Computer Science Foundation Exam

August 26, 2023

Section A

BASIC DATA STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

Name: _____

UCFID: _____

Question #	Max Pts	Category	Score
1	10	DSN	
2	5	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 <u>not</u> 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 <u>be neat</u>. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

1) (10 pts) DSN (Dynamic Memory Management in C)

The struct Monster_List maintains a list of monsters using a dynamically sized array of pointers to Monster. A function prototype is given for a function initializeMonster, which takes in a pointer to a Monster **that must already be pointing to memory that is allocated**, and then fills that memory with information about a default monster. Write a function getDefaultMonsters which takes in a positive integer n, creates a pointer to a Monster_List, allocates room for it, and then fills it with n default Monsters, and then returns a pointer to the Monster_list created. (Note: You must call initializeMonster in your solution.)

```
typedef struct Monster {
    // Details not necessary to solve the problem.
} Monster;

typedef struct Monster_List {
    Monster** mArray;
    int numMonsters;
} Monster_List;

// Initializes the monster pointed to by mPtr to be the default
// monster.
void initializeMonster(Monster* mPtr);

Monster_List* getDefaultMonsters(int n) {









}
```

2) (5 pts) ALG (Linked Lists)

Suppose we have a singly linked list implemented with the structure below and a function that takes in the head of the list.

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

void whatDoesItDo (node * head) {

    int tot = 0;
    while (head != NULL) {
        head->data += tot;
        tot = head->data;
        head = head->next;
    }
}
```

If we call whatDoesItDo(head) on the following list, show the list after the function has finished.

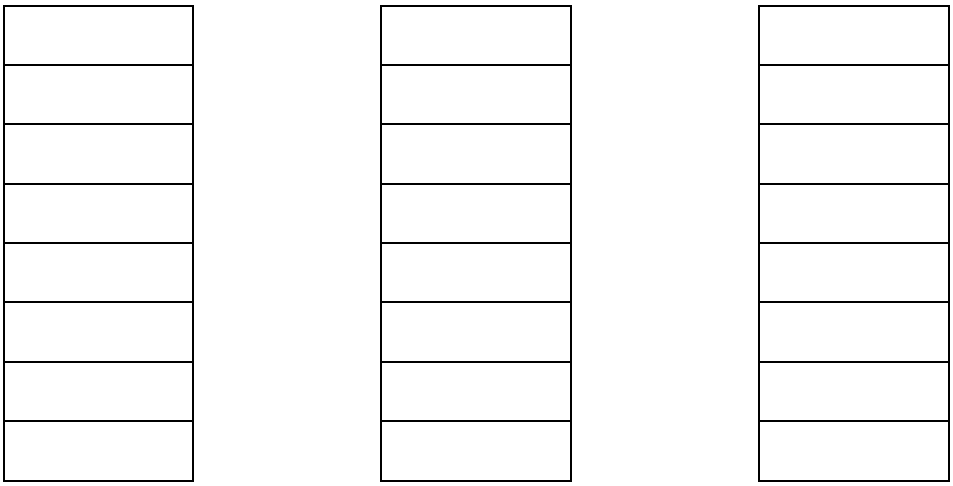
head → 3 → 9 → 7 → 1 → 4 → NULL? Please fill in the designated slots below.

→ ____ → ____ → ____ → ____ → ____ → NULL

3) (10 pts) ALG (Stack)

Convert the following infix expression to postfix using a stack. Show the contents of the stack at the indicated points (A, B, and C) in the infix expression.

```
          A                                 B                   C
4 + 3 * ( 2 – 6 / ( 9 – 7 * 1 ) + ( 2 + 3 ) * 1 ) – 8 / ( 1 + 1 )
```



A B C

Note: A indicates the location in the expression **AFTER** the multiplication and before the open parenthesis. B indicates the location in the expression **AFTER** the addition and before the open parenthesis. C indicates the location in the expression **AFTER the** close parenthesis and before the subtraction.

Resulting postfix expression:

Note: There are exactly the correct number of boxes above. These should be filled with 13 numbers and 12 operators.

Computer Science Foundation Exam

August 26, 2023

Section B

ADVANCED DATA STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

Name: _____

UCFID: _____

Question #	Max Pts	Category	Score
1	10	DSN	
2	10	DSN	
3	5	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 <u>not</u> 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 <u>be neat</u>. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

1) (10 pts) DSN (Binary Trees)

Consider using the following struct definition for a node of a binary search tree:

```
typedef struct node {
    int data;
    int height;
    struct node* left;
    struct node* right;
} node;
```

Assume that a binary search tree has been built with the data values in each struct filled in, but the heights are uninitialized. Write a **void recursive** function, `assignHeights`, **with no helper** functions, which takes in a pointer, `root`, to the root of a binary search tree, and assigns the height component of each node in the subtree pointed to by `root` to its correct height in the tree. Recall that the height of a leaf node is 0, and that more generally, the height of a node is the maximum number of links (left or right) to follow from that node to any leaf node in that subtree. (If root is NULL, then no action should be taken.)

```
void assignHeights(node* root) {

}
```

2) (10 pts) DSN (Binary Heaps)

A **mininum heap** is typically implemented with an array, with the root node (**minimum value**) being stored in index 1 of the array. To insert a new value into a heap, it's originally placed in the first open slot, followed by running a "percolate up" operation. Write a function that inserts a value into a heap in this manner. You may assume that the array is allocated to be big enough to store the newly inserted value. The function prototype is as follows:

```
void insert(int* heap, int curSize, int newVal);
```

`heap` is a pointer to an array which currently stores `curSize` number of values (but has room for at least 1 more). `newVal` is the new number to be inserted into the heap. Write this function which inserts the value newVal into this **minimum heap**. Take care to avoid infinite loops or array out of bounds issues. You may assume that index curSize+1 is in bounds for the array heap. Also, remember that index 0 of the array heap is unused. **You may not write any helper functions.**

```
void insert(int* heap, int curSize, int newVal) {




}
```

3) (5 pts) ALG (Tries)

Suppose we insert the following strings into a trie:

cantaloupe
loop
lop
lobby
cantilever

(a) How many nodes would the resulting trie contain, including the root node?

_____

(b) (b) What integer value would we store in the *count* variable in the **root node** of the resulting trie? Note that the count variable should be set to the number of valid words with end nodes stored within that sub-trie.

_____

(c) If we subsequently deleted both "lobby" and "loop" from the trie, how many total nodes would be deallocated over the course of those two deletion operations?

_____

Computer Science Foundation Exam

August 27, 2023

Section C

ALGORITHM ANALYSIS

NO books, notes, or calculators may be used,
and you must work entirely on your own.

Name: _____

UCFID: _____

Question #	Max Pts	Category	Score
1	10	ANL	
2	10	ANL	
3	5	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 <u>not</u> 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 <u>be neat</u>. 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.

1) (10 pts) ANL (Algorithm Analysis)

Consider the following problem:

Given two input values, ***n*** and ***k***, determine the number of strings of length ***n***, which only contains A's and B's, that have a run of ***k*** or more consecutive B's.

One algorithm to solve the problem is as follows:

Recursively generate each possible string of ***n*** letters, each A's and B's. These can be generated in alphabetical order, never storing more than 1 of the strings at the same time.

For each string generated, loop through the string from left to right, keeping a running tally of the current number of B's. (For example, with the string ABBABBBAAB, the running counter would update as follows 0 → 1 → 2 → 0 → 1 → 2 → 3 → 0 → 0 → 1.) If this running tally ever equals or exceeds ***k***, add 1 to a global counter storing the final result. For simplicities sake, assume that the loop completes going through the whole string before 1 is potentially added to the global counter.

With proof, determine the Big-Oh runtime of this algorithm in terms of the input parameter, ***n***.

2) (10 pts) ANL (Algorithm Analysis)

A program takes O(**nlgm**) time to process ***n*** data sets, each which have ***m*** values. For 100,000 data sets, each with 2^{16} values, the program takes 500 ms (milliseconds) to complete. Given this information, how many milliseconds would we expect the program to take to process 60,000 data sets, each with 2^{20} values?

3) (5 pts) ANL (Summations)

Solve the summation below. Your final result should be a function in terms of ***n***.

$$\sum_{k=1}^{2n} (\frac{k}{2} + 3)$$

Computer Science Foundation Exam

August 26, 2023

Section D

ALGORITHMS

NO books, notes, or calculators may be used,
and you must work entirely on your own.

Name: _____

UCFID: _____

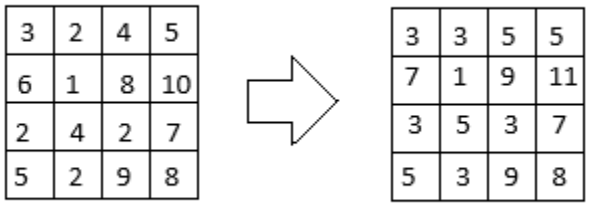
Question #	Max Pts	Category	Score
1	10	DSN	
2	5	ALG	
3	10	ALG	
TOTAL	25		

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 <u>not</u> 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 <u>be neat</u>. For each coding question, assume that all of the necessary includes (stdlib, stdio, math, string) for that particular question have been made.

1) (10 pts) DSN (Recursive Coding)

Imagine that a virus is spreading through an area we can model as a 2-dimensional integer array of size ***n*** by ***n***, where each value in the array stores the current level of virus at that grid location. The virus can be activated at a particular location. If that location currently has a virus level that is an even integer, then the virus level in that square increases by 1, and then the activation recursively activates the locations above, below, left and right of the immediate location. If the virus level in that square was odd, no change occurs and the virus doesn't spread. Luckily, once the virus level in a square increases by 1 due to an activation, it can't increase again. Here is a small example of a before and after picture of activating the virus in row 2, column 0:



Complete the **recursive** function below so that it takes in a pointer to the array storing the grid, the value of n, and the row and column of where the virus is activated, and updates the virus levels accordingly. Don't forget to make sure you don't go out of bounds of the array!

```
void activate(int** grid, int n, int row, int col) {

    if (row < 0 || row >= n) return;

    if ( _____________________________ ) return; // col out of bounds

    if ( _____________________________ ) return; // odd square

    _______________ ++;

    ___________________________________________ ;

    ___________________________________________ ;

    ___________________________________________ ;

    ___________________________________________ ;

}
```

2) (5 pts) ALG (Sorting)

Show the result after each iteration of performing Insertion Sort on the array shown below. For convenience, the result after the first and last iterations are provided. The first row of the table contains the original values of the array.

Iteration	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	16	3	18	15	1	8	12	4
1	3	16	18	15	1	8	12	4
2								
3								
4								
5								
6								
7	1	3	4	8	12	15	16	18

3) (10 pts) ALG (Base Conversion)

Perform each of the requested base conversions (the base of each number is written as a subscript):

(a) (2 pts) 347_{10} = _____ 5

(b) (2 pts) 361_{7} = _____ 10

(c) (3 pts) $3AD_{16}$ = _____ 2

(d) (3 pts) 247321_{8} = _____ 4