**Computer Science I – Final Exam Multiple Choice Version *A* (8/2/2012)**

**All questions are worth 4 points.**

1) Which of the following sorts is guaranteed to run in O(nlgn) to sort n items?

a) Quick Sort b) Merge Sort c) Insertion Sort d) Bubble Sort

e) None of the Above

2) What is the return value of the function call lucas(7)? The function is given below:

int lucas(int n) {

 if (n == 1) return 1;

 if (n == 2) return 3;

 return lucas(n-1) + lucas(n-2);

}

a) 29 b) 47 c) 18 d) 7 e) None of the Above

3) All nodes in a doubly linked list have pointers to the previous node in the last and the next node in the list. If we only have access to the front node of a doubly linked list of n elements, what is the run time of inserting an element to the front of the list?

a) O(n) b) O(lg n) c) O(n2) d) O(nlgn) e) None of the Above

4) What is the value of the following postfix expression?

9 6 + 42 6 / - 2 3 \* +

a) 2 b) 6 c) 9 d) 14 e) None of the Above

5) What is the result of converting 0F37 from hexadecimal to decimal?

a) 1537 b) 2055 c) 3895 d) 4151 e) None of the Above

6) What is the fewest number of nodes an AVL tree with height 6 can have? Note: An AVL tree of height 1 either contains 2 or 3 nodes.

a) 32 b) 33 c) 34 d) 64 e) None of the Above

7) Which of the following lines of code dynamically allocates 1024 bytes and sets each of these bytes to 0? Note: an int uses up 4 bytes of storage.

a) int\* array = (int\*)(malloc(sizeof(int)\*256));

b)int\* array = (int\*)(malloc(sizeof(int)\*1024));

c) int\* array = (int\*)(calloc(sizeof(int)\*256));

d) int\* array = (int\*)(calloc(256,sizeof(int)));

e) None of the Above

The next four questions involve filling in the following incomplete function that is supposed to find the kth smallest value in the binary search tree pointed to by root and analyzing the function.

struct tree\_node {

 int data;

 struct tree\_node\* left;

 struct tree\_node\* right;

};

// Pre-condition: root is not NULL and points to a tree with k

// or more elements.

int get\_rank(struct tree\_node\* root, int k) {

 int numleft = /\*\*\* Q8 \*\*\*/ ;

 if (numleft == k-1)

 return /\*\*\* Q9 \*\*\*/;

 else if (numleft > k-1)

 return get\_rank(root->left, k);

 else

 return get\_rank(root->right, /\*\*\* Q10 \*\*\*/ );

}

int num\_nodes(struct tree\_node\* root) {

 if (root == NULL) return 0;

 return 1 + num\_nodes(root->left) + num\_nodes(root->right);

}

8) What segment of code should replace /\*\*\* Q8 \*\*\*/?

a) num\_nodes(root) b) root(num\_nodes)

c) num\_nodes(root->right) d) num\_nodes(root, left)

e) None of the Above

9) What segment code of should replace /\*\*\* Q9\*\*\*/?

a) numleft b) k-1 c) k d) root->data

e) None of the Above

10) What segment code of should replace /\*\*\* Q10\*\*\*/?

a) k-numleft-1 b) k-numleft c) numleft-k

d) numleft-k-1 e) None of the Above

11) If there are n items in the binary search tree with root my\_root, what is the worst case run-time of the function call get\_rank(my\_root, 1)?

a) O(1) b) O(lg n) c) O(n) d) O(n2) e) None of the Above

12) In what order does the recursive permutation algorithm shown in class print out the permutations of the letters “PUT”?

a) PUT b) PUT c) PUT d) PUT e) None of the Above

 PTU PTU PTU PTU

 TPU UPT UPT UTP

 TUP UTP UTP UPT

 UPT TUP TPU TUP

 UTP TPU TUP TPU

13) Consider inserting items into a hash table using the linear probing strategy. Let the hash function be f(x) = (x2 + 7) % 13, for a hash table of size 13. Consider inserting the following values into the table: 7, 10, 6, 3, 8 and 5. In which index would the last item inserted, 5, be stored?

a) 5 b) 6 c) 7 d) 8 e) None of the Above

14) Consider inserting items into a hash table using the quadratic probing strategy. Let the hash function be f(x) = (x2 + 7) % 13, for a hash table of size 13. Consider inserting the following values into the table: 7, 10, 6, 3, 8 and 5. In which index would the last item inserted, 5, be stored?

a) 5 b) 6 c) 7 d) 8 e) None of the Above

15) American swimmer Missy Franklin recently won a gold medal in the Olympic 100m backstroke swimming competition with a time of 58.33 seconds. How far did Missy swim in those 58.33 seconds?

a) 100 meters b) 100 kilometers c) 100 miles d) 100 football fields e) 100 light years

**COP 3502 Final Exam Free Response (8/2/2012)**

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

1) (10 pts) Solve the following recurrence relation using the iteration technique. Leave your final answer in Big-Oh notation.

a) (6 pts) Iterate the formula for three steps so that your equation is of the form $T\left(n\right)=8T\left(\frac{n}{A}\right)+\frac{B}{C}n^{2}$, where A, B and C are positive integer constants. (Note: We are counting the original formula as iterating one step.)

 $T\left(n\right)=2T\left(\frac{n}{2}\right)+n^{2}$

b) (2 pts) Make a guess for the formula for T(n) after iterating k steps:

c) (2 pts) Plugging in n = 2k, give a final answer, in terms of Big-Oh notation for the recurrence relation.

2) (10 pts) Draw the result of inserting the following items into an empty minimum heap, in this order: 17, 13, 12, 9, 22, 16, 8, 1, 14, and 2. Put a box around the state of the heap after each insertion.

3) (10 pts) Consider the problem of printing out each five digit number (that can start with 0), where each digit is distinct and the difference between consecutive digits must exceed 2. An example of such a number is 07259. We can use backtracking to print out all such numbers. Two functions are given below: a wrapper function that calls a recursive backtracking function and solves the original problem, and the recursive function that solves arbitrary sub-problems. The wrapper function is completed for you. Complete the recursive function so that the task is properly solved. A call to the wrapper function should print out all the desired numbers.

#define SIZE 5

#define NUMDIGITS 10

int print() {

 int digits[SIZE];

 int used[NUMDIGITS];

 int i;

 for (i=0; i<NUMDIGITS; i++)

 used[i] = 0;

 printRec(digits, 0, used);

}

void printRec(int digits[], int k, int used[]) {

 int i;

 if (k == SIZE) {

 for (i=0; i<k; i++)

 printf("%d", digits[i]);

 printf("\n");

 return;

 }

 // Fill in recursive code here.

}

4) (10 pts) Consider the following problem: Given an array of 16 positive integers and a difference value D, determine if there is a way to split the array into two groups of numbers (where each of the 16 numbers is in exactly one group), so that the difference in sums of the two groups is D. D is assumed to be positive. We can solve this problem using bitwise operators. The idea is as follows:

1) We try half of the subsets of these positive integers, labeled 0 through 215 – 1. Each of these numbers represents a subset. For example, the integer 73, which in binary is 0000 0000 0100 1001, represents a subset with items 9, 12 and 15. (These would be array indexes into the input array of values.) Note that alternatively, this could represent a subset with items 0, 3, and 6, which would be equally valid.

2) For each of these subsets, you can simultaneously add up the sum of the values not included. (For 73, this would be items 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13 and 14.)

3) Subtract these two values and take their absolute value. If it’s equal to D, return 1, to indicate that a solution exists. If not, continue the search. If none of the 215 pairs of differences equals 1, return 0.

Use this strategy to solve the given problem by completing the function below. You may assume that the input array to the function is exactly size 16.

#include <math.h>

int hasDifference(int array[], int D) {

 int i;

 for (i=0; i<(1 << 15); i++) {

 int sum1 = 0, sum2 = 0, mask = i, index;

 for (index=15; index>=0; index--) {

 if ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

 else

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

 mask = mask >> 1;

 }

 if ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

 return \_\_\_\_\_ ;

 }

 return \_\_\_\_\_;

}