Computer Science Foundation Exam

August 8, 2014

Section I A

COMPUTER SCIENCE

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

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	DSN	7	
2	10	ANL	7	
3	10	ALG	7	
4	10	ALG	7	
5	10	ALG	7	
TOTAL	50			

You must do all 5 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.

1) (10 pts) DSN (Recursive Functions)

An odd-recursive palindrome is a palindrome of the form WxW, where W is an odd-recursive palindrome and x is any letter in the alphabet. (Note: A palindrome is a string that reads identically forwards and backwards.) We define any single letter as an odd-recursive palindrome as well. Examples of odd-recursive palindromes are 'a', 'tat', 'abacaba' and 'abacabazabacaba'. Write a function that takes in a string, a starting index into that string, inclusive, and an ending index into that string, exclusive, and returns true if and only if the designated substring is an odd-recursive palindrome. You may assume that all characters in str are letters and that sIndex and eIndex are valid starting and ending indices. Fill out the function prototype provided below. Note: You may not use any functions in the string.h library.

```
int oddRecPal(char* str, int sIndex, int eIndex) {

    // 1 pt - positive base case
    if (eIndex - sIndex == 1) return 1;

    // 1 pt - negative base case
    if ((eindex-sIndex)%2 == 0) return 0;

    // 3 pts - recursive call, could be to right also...
    int left = oddRecPal(str, sIndex, (sIndex+eIndex)/2);

    // 1 pt - recursive fail
    if (!left) return 0;

    int i, offset = 1+(eIndex-sIndex)/2;

    // 3 pts - fail case left and right aren't equal.
    for (i=sIndex; i<(sIndex+eIndex)/2; i++)
        if (str[i] != str[i+offset])
            return 0;

    // Passes all tests
    return 1;
}
```

2) (10 pts) ANL (Summations and Algorithm Analysis)

(a) (5 pts) Determine the value of the following sum in terms of n: $\sum_{i=n+1}^{2n}(3i-2)$

$$\sum_{i=n+1}^{2n}(3i-2) = \left(3\sum_{i=n+1}^{2n} i\right) - \sum_{i=n+1}^{2n} 2$$
$$= 3\left(\sum_{i=1}^{2n} i - \sum_{i=1}^{n} i\right) - 2n$$
$$= 3\left(\frac{2n(2n+1)}{2} - \frac{n(n+1)}{2}\right) - 2n$$
$$= 3\left(\frac{4n^2+2n-n^2-n}{2}\right) - \frac{4n}{2}$$
$$= \frac{9n^2-n}{2}$$
$$= \frac{n(9n-1)}{2}$$

Grading: 1 pt first split, 1 pt second split, 2 pts eval sum to I, 1 pt final answer

(b) (5 pts) Determine the run time of the code segment shown below, in terms of n. Provide your answer as a Big-Theta bound.

```
int n;
scanf("%d", &n);
int i, step = 1, total = 1;

for (i=0; i<n*n; i+= step) {
    total++;
    step += 2;
}
```

i successively equals 0, 3, 8, 15, 24, ... These values are 1 less than the perfect squares. (Note: We can formally prove this pattern by showing the sum of the first n odd positive integers is n^2.) Thus, if n is 5, the loop runs 5 times. More generally, no matter what n is, i will equal $n^2 - 1$ at the beginning of the n^{th} iteration and will be larger than n^2 after the completion of the n^{th} iteration. Thus, the run time of this code segment is $\theta(n)$.

Grading: 1 pt answer, 4 pts explanation

3) (10 pts) ALG (Stacks)

(a) (8 pts) Convert the following infix expression to an equivalent postfix expression. Show the state of the operator stack at each of the indicated points:

((2 + 3 * [A] 6) / (9 - 8 / 2 [B]) + 9) * [C] 7

A	B	C
	/	
*	-	
+	(	
(	/	
(	(	*

Final Postfix Expression:

2 3 6 * + 9 8 2 / - / 9 + 7 *

Grading: 2 pts for each stack, 2 pts for final expression, award a whole number of points. Give partial on any part that is roughly 1/2 or more correct and don't double count points off for cascading errors.

(b) (2 pts) What is the value of the postfix expression that is the result of part (a)?

91 (Grading: 2 pts all or nothing.)

4) (10 pts) ALG (AVL Trees)

(a) (5 pts) Show the result of inserting 8 into the AVL tree shown below:

```
        22
       /    \
    12       27
   /  \        \
  6   20        33
 / \
4   9
```

An imbalance occurs at the 12 so we must restructure the trio 12, 6, 9:

```
        22
       /    \
     9       27
   /  \        \
  6   12        33
 / \     \
4   8    20
```

Grading: 1 pt for keeping 22, 27, 33 where they are, 1 pt for making 9 child of 22, 1 pt for placement of 12, 1 pt for placement of 6, 1 pt for rest

(b) (5 pts) Show the result of deleting 27 from the original AVL tree shown in part (a).

An imbalance occurs at the 22, so we must restructure the trio 6, 12, 22:

```
        12
       /    \
    6        22
   / \      /  \
  4   9   20   33
```

Grading: 1 pt for 12 at root, 2 pts for left of 12, 2 pts for right of 12

5) (10 pts) ALG (Base Conversion)

(a) (5 pts) Convert $AB38_{16}$ to binary.

Each hex "digit" converts to four bits, so we can convert directly:

$AB38_{16}$ = $1010\ 1011\ 0011\ 1000_2$

Grading: 1 pt for each hex digit, 1 pt bonus for getting all four correct. 4 pts for writing backwards

(b) (5 pts) Convert 2014 in base 10 to base 5.

```
5 | 2014
5 |  402   R 4
5 |   80   R 2
5 |   16   R 0
5 |    3   R 1
       0   R 3
```

Thus, $2014_{10} = 31024_5$.

Grading: 1 pt for each digit, decide other partial as you see fit.