Computer Science Foundation Exam

August 25, 2018

Section I A

DATA STRUCTURES

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

Name: _____

UCFID: _____

NID: _____

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 (Dynamic Memory Management in C)

Consider allocating space for an array of arrays, where each of the individual lengths of the different one dimensional arrays may differ. For example, we might want 5 arrays, which have lengths 10, 5, 20, 100 and 50, respectively. Write a function `makeArray` that takes in an array of integers itself and the length of that array (so for the example above the first parameter would be the array storing 10, 5, 20, 100 and 50 and the second parameter would have a value of 5) and allocates space for an array of arrays where each of the individual arrays have the lengths specified by the values of the input array. Before returning a pointer to the array of arrays, the function should store 0 in every element of every array allocated.

```
int** makeArray(int* lengths, int numarrays) {

}
```

2) (10 pts) DSN (Linked Lists)

Consider storing an integer in a linked list by storing one digit in each node where the one's digit is stored in the first node, the ten's digit is stored in the second node, and so forth. Write a ***recursive function*** that takes in a pointer to the head of a linked list storing an integer in this fashion and returns the value of the integer. Assume that the linked list has 9 or fewer nodes, so that the computation will not cause any integer overflows. (For example, 295 would be stored as 5 followed by 9 followed by 2.) Use the struct shown below:

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

int getValue(node *head) {




}
```

3) (5 pts) ALG (Stacks/Queues)

Consider modeling cars lining up at a traffic light in a simulation. Would it be better to utilize a stack in the simulation or a queue, to store the cars? Clearly explain your choice.

Computer Science Foundation Exam

August 25, 2018

Section I B

DATA STRUCTURES

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

Name: _____

UCFID: _____

NID: _____

Question #	Max Pts	Category	Score
1	10	DSN	
2	5	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 <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 Search Trees)

Complete writing the function shown below ***recursively***, so that it takes in a pointer to the root of a binary search tree of strings, *root*, and a string, *target*, and returns 1 if the string is contained in the binary search tree and false otherwise. You may assume all strings stored in the tree contain lowercase letters only. In order to receive full credit, your function must run in O(h) time, where h is the height of the binary search tree storing all of the words.

```
typedef struct bstNode {
   struct bstNode *left, *right;
   char str[100];
} bstNode;

int search(bstNode *root, char* target){




}
```

2) (5 pts) ALG (Hash Tables)

There are two hash functions that take in strings as input shown below. Each returns an integer in between 0 and 1,000,002. (Note: 1,000,003 is a prime number.) Which of these two is a better hash function? Explain the weakness in the other function.

```
int f1(char* str) {

    int i = 0, res = 0;
    while (str[i] != '\0') {
        res = (256*res + (int)(str[i]))%1000003;
        i++;
    }
    return res;
}

int f2(char* str) {

    int i = 0, res = 0;
    while (str[i] != '\0') {
        res = (res + (int)(str[i]))%1000003;
        i++;
    }
    return res;
}
```

3) (10 pts) DSN (Tries)

The word "intention" is such that four of its prefixes, "i", "in", "intent" and "intention" are words themselves. Write a function that takes in a pointer to the root of a trie storing a dictionary of words and returns the maximum number of words that are prefixes of a single word. Use the struct definition and function prototype given below.

```
typedef struct TrieNode {
    struct TrieNode *children[26];
    int flag; // 1 if the string is in the trie, 0 otherwise
} TrieNode;

int maxNumPrefixWords(TrieNode* root) {




}
```

Computer Science Foundation Exam

August 25, 2018

Section II A

ALGORITHMS AND ANALYSIS TOOLS

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

Name: _____

UCFID: _____

NID: _____

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 <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)

Below is a program which includes a single function call to the function mysqrt. The function mysqrt includes a while loop. Give an estimate as to how many times that while loop will run during the single function call from main. (Note: a portion of credit for this question will be awarded to the theoretical analysis of the code and the rest of the credit will be awarded to applying that analysis in concert with estimation techniques without a calculator.)

```
int main() {
    printf("%.6f\n", mysqrt(1000));
    return 0;
}

double mysqrt(double n) {
    double low = 1, high = n;
    if (n < 1) {
        low = n;
        high = 1;
    }
    while (high - low > .000001) {
        double mid = (low+high)/2;
        if (mid*mid < n)
            low = mid;
        else
            high = mid;
    }
}
```

2) (5 pts) ANL (Algorithm Analysis)

An algorithm to process input data about n cities takes O($n2^n$) time. For n = 10, the algorithm runs in 5 milliseconds. How many ***seconds*** should the algorithm take to run for an input size of n = 20?

3) (10 pts) ANL (Recurrence Relations)

Use the iteration technique to solve the following recurrence relation in terms of n:

$$T(n) = 3T(n-1) + 1, for\ all\ integers\ n > 1$$
$$T(1) = 1$$

Please give an exact closed-form answer in terms of n, instead of a Big-Oh answer.

(Note: A useful summation formula to solve this question is $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$.)

Computer Science Foundation Exam

August 25, 2018

Section II B

ALGORITHMS AND ANALYSIS TOOLS

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

Name: _____

UCFID: _____

NID: _____

Question #	Max Pts	Category	Score
1	10	DSN	
2	5	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 <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)

The code below returns the number of zeros at the end of n!

```
int zeros(int n) {
    int res = 0;
    while (n != 0) {
        res += n/5;
        n /= 5;
    }
    return res;
}
```

Rewrite this method ***recursively***:

```
int zeros(int n) {









}
```

2) (5 pts) ALG (Sorting)

Show the state of the array below for each iteration of ***selection sort***:

Original	13	27	12	9	88	45	22	31
1st iteration	13	27	12	9	31	45	22	88
2nd iteration								
3rd iteration								
4th iteration								
5th iteration								
6th iteration								
7th iteration	9	12	13	22	27	31	45	88

3) (10 pts) ALG (Backtracking)

We define a number as Digit Sum Divisible if, for each value of i, the sum of its first i digits is divisible by i. For example, the number 6451 is Digit Sum Divisible because 6 is divisible by 1, 6 + 4 = 10 is divisible by 2, 6 + 4 + 5 = 15 is divisible by 3 and 6 + 4 + 5 + 1 = 16 is divisible by 4. Consider writing a backtracking function that outputs all Digit Sum Divisible numbers of a particular length given a particular prefix. This function would take in a prefix, such as "64" and a number of digits left to add to it (for this example, 2), and the function would print out each Digit Sum Divisible number starting with the digits 64 that are four digits long. The function would use backtracking because instead of adding each possible digit to the given prefix and making a recursive call, it would first check to see if doing so would maintain the divisibility requirement for the next length. (In this example, 640 would be skipped since 6 + 4 + 0 = 10 and 10 isn't divisible by 3.) ***Write out a tree structure that shows each unique prefix that occurs for each recursive call for the specific function call with the prefix 64 and 2 digits left to add to it.*** (Note: The root node of your tree should be 64, each child of 64 should be a three digit number, and each child of those children should have a four digit number. There should be eight leaf nodes in the tree, so make sure to leave enough room for eight nodes at the bottom level. These leaf nodes are the eight numbers the function would print out for this specific call.) **Note: Please do NOT write any code for this problem, just write out the underlined specified task above.**