing happening)

|  |  |
| --- | --- |
| Answer |  |

1. [3pts] Assume a pointer called *nums* is pointing to an array of 5 ints (given below). What will be printed if the following runs?

//values in array pointed to by nums: [5, 10, 15, 20, 25]

int\* temp = (nums+3)

std::cout << temp[0];

|  |  |
| --- | --- |
| Answer |  |

1. [3pts] If a recursive function call recFunc has a parameter called num. If a total of four calls of recFunc are on the call stack, how many instances of num exist?

|  |  |
| --- | --- |
| Answer |  |

1. [3pts] Once the world is normal again, what data structure would you use to simulate a grocery line (where the first customer in line is the first one to check out)?

|  |  |
| --- | --- |
| Answer |  |

PAGE 11 SHOULD BE THE FINAL PAGE OF THE EXAM. CHECK THE PAGE NUMBER BELOW. IF IT'S NOT PAGE 11, PLEASE CORRECT ANY FORMATTING ISSUES

Thank you so much for taking my test.