I&C SCI 46 Diagnostic Exam 1, Winter 2023

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO

SILENCE AND STORE ALL ELECTRONICS

This is a diagnostic exam intended to help you evaluate your readiness for the real exam.

The following rules apply to you, whether you think they do or not. Read and understand them; failure to abide by these rules, or directions given by course staff during the exam, may result in disciplinary action, including but not limited to a failing grade in the class.

- This exam is solely for students enrolled in this lecture. Anyone not enrolled in this lecture may not take an exam.

- You may not open the exam or begin writing until the instructor has explicitly given you permission to do so.

- Keep your UCI ID readily accessible during the test. Proctors may request to see it.

- This exam is closed book, closed notes, and is individual effort. Once course staff begin passing out exams, you may not communicate with anyone other than proctors for any reason, nor may you have electronics, including calculators watches and phones, available to you during the test for any reason.

  YOU DO NOT NEED A CALCULATOR!

- If you leave your seat during the test for any reason, your instructor may collect it and deem you to have turned it in. Do not ask proctors for an exemption to this, they are not authorized to grant such.

- You must take the exam in your assigned seat unless the professor (not a TA) tells you otherwise. The instructor will call to cease writing at X:48 AM, at which point you must immediately cease writing and close the exam. You may not write any further at that point, including finishing one’s current sentence.

- If you believe a question is ambiguous, write at least two reasonable interpretations and indicate clearly which one you will be using. Then answer your question with that assumption. Unless your interpretation makes the problem much more trivial than intended, we will grade your response as if one of us had made that clarification.

- The purpose of the real exam is to evaluate how well you understand the material presented in the course. It is an academic integrity violation to do anything that subverts the goals of this assessment including, but not limited to, not doing your own work or submitting that of anyone else.

- We will only grade responses marked in the space provided for each question.
Nothing you write on this page will be graded. The next page in this booklet contains a spot to answer these questions. You may use this page as scratch paper if you would like, and room to do so exists.

1. (1 point) For each of the following functions, determine whether it is $O(n)$, $\Omega(n)$, or both. You do not need to provide proof or justification. Make sure you clearly indicate, for each one, whether you believe it is $O(n)$, $\Omega(n)$, or both. If the graders cannot determine your answer, you will not get credit for it.

Note that we are not asking you to provide the “best” $O$-notation/$\Omega$-notation; merely to describe it with one or both of the choices provided.

\[ a(n) = n(\log n)^{10} \quad \square O(n) \quad \square \Omega(n) \]

\[ b(n) = n^{0.99} \quad \square O(n) \quad \square \Omega(n) \]

\[ c(n) = 10^{100} n \quad \square O(n) \quad \square \Omega(n) \]

2. (1 point) Consider the following graph. Clearly indicate the edges that constitute a breadth-first search tree starting from vertex $G$ on the answer page.

3. (2 points) Suppose I want to have a “Blue and Gold Stack” class. This is a class that has color-coded push, pop, top and size functions; for example, there is a “blue push” and a “gold push.” Explain how you can implement this using a single array whose capacity is set at some value $C$, which will always be larger than the combined sizes of the blue and gold stacks. Your approach may only use the array of size $C$ and $O(1)$ additional space. You do not need to write code, but instead describe how you would implement the eight functions.

You will earn no credit if you solve this by simply having two traditional stacks as private data members and redirecting the calls to bluePush() to the blue one, etc.
Write your answer for question 1 here

\[ a(n) \quad \square \Theta(n) \quad \square \Omega(n) \]

\[ b(n) \quad \square \Theta(n) \quad \square \Omega(n) \]

\[ c(n) \quad \square \Theta(n) \quad \square \Omega(n) \]

Write your answer for question 2 here.

\[ \begin{array}{cccc}
Q & K & H & X \\
B & M & W & \\
Z & P & G & E
\end{array} \]

Write your answer for question 3 here. Be sure the grader can find each part.
4. (1 point) Suppose I am writing a program that will use a skip list with maximum height $h = 3\lceil \log n \rceil$ - that is, when I add an element to the skip list, if I get $3\lceil \log n \rceil$ “heads” in a row, I will stop adding height to the element. You cannot modify my code or select the input. You can, however, change the way the coin works from “50% heads, 50% tails” (current status) to another set of probabilities (which must still add up to 100%). You cannot introduce other results: the coin can still only land on heads or tails (no sideways landing, for instance).

Your goal is to cause my Skip List to use the maximum possible amount of space. How would you bias the coin to achieve this effect? Explain briefly why this will cause the given problem.

5. (2 points) The most disgusting recursive problem ever.

The McDonald’s near campus sells Chicken McNuggets in orders of 6, 9, or 20. Suppose you are ordering for a party and you know exactly how many McNuggets will be eaten by guests. It turns out that, for any integer $n \geq 44$, you can order exactly $n$ Chicken McNuggets at this McDonald’s.

For purposes of this problem, you cannot throw out McNuggets or allow them to go uneaten, such as by acquiring $n = 44$ by buying two twenty packs and a six pack, then discarding two. If the thought of this many Chicken McNuggets is too disgusting, you may pretend you are buying $n \geq 44$ celery sticks in bunches of 6, 9, or 20 (feel free to rename the function below in that case).

Finish the recursive function below to complete the ordering and return the counts by reference parameters. You may assume for this problem that there will be no overflow or underflow at any point in the problem and that stack space is not a concern. The code has been started for you and is part of a correct solution.

You may write in your choice of C++ or pseudo-code. You do not need to delete the list when you are done. If you elect to write in C++, we will not deduct points for minor syntax errors, as long as your intent is clear.

```cpp
void buyChicken ( unsigned n , unsigned & num6Packs , unsigned & num9Packs ,
                 unsigned & num20Packs )
{
    if ( 44 == n ){
        num20Packs = 1;  num6Packs = 4;   num9Packs = 0;
    } else if (45 == n ){
        num20Packs = 0;  num6Packs = 3;   num9Packs = 3;
    } else if (46 == n ){
        num20Packs = 2;  num6Packs = 1;   num9Packs = 0;
    }
}
```
Write your answer for question 4 here.

Write your answer for question 5 here. You do not need to copy over the starting code given, nor should you.
6. (3 points) Consider the following definition for a node in a linked list:

```cpp
struct Node {
    Node(unsigned v, Node * n) : value(v), next(n) {}
    unsigned value;
    Node * next;
};
```

Suppose we have a linked list of these and wish to answer a question: is there a subset of these Nodes such that the combined value equals a particular value ‘target’?

Complete the following function that returns a bool value to indicate the answer to the question. You may assume that the initial call gives a link to the front of the linked list and that no overflow or underflow will occur at any point in the run of a program.

You may write in your choice of C++ or pseudo-code. You do not need to delete the list when you are done. If you elect to write in C++, we will not deduct points for minor syntax errors, as long as your intent is clear.

*Hint: if there is a subset of the Nodes whose combined value adds to the target, then either the first element is in that subset or it isn’t.*

*This problem is substantially easier if you use recursion. For many students, this will be the hardest question on the test. You may wish to consider answering it last. While a full page is available on the answer sheet, there are concise answers to this problem.*

```cpp
bool subsetSum(Node * front, unsigned target)
```

Write your answer for question 6 here. It is very likely you will not need the entire space.

```c
bool subsetSum(Node *front, unsigned target)
{
```