Computer Science Foundation Exam

August 8, 2020

Section I A

DATA STRUCTURES

ONLINE EXAM

Directions: You may either directly edit this document, or write out your answers in a .txt file, or scan your answers to .pdf and submit them in the COT 3960 Webcourses for the Assignment "Section I A". Please put your *name, UCFID and NID* on the top left hand corner of each document you submit. Please aim to submit 1 document, but if it's necessary, you may submit 2. Clearly mark for which question your work is associated with. If you choose to edit this document, please remove this cover page from the file you submit and make sure your *name, UCFID and NID* are on the top left hand corner of the next page (first page of your submission).

Question #	Max Pts	Category	Score
1	5	DSN	
2	10	DSN	
3	10	DSN	
TOTAL	25		

You must do all 3 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

Name: ______________________
UCFID: ______________________
NID: ________________________

1) (5 pts) DSN (Dynamic Memory Management in C)

Suppose we have a structure to store information about cases of juice. The structure is shown below: the name of the juice in the case is statically allocated. The structure also contains the number of containers of juice in that case. Complete the function below so that will takes 2 parameters: the name of a juice and an integer. Your function should create a new case of juice by allocating space for it, copying in the contents specified by the formal parameters into the struct and returning a pointer to the new case. You may assume that the pointer new_name is pointing to a valid string storing the name of a juice for which memory has already been allocated and is 127 or fewer characters.

```
#include <string.h>
#include <stdlib.h>

struct juice_case {
     char name[128];
     int num_bottles;
};

struct juice_case* create_case(char *new_name, int new_number) {









}
```

2) (10 pts) ALG (Linked Lists)

Suppose we have a linked list implemented with the structure below. The function below takes in a pointer, **`head`**, to a linked list which is guaranteed to store data in strictly ascending order. If the list doesn't contain the value 3, the function should create a struct node storing 3 in its data component, insert the node so that the listed pointed to by head stores its data, including 3, in strictly ascending order, and returns a pointer to the front of the resulting list. If a node already exists storing 3 in the list pointed to by head, then return head and make no changes to the list.

```
typedef struct node {
    int data;
    struct node* next;
} node;

node* addValue3(node* head) {

    if ( __________________ || _______________________ ) {
        node* tmp = malloc(sizeof(node));
        tmp->data = 3;
        tmp->next = head;
        return tmp;
    }

    if ( _____________________ )
        return head;

    node* iter = head;

    while (iter->next != NULL && _________________________ )

        iter = ________________;

    if ( _______________________ && _____________________ )
        return head;

    node* tmp = malloc(sizeof(node));
    tmp->data = 3;

    tmp->next = ______________________ ;

    iter->next = _____________________ ;

    return __________________ ;
}
```

3) (10 pts) DSN (Stacks)

Suppose we have implemented a stack using a linked list. The structure of each node of the linked list is shown below. The stack structure contains a pointer to the head of a linked list and an integer, size, to indicate how many items are on the stack.

```
typedef struct node {
    int num;
    struct node* next;
} node;

typedef struct stack {
    struct node *top;
    int size;
} stack;
```

The generalized Towers of Hanoi game can be represented by **`numTowers`** stacks of integers, where the values in each stack represent the radii of the disks from the game for the corresponding tower. Recall that a valid move involves taking a disk at the top of one stack and placing it on the top of another stack, so long as that other stack is either empty or the disk currently at the top of the other stack is bigger than the disk about to be placed on it. Complete the function below so that it takes in an array of three stacks representing the contents of the three towers in Towers of Hanoi and prints out all of the valid moves that could be made from that state, but doesn't move anything. You may assume that the array of stacks passed into the function represent a valid state in a Towers of Hanoi game, where the value stored in the stack is the corresponding disk radius and the disk radii range from 1 to n, for some positive integer n. Assume that you have access to the following functions that involve a stack and that they work as described:

// Returns the value stored at the top of the stack pointed to by s. If stack pointed to by s is empty, a
// random value is returned.
```
int peek(stack *s);
```

// Returns 1 if the stack pointed to by s is empty, and 0 otherwise.
```
int isEmpty(stack *s);
```

```
void printValidMoves(stack towers[], int numTowers) {

    for (int i=0; i<numTowers; i++) {
        for (int j=0; j<numTowers; j++) {

            if ( ___________________________ ) continue;

            if ( _____________ || ____________________________________ )

                printf("Valid move from tower %d to tower %d.\n", i, j);
        }
    }
}
```

Computer Science Foundation Exam

August 8, 2020

Section I B

DATA STRUCTURES

ONLINE EXAM

Directions: You may either directly edit this document, or write out your answers in a .txt file, or scan your answers to .pdf and submit them in the COT 3960 Webcourses for the Assignment "Section I B". Please put your *name, UCFID and NID* on the top left hand corner of each document you submit. Please aim to submit 1 document, but if it's necessary, you may submit 2. Clearly mark for which question your work is associated with. If you choose to edit this document, please remove this cover page from the file you submit and make sure your *name, UCFID and NID* are on the top left hand corner of the next page (first page of your submission).

Question #	Max Pts	Category	Score
1	5	ALG	
2	10	ALG	
3	10	ALG	
TOTAL	25		

You must do all 3 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

Name: ______________________
UCFID: ______________________
NID: ________________________

1) (5 pts) ALG (Binary Search Trees)

Consider the following tree traversals:

Pre-order: 7 2 9 14 16
Post-order: 2 16 14 9 7
In-order: 2 7 9 14 16

Is it possible for a **single** binary **search** tree to give rise to all three of those traversals? If so, draw the tree. If not, clearly explain why it's not possible.

2) (10 pts) ALG (Heaps/Hash Tables)

Consider the following hash table and the strings it already contains, along with the hash function being used to insert strings into the table. Assume the table only stores alphabetic strings.

Note: The length of the hash table is **<u>11</u>**.

llama	xenon	want	yurt		mop	nook			uvula	
0	1	2	3	4	5	6	7	8	9	10

```
// This function (which is pretty bad for hashing strings in the real
// world, by the way) assumes str is non-NULL and non-empty.
int hash(char *str)
{
   // Note: This converts letters on the range 'a' through 'z' or
   // 'A' through 'Z' to integers on the range 0 through 25.
   // For example: 'a' -> 0, 'b' -> 1, ..., 'z' -> 25.
   return (tolower(str[0]) - 'a')%11;
}
```

For each of the following questions, refer to the original hash table above. For example, in part (b), refer to the original table – **<u>not</u>** the table that contains the string you come up with in part (a).

a. (2 pts) Give a string that, if **<u>inserted</u>** into the table above using **<u>quadratic probing</u>**, would cause us to encounter the **<u>minimum</u>** number of collisions possible.

b. (2 pts) Give a string that, if **<u>inserted</u>** into the table above using **<u>quadratic probing</u>**, would cause us to encounter the **<u>maximum</u>** number of collisions possible.

c. (5 pts) Give all the alphabetic letters someone could have used to start their string in order to give a correct answer for part (b) of this problem.

d. (1 pt) Give a string that, if inserted into the table above using **<u>linear probing</u>**, would cause us to encounter the **<u>maximum</u>** number of collisions possible.

3) (10 pts) ALG (AVL Trees)

List the ranges of all the integer values that would cause a **<u>double</u>** rotation to occur if inserted into the following AVL tree (as opposed to a single rotation or no rotation at all). (For example: "-10 through -5 and any value greater than 93.") You may assume we do not allow the insertion of duplicate values into the tree. **<u>Note: A double rotation can alternately be described as a restructuring where, out of the three nodes that need to move structurally, the new root node was previously two levels below the node that needed to be restructured. These cases are also called the C-A-B and A-C-B cases.</u>**

```
          34
         /  \
       20    48
      /  \     \
    12  25      93
   /
-10
```

Computer Science Foundation Exam

August 8, 2020

Section II A

ALGORITHMS AND ANALYSIS TOOLS

ONLINE EXAM

Directions: You may either directly edit this document, or write out your answers in a .txt file, or scan your answers to .pdf and submit them in the COT 3960 Webcourses for the Assignment "Section II A". Please put your *name, UCFID and NID* on the top left hand corner of each document you submit. Please aim to submit 1 document, but if it's necessary, you may submit 2. Clearly mark for which question your work is associated with. If you choose to edit this document, please remove this cover page from the file you submit and make sure your *name, UCFID and NID* are on the top left hand corner of the next page (first page of your submission).

Question #	Max Pts	Category	Score
1	10	ANL	
2	5	ANL	
3	10	ANL	
TOTAL	25		

You must do all 3 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib.h, stdio.h, math.h, string.h) for that particular question have been made.

Name: ______________________
UCFID: ______________________
NID: ________________________

1) (10 pts) ANL (Algorithm Analysis)

There is a very long corridor of rooms, labeled 1 through n, from left to right. It is reputed that in the very last room, room *n*, there is the Treasure of the Golden Knight. Unfortunately, you don't know what *n* is equal to. Whenever you are in a particular room, you are allowed to ask questions of the form, "Is there a room 2^k slots to the right of my current location?", where k is a non-negative integer. For a fee, Knightro, an omnipresent, omnipotent, omniscient knight, will answer your question correctly, with either "yes" or "no." After you ask 1 or more questions from a single room, Knightro will move you, for free, to any of the rooms you asked a question about for which he replied "yes." Your goal is to get to room n by asking as few questions as possible, to reduce the fee that you pay Knightro. Devise a strategy to find the value of n and clearly outline this strategy. How many questions, in terms of n, will your strategy use, in the worst case? Answer, with proof, this last question with a Big-Oh bound in terms of n. **(Note: Any strategy that works will be given some credit. The amount of credit given will be based on how efficient your strategy is, in relation to the intended solution.)**

2) (5 pts) ANL (Algorithm Analysis)

An algorithm to find a particular value among n total values takes O(log(n)). On a data set with $n = 2^{30}$, it took 1.2 seconds to find the desired value. How many **milliseconds** will it take to find a value in a data set with $n = 2^{20}$? (Note: For ease of computation, you may use a logarithm with base 2.)

3) (10 pts) ANL (Summations)

Using the fact that if $x \neq 1$, then $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$, for positive integers n, determine the following summation, in terms of n (assume n is a positive integer). Express your answer as a fraction, where the numerator has two terms.

$$\sum_{i=2n+1}^{3n} 4^i$$

Computer Science Foundation Exam

August 8, 2020

Section II B

ALGORITHMS AND ANALYSIS TOOLS

ONLINE EXAM

Directions: You may either directly edit this document, or write out your answers in a .txt file, or scan your answers to .pdf and submit them in the COT 3960 Webcourses for the Assignment "Section II B". Please put your *name, UCFID and NID* on the top left hand corner of each document you submit. Please aim to submit 1 document, but if it's necessary, you may submit 2. Clearly mark for which question your work is associated with. If you choose to edit this document, please remove this cover page from the file you submit and make sure your *name, UCFID and NID* are on the top left hand corner of the next page (first page of your submission).

Question #	Max Pts	Category	Score
1	5	DSN	
2	10	ALG	
3	10	DSN	
TOTAL	25		

You must do all 3 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

Name: ______________________
UCFID: ______________________
NID: ________________________

1) (5 pts) DSN (Recursive Coding)

Write a ***recursive*** function that returns the sum of all of the even elements in an integer array ***vals***, in between the indexes ***low*** and ***high***, inclusive. For example, for the function call sumEven(vals, 3, 8) with the array vals shown below, the function should return 24 + 8 + 10 = 42, since these three numbers are the only even numbers stored in the array in between index 3 and index 8, inclusive.

index	0	1	2	3	4	5	6	7	8	9
vals[i]	15	13	28	19	24	8	7	99	10	14

```
int sumEven(int vals[], int low, int high) {












}
```

2) (10 pts) ALG/DSN (Sorting)

(a) (5 pts) Consider running a Bubble Sort on the array shown below. How many swaps will execute for the duration of the algorithm running on the array shown below? Explain how you got your answer.

97	16	45	63	13	22	7	58	72

Reasoning:

Number of Swaps: _________

(b) (5 pts) List the **best case** run time of each of the following sorting algorithms, in terms of n, the number of items being sorted. Assume all items being sorted are distinct.

(i) Insertion Sort ____________

(ii) Selection Sort ____________

(iii) Heap Sort ____________

(iv) Merge Sort ____________

(v) Quick Sort ____________

3) (10 pts) DSN (Backtracking)

A “unique” positive integer of n digits is such that no two adjacent digits differ by less than 2. Specifically, given an n digit number, $d_0d_1 \ldots d_{n-1}$, where d_0 is the most significant digit, (and thus, this one digit can’t be 0), $|d_i - d_{i+1}| \geq 2$ for all i ($0 \leq i \leq n-2$). Consider the problem of printing out all “unique” positive integers of n digits via backtracking, in numerical order. Fill in the code below to complete the task. (To run the code, one would have to call printWrapper with their desired parameter.)

```
#include <stdio.h>
#include <math.h>
void print(int number[], int n);
void printWrapper(int n);
void printRec(int number[], int k, int n);

void printWrapper(int n) {
    int* array = malloc(sizeof(int)*n);
    printRec(array, 0, n);
    free(array);
}

void printRec(int number[], int k, int n) {
    if (k == n) {
        ______________________ ;

        ______________________ ;
    }

    int start = 0;

    if ( ______________ )

        start = ___ ;
    for (int i=start; i < ____ ; i++) {

        if (k > 0 && ________________________ )
            continue;

        number[ ____ ] = _____ ;

        __________________________________ ;
    }
}

void print(int number[], int n) {
    for (int i=0; i<n; i++)
        printf("%d", number[i]);
    printf("\n");
}
```