Exam 2
COSC 220: Computer Science II
Dr. Joseph Anderson
Due: 11:00pm, 09 April 2020

Instructions: Read the “honesty pledge” below and sign where indicated. Complete the exam and scan/photograph the pages. You may either combine them into a multi-page document or compress them all in a single folder. In either case, preserve the question order as best as possible. Your grade will be posted on MyClasses; standard procedures for discussion of grades will apply. Late submissions will not be accepted.

During this exam:

1. ✗ You may not communicate with classmates, peers, faculty, or anybody else about the content on this exam until after the due date and time.

2. ✗ You may not consult any online or digital resource; this includes videos, web-pages, and forums.

3. ✔ You may use your own notes or programs from class lectures or the course textbook (see the course syllabus for text title and edition).

4. ✔ You may prepare your exam submissions digitally (in a word processor or typesetting system).

5. ✔ You may test any code portions of this exam by writing them and compiling on your own system, but only submit the code that is relevant to the question at hand. If a question asks for only a single function, provide only the code for that function.

Honesty Pledge
I, the undersigned, acknowledge that I have read and will adhere to the above instructions and stipulations for the completion of this exam. I understand that violation of any one of these restrictions constitutes academic misconduct in the context of this exam, and will result in 1) a grade of zero on this exam and 2) a report to the Salisbury University Office of Academic Affairs for further action, as appropriate according to the academic policies provided at https://www.salisbury.edu/administration/academic-affairs/misconduct-policy.aspx.

_________________________
Name (print)

_________________________
Name (sign)
Questions

1. (10 pts) Answer the following:

   (a) What is the best and worst case asymptotic time complexity of Bubble Sort, Insertion Sort, and Selection Sort? Which inputs realize these scenarios for each algorithm?

   (b) Describe the principles behind the Mergesort algorithm, using some pseudocode to illustrate. What is the asymptotic behavior of its runtime?
2. (5 pts) Consider running MergeSort on an array with the values 4, 15, 19, 22, 30, 48. Give a permutation of these values in which no items are swapped until the final merge of the left three values with the right three values, and when that merge happens, each value changes position. Justify your answer!
3. (10 pts) Answer the following:

(a) For a class called MyClass, what is the prototype one must use to define a copy constructor for the class? Explain, possibly with an example, the purpose of explicitly defining a class copy constructor.

(b) For the class called MyClass, what is the syntax to define the overloaded assignment operator? What is the functionality and purpose of the overloaded assignment operator, as opposed to the copy constructor?

(c) When is the copy constructor called? When is the overloaded assignment operator called?
4. (10 pts) Consider a class called `ImperialLength` with three private integer values: `inches`, `feet`, and `yards`. Define the following operators as class members:

```cpp
// add two lengths
ImperialLength operator+(ImperialLength);

// compares two lengths
bool operator<(ImperialLength);

// allow read-only extraction of inches, feet, or length with indices 0, 1, 2.
int operator[](int)

// conversion to an integer representing total number of inches
operator int();
```
5. (10 pts) Prove the following (by definition, not theorem):

(a) $x^3 + 2x + 10$ is $O(x^3)$.
(b) $x^2$ is not $O(100x)$.
(c) $x \log_2 x$ is $O(x^2)$.
(d) $1/x$ is $O(1)$.
6. (10 pts) What is the running time of the following function, in terms of the parameter \( n \), in simplified Big-O notation?

    function A(n):
        s := 0
        for i = 1 to 2n
            for j = 1 to i^2
                s := s * i + j
            end
        end
        return s
    end
7. (10 pts) Consider the pseudocode for the following recursive function:

```plaintext
function A(n):
    if n < 10
        return 5
    end

    for i = n/2 to n
        s := s * s
    end

    return A(n/2) + A(n/2)
end
```

(a) What is the base case of the recursion?
(b) Write the runtime of the function as a recurrence relation.
(c) Simplify the recurrence relation by unrolling, and state the total cost using Big-O notation.
8. (5 pts) What are the defining principles behind stacks and queues? Give one real-world example (not discussed in class or assignments) of applications where each data structure could be used to solve a problem.
9. (5 pts) What is the output of the following code?

```cpp
#include<iostream>

void function(int);

int main(){
    int x = 6;
    function(x);
    return 0;
}

void function(int num){
    if (num > 0){
        for (int x = 0; x < num; x++){
            std::cout << "*";
        }
        std::cout << std::endl;
        function(num - 1);
    }
}
```
10. (10 pts) Write a function that accepts an array of integers and a number indicating the length of the array as arguments. The function should recursively calculate the sum of all the numbers in the array. Write the running time of the algorithm as a recurrence relation, \( T(n) \), and give a simplified big-O analysis by unrolling the recurrence.
11. (10 pts) Suppose a program has the following structure defined:

```c++
struct QueueNode{
    int priority;
    std::string name;
    QueueNode* next;
}
```

This structure is used to store some names in a *priority queue*: a queue in which the head node has the highest priority value among all the nodes, and the rest of the nodes are in the linked-list in descending order of their priority field.

Write a recursive function with prototype `void enqueue(QueueNode *start, int p, std::string n)` that inserts a new node with priority `p` and name `n` into the queue with head node `start`. Note that the new item must be inserted into the proper location, maintaining a sorted list at its completion. You may assume that the function is not called on an empty list. What is the running time of the algorithm for a list of `n` elements?
12. (5 pts) Describe how recursion is able to solve iterative problems without using loops. What computational mechanisms work to manage the various “states” that the problem can be in? In practice, can you think of any penalties one might suffer from taking advantage of this strategy? Should every algorithm be written recursively?