**COP 3502 Exam #2 - Page 1 - Algorithm Analysis (7/10/2012)**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) (10 pts) Use the iteration technique to find the **exact closed form solution** for T(n) to the following recurrence relation:

 T(n) = T(n – 1) + n, T(1) = 1

2) (10 pts) Use the Master Theorem to solve the following five recurrence relations. Provide your answer in order notation.

a) $T\left(n\right)=3T(\frac{n}{2}$) + n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b) $T\left(n\right)=4T\left(\frac{n}{4}\right)+n$ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

c) $T\left(n\right)=8T\left(\frac{n}{3}\right)+ n^{2}$ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

d) $T\left(n\right)=8T\left(\frac{n}{4}\right)+n^{1.5}$ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

e) $T\left(n\right)=9T\left(\frac{n}{3}\right)+n^{1.5}$ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) (5 pts) What is the run-time of the following function in terms of the formal parameters a and b? You may assume that both a and b are non-negative.

int function(int a, int b) {

 if (a == 0 && b == 0)

 return 0;

 if (a%2 != b%2)

 return a%2 - b%2;

 return function(a/2, b/2);

}

**COP 3502 Exam #2 - Page 2 - Linked Lists, Stacks, Queues (7/10/2012)**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) (5 pts) Determine the value of the following postfix expression:

9 8 + 2 3 + - 4 / 7 8 2 4 + - \* \*

\_\_\_\_\_\_\_\_\_\_\_

2) (10 pts) Write a **recursive** function that takes in a pointer to the front of a linked list of integers and returns the sum of every other node, starting with the second. If the list is empty or contains one element, return 0. (Note: If you write an iterative function, the maximum credit awarded will be 6 points.) Use the struct definition and function prototype provided below:

struct ll {

 int data;

 struct ll\* next;

};

int sumEveryOther(struct ll\* front) {

}

3) (10 pts) Complete the program below so that it reads in a file ("lines.txt") storing information about enqueues and dequeues into two queues and prints out one line for each enqueue and dequeue. You are guaranteed that all operations specified are possible, so you don't need to check to see if there is enough memory for an enqueue or if a dequeue is being performed on an empty queue. The file format is as follows: the first line contains a single positive integer, *n*,representing the number of operations in the file. Each of the following *n* lines will either contain instructions for an enqueue or a dequeue. The first number on each line will contain an integer, 1 or 2, specifying which queue is being operate on. The second number on each line will contain a second integer, 1 or 2, specifying either an enqueue (1) or dequeue(2). If the second integer is 1 (enqueue), then a third positive integer will be on the line specifying the number to be enqueued. For each action, output a line with one of the two following formats:

Enqueue <item> into line <linenumber>.

Dequeue <item> out of line <linenumber>.

Fill in the scaffold below to complete the task:

#include <stdio.h>

void enqueue(struct queue\* q, int item);

int dequeue(struct queue\* q);

void initialize(struct queue\* q);

int main() {

 struct queue q1, q2;

 initialize( \_\_\_\_ );

 initialize( \_\_\_\_ );

 FILE\* ifp = fopen("lines.txt", "r");

 int i, n;

 fscanf(ifp, "%d", &n);

 for (i=0; i<n; i++) {

 int q\_num, op;

 fscanf(ifp, "%d%d", &q\_num, &op);

 }

 fclose(ifp);

 return 0;

}

**COP 3502 Exam #2 - Page 3 - Binary and AVL Trees (7/10/2012)**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) (10 pts) A binary tree can store an arithmetic expression as follows: All leaf nodes store operands, while all non-leaf nodes store operators. Each non-leaf node must have two children. The value stored in such a tree is as follows: if it's a leaf node, the value is just the value stored in the node. Otherwise, the result of carrying out the operation stored in the root with the value of the left and right trees, in that order. For example, the tree

 -

 / \

 7 +

 / \

 2 3

evaluates to 2, since the left of the root evaluates to 7 and the right evaluates to 5.

Consider storing expressions with only positive numbers in such a tree. If the integer -1 is stored in a node, this indicates addition and if the integer -2 is store in a node, this indicates subtraction. Complete the function below so that it returns the value of a tree stored in this manner. You may assume that the pointer passed to the function points to a non-empty valid expression tree, as described above. Use the struct provided.

struct bintreenode {

 int data;

 struct bintreenode\* left;

 struct bintreenode\* right;

};

int evaluate(struct bintreenode\* root) {

}

2) (10 pts) Insert the following items into an AVL tree, in the order shown: 5, 10, 15, 30, 20 and 17. Put a box around what the tree looks like after each insert. Thus, you should have 6 boxes around valid AVL trees with 1, 2, 3, 4, 5, and 6 nodes respectively.

3) (5 pts) What is the result of running a post-order traversal on the binary tree shown below:

 30

 / \

 20 40

 / / \

 5 33 50

 / \ / \ /

 2 16 31 37 45

 \

 47

\_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_ , \_\_\_\_

**COP 3502 Exam #2 - Page 4 - Sorting (7/10/2012)**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) (5 pts) What is the worst-case run-time of each of the following sorts, for sorting n elements?

 a) Bubble Sort \_\_\_\_\_\_

 b) Quick Sort \_\_\_\_\_\_

 c) Selection Sort \_\_\_\_\_\_

 d) Merge Sort \_\_\_\_\_\_

 e) Insertion Sort \_\_\_\_\_\_

2) (5 pts) Show the result of running the partition on the following array, as shown in class.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
| array | 12 | 3 | 4 | 18 | 16 | 8 | 7 | 9 | 11 | 17 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
| array |  |  |  |  |  |  |  |  |  |  |

3) (5 pts) Consider running Merge Sort on the following array. What are the contents of the array right BEFORE the last Merge occurs?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
| array | 12 | 3 | 16 | 18 | 4 | 8 | 11 | 17 | 7 | 9 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
| array |  |  |  |  |  |  |  |  |  |  |

4) (10 pts) Consider the following function that sorts its input array, as long as the values in the array are within the range [0, 100000):

// Pre-condition: All values in array are in between [0, 100000).

void diffsort(int array[], int length) { // line 1

 int freq[100000]; // line 3

 int i;

 for (i=0; i<100000; i++) // line 6

 freq[i] = 0; // line 7

 for (i=0; i<length; i++) // line 9

 freq[array[i]]++; // line 10

 int index = 0, j; // line 12

 for (i=0; i<100000; i++) { // line 13

 for (j=0; j<freq[i]; j++) { // line 14

 array[index] = i; // line 15

 index++; // line 16

 }

 }

}

a) (5 pts) Consider a slightly generalized of this algorithm that takes in an array of integers in the range [0, m) with n values in it. (In this version, m is set to 100000.) What is the run-time of this algorithm in terms of both n and m? Give a brief justification for your answer.

b) (5 pts) What type of error would potentially occur if the input array to the diffsort function contained a value outside of the range [0, 100000)? On what line would that error occur?