Computer Science Foundation Exam

May 8, 2009

Computer Science

Section 1A

Name: Grading Criteria

PID: ____________________

	Max Pts	Type	Passing Threshold	Student Score
Q1	**11**	**DSN**	**8**	
Q2	**10**	**ANL**	**7**	
Q3	**10**	**ALG**	**7**	
Q4	**10**	**ALG**	**7**	
Q5	**9**	**ALG**	**6**	
Total	**50**		35	

You must do all 5 problems in this section of the exam.

Partial credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. Do your rough work on the last page.

1) (11 points) **Recursion** Write a **recursive** function that encrypts a string using a Caesar cipher. For those of you unfamiliar, a Caesar cipher works by shifting each letter three places in the alphabet, so 'a' becomes 'd', 'b' becomes 'e', 'c' becomes 'f', and so on. Letters at the end of the alphabet wrap around, so 'x', 'y', and 'z' become 'a', 'b', and 'c', respectively. You may assume that the string consists entirely of lowercase letters. As an example, if your function is passed a string containing "computer" then after your function is called, it should contain "frpsxwhu".

```
void caesar_cipher(char* str, int length)
{
```

Solution 1:

```
     if(length <= 0)
          return;
     str[length - 1] += 3;
     if(str[length - 1] > 'z')
          str[length - 1] -= 26;
     caesar_cipher(str, length - 1);
```

Solution 2:

```
     if(str[0] == '\0')  // Something like this is a little iffy, but it should be accepted
          return;
     str[0] = ((str[0] - 'a') + 3) % 26 + 'a';
     caesar_cipher(str + 1, length - 1);
```

Grading Criteria:

There are many ways to approach this problem. Be reasonable when grading.
Base case – 3 points
Dealing properly with individual characters – 4 points
Making a proper recursive call – 4 points

```
}
```

2) (10 points) **Summations**
a) Consider the following code fragment:

```
for(i = 4; i <= n * n; i++) {
      sum = sum + 5 + i + n;
      for(j = 1; j <= 3 * i; j++) {
            sum = sum + n + j;
      }
}
```

Write, but don't solve, a summation to describe the number of additions (not counting increments) performed by that code fragment in terms of the variable *n*.
b) Obtain a simplified closed form solution for the following summation:

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

Solution:
a)

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

b)

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

$$=2\frac{2n(2n+1)}{2}-2\frac{n(n+1)}{2}+6n^3-3n^3=2n(2n+1)-n(n+1)+3n^3$$

$$=n\left(2(2n+1)-(n+1)+3n^2\right)=n\left(4n+2-n-1+3n^2\right)=n\left(3n^2+3n+1\right)$$

Grading Criteria:
a)
Correct bounds on the outer summation – 2 points
Correct bounds on the inner summation – 2 points
Answer is otherwise correct – 1 point

b)
Splitting the summation into pieces – 2 points
Applying the correct formulas to the resulting pieces – 2 points
Simplifying the resulting closed form

---

3) (10 points) **Stack Applications** Transform the following infix expression into its equivalent postfix expression using a stack. Show the contents of the stack at the indicated points 1, 2 and 3 in the infix expressions.

```
                    1                       2                      3
(   A   *   (   B   +   C   )   -   D   )   /   (   E   +   F   )  -   G
```

Solution:

+		
(		
*		
(	/	-
1	2	3

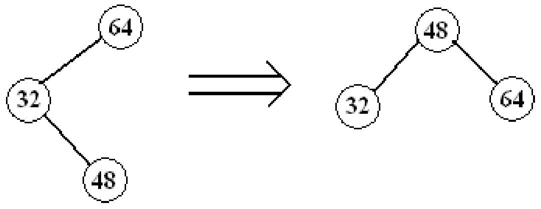
Resulting postfix expression:

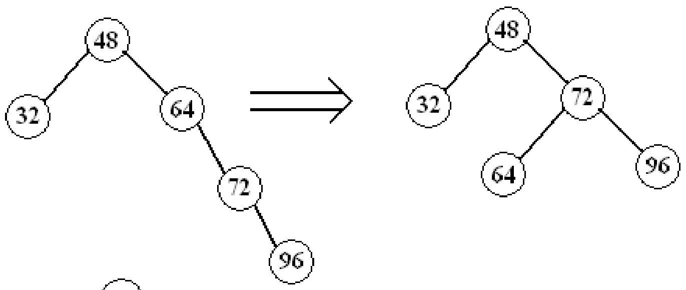
A	B	C	+	*	D	-	E	F	+	/	G	-							

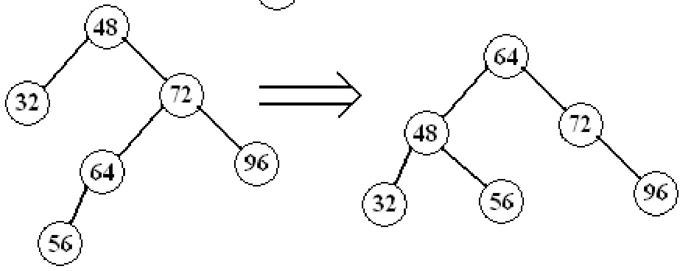
Grading Criteria:
Each correct stack – 2 points
Resulting expression – 4 points

4) (10 points) **AVL Trees** Construct an AVL tree by inserting integers into an initially empty tree in the following order: 64, 32, 48, 72, 96, 56. Draw the state of the tree before and after each necessary rotation. Be sure to draw the final state of the tree.

Solution:



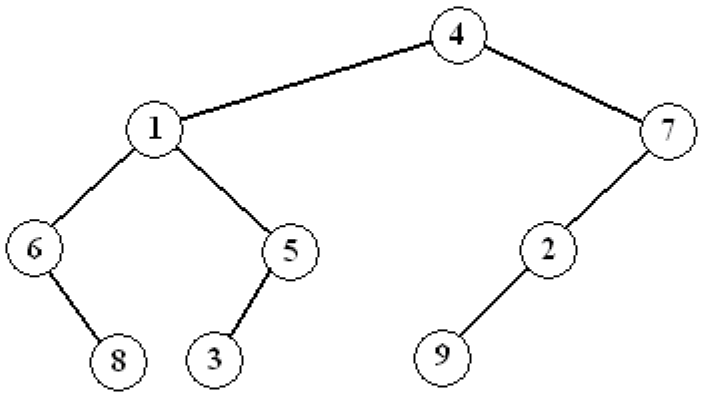




Grading Criteria:
3 points per rotation
1 point for having the correct final tree

---

5) (9 points) **Binary Tree Traversals**



Give the preorder, inorder, and postorder traversals of the binary tree shown above.

Solution:

Preorder:

4 1 6 8 5 3 7 2 9

Inorder:

6 8 1 3 5 4 9 2 7

Postorder:

8 6 3 5 1 9 2 7 4

Grading Criteria:

3 points per traversal