Computer Science 3711
Winter 2005

Midterm Exam

February 15, 2005

Instructor:
T. Wareham

NAME: _____ STUDENT ID #: _____

- This exam is out of 60 marks.
- This exam has 7 pages (including this cover page). There are 5 questions, most of which have multiple parts.
- Please answer all questions in the space provided on this exam; if you find it necessary to continue an answer on the back of a sheet of paper, that is fine, but please make a note on the front side, *e.g.*, "answer cont'd on back".

Question		Mark
1.	6	
2.	14	
3.	9	
4.	16	
5.	15	
	60	

1. (6 marks) Circle the letters associated with the appropriate answers in the following multiple choice questions. Note that some of these questions may have more than one answer whose letter needs to be circled.

(a) (3 marks) Which of the following is true of the function $T(n) = \frac{n^3}{4}$?

a) $T(n)$ is $\Theta((n+100)^2)$ b) $T(n)$ is $O(57 \log_2 2^{n^4})$ c) $T(n)$ is $\Omega((n+2)^3)$

(b) (3 marks) Which of the following is true of the function $T(n) = 10^{\log_3 n}$?

a) $T(n)$ is $\Theta(n^{\log_2 7})$ b) $T(n)$ is $\Omega(3^{\log_{10} n})$ c) $T(n)$ is $O(n^2\sqrt{n})$

2. (14 marks)

a) (4 marks) Give the recurrence for the asymptotic worst-case time complexity of the following recursive algorithm:

```
procedure FUNKY-REC(m, n)
    if (n < 5)
        sum = 0
        for i = 1 to m do
            sum = sum + (FUNKY-ITR(n) / i) - j
        return(sum)
    else if (n > 10)
        sum = FUNKY-REC(m, n - 9) + (FUNKY-ITR(m) / 12)
        return(sum)
    else if (n > 3)
        sum = 0
        for i = 1 to n do
            sum = FUNKY-ITR(n) * l
        return(sum)
    else
        sum = 0
        for i = 1 to m do
            l = FUNKY-REC(m, n - 6)
            sum = sum + (FUNKY-ITR(m) / i) - l
        return(sum)
```

Assume that procedure `FUNKY-ITR(x)` runs in $O(\log_2 x)$ time.

b) (10 marks) Derive an upper bound for $T(n)$ using any method discussed in class or in the textbook, where

$$T(n) = \begin{cases} n^2 & n \leq 1 \\ T(n/2) + cn & n > 1 \end{cases}$$

3. (9 marks) Consider the following algorithm:

```
sum = 0;
for i = 1 to n * n do
    if (T2(i, n))
        for k = 1 to n do
            if ((k % 2) == 0)
                l = T3(n, k) - 2
            sum = (sum * j / l)
    for j = 1 to n do
        sum = sum * j
if (T1(n, sum))
    sum = sum / 2
```

a) **(3 marks)** Give the parameterized asymptotic worst-case time complexity for this algorithm.

b) **(2 marks)** Give the asymptotic worst-case time complexity of this algorithm when the T1, T2, and T3 operations require $O(n^2)$, $O(n)$, and $O(n^2)$ time, respectively.

c) **(2 marks)** Give the asymptotic worst-case time complexity of this algorithm when the T1, T2, and T3 operations require $O(n)$, $O(\log_2 n)$, and $O(log_2 n)$ time, respectively.

d) **(2 marks)** Give the asymptotic worst-case time complexity of this algorithm when the T1, T2, and T3 operations require $O(1)$, $O(1)$, and $O(1)$ time, respectively.

4. (16 marks)

Consider the following algorithm:

```
DFS-V(i, sol, U, B, VL, VU)
    if (SIZE(sol) > B)
        print("pruned: solution too big for knapsack")
    else if (VALUE(sol) > VU)
        print("pruned: solution too valuable")
    else if (i == n)
        if ((VALUE(sol) >= VL) and (VALUE(sol) <= VU))
            print(sol)
    else
        DFS-V(i + 1, sol, U, B, VL, VU)
        DFS-V(i + 1, UNION-COPY(sol, U[i]), U, B, VL, VU)
```

The problem solved here is a variant of $0/1$ KNAPSACK that prints all viable knapsack-loads of a given value val such that $VL \leq val \leq VU$. In this algorithm, U is the set of items, n is the number of items in U, B is the knapsack size bound, sol is a subset of U, VL and VU are lower and upper bounds on the summed value of the items in a solution, respectively, SIZE(sol) returns the sum of the sizes of the items in sol, VALUE(sol) returns the sum of the values of the items in sol, and UNION-COPY($sol, U[i]$) returns a copy of sol to which the ith item in U has been added. To answer this question, sketch the implicit tree of solution-nodes generated by the algorithm above (marking leaf-nodes which are printed by a circle and nodes that are pruned with a square) with the call DFS-V($0, sol, U, B, VL, VU$) when $U = \{X, Y, Z\}$ such that $size(X) = 3$, $size(Y) = 1$, $size(Z) = 2$, $value(X) = 2$, $value(Y) = 2$, and $value(Z) = 1$, sol is the empty set, and B, VL, and VU have the following values:

i) (8 marks) $B = 5$, $VL = 1$, and $VU = 3$:

ii) (8 marks) $B = 3$, $VL = 2$, and $VU = 5$:

5. (15 marks) The length of the longest common vowel-preferred subsequence (LCVS) of two strings s and s' is defined by the following recurrence:

$$D(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \max(D(i-1,j-1) + COST(s(i), s'(j)), & \\ \quad D(i,j-1) - 2, & \\ \quad D(i-1,j) - 2) & \text{otherwise} \end{cases}$$

For any two symbols x and y, function $COST(x,y)$ returns 3 x and y are both vowels, 2 if x and y are both consonants, and 1 otherwise (recall that a vowel is one of the letters $\{'A', 'E', 'I', 'O', 'U'\}$ and a consonant is any letter of the alphabet that is not a vowel). Given the above, determine the longest common vowel-preferred subsequence of the strings **AUQIF** and **TUA** – that is, fill in the dynamic programming matrix given below (including one backpointer per matrix-cell), show one of the backpointer paths that gives an optimal LCVS (given that traceback starts at the lower right-hand corner matrix-cell, *i.e.*, $D(|s|, |s'|)$, and goes back to a matrix cell in row 0 or column 0), and show the LCVS corresponding to this path, *i.e.*, the sequence of matching symbol-pairs induced by the recursive subclause $\max(D(i-1,j-1) + COST(s(i), s'(j))$ and its associated diagonal backpointers.

			A	U	Q	I	F
		j	0	1	2	3	4	5
	i							
	0							
T	1							
U	2							
A	3							