Name: ______________________________________________

Unity Id: ____________________________________________ (typically this is the prefix of your email)

# CSC 505 Mock Midterm 2

Time: 75 Minutes

**Instructions:** You have 75 minutes to complete this exam. Extension will not be granted. Please work carefully and print answers neatly. Good luck!

**Honor Pledge:** I pledge that I have neither given nor received any aid while taking this exam. I also understand that except for writing utensils and a calculator (not programmable) no further tools – including cell phones – are allowed during the exam.

Signature: ______________________________

| ***Problem*** | ***Score*** |
|---|---|
| 1 15 points | |
| 2 13 points | |
| 3 11 points | |
| 4 10 points | |
| 5 11 points | |
| Σ 60 points | |

Name:________________________________

**Problem 1.** (15points) Prove or disprove rigorously (ie give values for c and $n_0$ that will make your argument work) using the formal definitions of O, Ω, and o (little-Oh). Here, lg indicates the binary logarithm.

a) (4 points) $n^2 + n\lg(n^2) - 2n \in \Omega(n^2)$

$n^2 + n\lg(n^2) - 2n \geq n^2$ for $n \geq 2$ since $n\lg(n^2) – 2n >= 0$ for $n \geq 2$

Hence, using c= 1 , $n_0 = 2$ we get $n^2 + n\lg(n^2) - 2n \geq c\, n^2 \geq 0$ for all $n > n_0$

OTHER CONSTANTS AND BOUNDS MIGHT BE USED HERE AS WELL.

b) (4 points) Assume $k>1$ is a constant. $\sum_{i=1}^{2n} (i)^k \in O(n^{k+1})$.

The statement is true. Since $(i)^k > 0$ for any i>0, k>1, and $(i)^k <= (2n)^k <= 2^k n^k$ for 1<=i <= 2n we have $0 < \sum_{i=1}^{2n} (i)^k <= 2n2^k n^k = 2^{k+1} n^{k+1}$ for n>1. Choose $c=2^{k+1}$, n0=1 to see $\sum_{i=1}^{2n} (i)^k \in O(n^{k+1})$.
OTHER CONSTANTS AND BOUNDS MIGHT BE USED HERE AS WELL.

c) (7 points) Rank the following functions by order of asymptotic growth; that is, find an arrangement $g_1, g_2, \ldots$ of the below functions with $g_1 \in \Omega(g_2)$, $g_2 \in \Omega(g_3)$ …. **If functions are asymptotically equivalent, i.e. $g_k \in \Theta(g_{k+1})$ mark them by a "*".** Here, lg indicates the binary logarithm.

$9!$, $n^5+\lg(n!)+(4/5)^n$, $n^{1.01}$, $2^n$, $(99/100)^{n/2}$, $\lg(n!)$, $2^{5\lg(n)}$

| | | | | | | |
|---|---|---|---|---|---|---|
| $2^n$ | $n^5+\lg(n!)+(4/5)^n$ * | $2^{5\lg(n)}$ * | $n^{1.01}$ | $\lg(n!)$ | $9!$ | $(99/100)^{n/2}$ |

Name:________________________________

**Problem 2** (13 points). Miscellaneous.

a) (8 points). Suppose you have to choose among three algorithms to solve a problem:

• Algorithm A solves an instance of size n by recursively solving four instances of size *n*/2, and then combining their solutions in time $2n^2$.

• Algorithm B solves an instance of size *n* by recursively solving thirty instances of size *n*/3, and then combining their solutions in time $n^2$.

• Algorithm C solves an instance of size *n* by recursively solving two instances of size *n*-1, the algorithm does not need additional time for combining the solutions of the recursive calls.

Where possible, give asymptotic big-theta bounds for the worst-case running time of the algorithms. Using these bounds, which algorithm is preferable? Assume that for small instances all three algorithms need constant time *c*. **To receive full marks justify your answer. If possible, use the Master Theorem, or give a justification why the Master Theorem cannot be applied. You do not have to verify the regularity condition of case 3 in the Master Theorem.**

Solution. For Algorithm C the MT does not apply, because it recurs on a problem instance of size n-1 and not n/b, the algorithm has the recurrence TC(n)=2TC(n-1) resulting in an exponential running time $\Theta( 2^n)$. Algorithm A follows the recurrence TA(n)=4TA(n/2)+2n^2. By the Master theorem case 2, Algorithm A's running time is $\Theta ( n^2 \lg(n) )$. Algorithm B follows the recurrence TB(n)=30TB(n/3)+n^2. By MT case 1, Algorithm B's running time is $\Theta ( n^{\log 3(30)} )$. Since $n^2 \lg(n)$ is in $o( n^{\log 3(30)} )$, and Algorithm C has exponential running time, Algorithm A is preferable.

b) (5 points). In class, we have analyzed COUNTING-SORT. COUNTING-SORT assumes that each of the *n* input elements is an integer in the range 0 to *k*, for some integer *k*. We assume that the input is an array *A*[1..*n*], and thus *A.length*=*n*. We require two other arrays: the array *B*[1..*n*] holds the sorted output, and the array *C*[0..*k*] provides temporary working storage. In the pseudocode below are some mistakes. Please find and correct them. Tip: there are 5 mistakes.

```
COUNTING-SORT(A, B, k)
    let C[0..k] be a new array
    for i = 0 to k
        C[i] = 1              *0
    for j = 1 to A.length
        A[C[j]] = A[C[j]] + 1         * C[A
    // C[i] now contains the number of elements equal to i.
    for i = 1 to k
        C[i] = C[i] + C[i+1]          * -
    // C[i] now contains the number of elements less than or equal to i.
    for j = A.length downto 1
        B[C[A[j]]] = C[A[j]]          * A[j]
        C[A[j]] = C[A[j]] - 1
```

Mistakes in red, corrections are marked by "*", see textbook page 195.

Name:________________________________

**Problem 3** (11 points). DFS and SCC. The directed graph G = (V, E) is described below in adjacency list format with the name of each vertex followed by the list of its (outgoing) neighbors enclosed in square brackets.

a:[b, c], b:[d], c:[b], d:[c], e:[c,f,g], f:[c,g], g:[f ]

a) (1 points) Draw G.

b) (6 points) Assume you do a depth-first search starting at vertex a. Each adjacency list is processed in the order specified by the above description. Give the discovery and finish times for each vertex and the classification of each edge as Tree (**T**), Back (**B**), Forward (**F**), or Cross (**C**) edge in the tables below. Start the clock with 1.

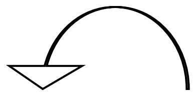
| Vertex | A | B | C | d | e | f | G |
|---|---|---|---|---|---|---|---|
| discovery time | 1 | 2 | 4 | 3 | 9 | 10 | 11 |
| finish time | 8 | 7 | 5 | 6 | 14 | 13 | 12 |

| Edge | ab | ac | bd | cb | dc | ec | ef | eg | fg | gf |
|---|---|---|---|---|---|---|---|---|---|---|
| Classification | T | F | T | B | T | C | T | F | T | B |

c) (2 points) List the vertices of each strongly connected component of G, giving each component a capital Roman numeral name such as I, II, etc.

I: a    II: b,c,d    III: e    IV: f,g

d) (2 points) Draw the directed acyclic graph formed by the strongly connected components of G.



I -> II <- III ~~<-~~-> IV

Name:________________________________

**Problem 4.** (10 points) The Stirling number $S(n,k)$ counts the number of ways to partition the set $\{1, 2, \ldots, n\}$ into $k$ nonempty subsets. For example, $S(4,2) = 7$ since there are 7 different partitions of the set $\{1, 2, 3, 4\}$ into two nonempty sets:

| | |
|---|---|
| 1. $\{1\} \cup \{2, 3, 4\}$ | 5. $\{1, 2\} \cup \{3, 4\}$ |
| 2. $\{2\} \cup \{1, 3, 4\}$ | 6. $\{1, 3\} \cup \{2, 4\}$ |
| 3. $\{3\} \cup \{1, 2, 4\}$ | 7. $\{1, 4\} \cup \{2, 3\}$ |
| 4. $\{4\} \cup \{1, 2, 3\}$ | |

$S(n,k)$ can be computed via the following recurrence:

$$S(n,k) := \begin{cases} 1 & \text{if } n=k=0 \\ 0 & \text{if } n=0 \text{ and } k>0 \text{ or } k=0 \text{ and } n>0 \\ S(n-1,k-1) + k*S(n-1,k) & \text{otherwise} \end{cases}$$

a) (5 points) For $0 \le n, k \le 3$, fill in the table below with the corresponding values for S(n,k).

| **n↓ \ k→** | **0** | **1** | **2** | **3** |
|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 |
| **1** | 0 | 1 | 0 | 0 |
| **2** | 0 | 1 | 1 | 0 |
| **3** | 0 | 1 | 3 | 1 |

b) (3 points) Complete the pseudocode for a non-recursive dynamic programming algorithm to compute $S(n,k)$.

```
FOR i=0 TO k DO
    FOR j=0 TO n DO
        IF ___i=0_AND j=0_________________ THEN S(j,i) ← 1
        ELSE
            IF ( ______i=0 AND j>0________________ ) THEN _______S(j,i) ←0_______________
            IF (    i>0   AND   j=0        ) THEN _________ S(j,i) ←0____________
            IF (    i>0   AND ___j>0______ ) THEN ___S(j,i) ← S(j-1,i-1) +i* S(j-1,i)________
RETURN ________S(n,k)_____________
```

c) (2 points) Assume k>0 is constant and n → ∞. What is the asymptotic worst-case running time of the algorithm described in problem b)? Θ(nk)

Name:______________________________

**Problem 5** (11 points). Mark the correct answer (t: true, f: false). Beware of guessing: correct answer +1pt. incorrect answer -.5pt. no answer 0pts.

a) [ t **f** ] We do not know an algorithm to solve the **0-1** knapsack problem.

b) [ **t** f ] Using a linked list weighted union implementation of the disjoint sets data structure, a sequence of *n* disjoint set operations, of which *m* are MAKE SET operations, takes O(*mn*) in worst case.

c) [ **t** f ] A MAX-HEAP can be transformed into a MIN-HEAP heap in linear time.

d) [ t **f** ] If the depth-first search traversal of an undirected graph G yields no back edge, then the graph is a tree.

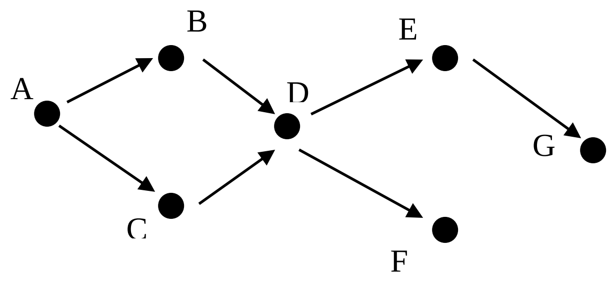
e) [ **t** f ] The **fractional** knapsack problem can be solved by a greedy algorithm.

f) [ **t** f ] Let T be a complete binary tree with *n* nodes. Finding a path from the root of T to any leave v $\in$ T using breadth-first search takes $\Omega(n)$ time in worst case.

g) (2 points) Please give a one sentence definition for the term **"stable sorting algorithm."**

**A sorting algorithm is stable if elements with the same key value remain in the same relative order.**

h) (3 points) List **ALL** topological sorts of the graph below. It is enough to give the vertex order for each sort; an example is given below.



**Topsort 1: A, B, C, D, E, F, G Topsort 2: A, B, C, D, F, E, G Topsort 3: A, B, C, D, E, G, F**
**Topsort 4: A, C, B, D, E, F, G Topsort 5: A, C, B, D, F, E, G Topsort 6: A, C, B, D, E, G, F**

Name:______________________________