ECE608 Midterm 1 Spring 2009

Please fill in the following information.

THEN PUT YOUR EMAIL ADDRESS ON EVERY PAGE OF THE EXAM !

NAME:

Email (please put complete address):

Neatness counts. We will not grade what we cannot read.

Exam is worth 100 points. Please phrase your answer succinctly. Write your answer on the same page, on which the question is given. There are three questions and a total of seven pages. Make sure you turn in all these pages.

Do not attempt to look at other students' work. Keep your answers to yourself.
Any sort of cheating will result in a zero grade.

Read and sign the statement below. Wait for instructions to start the examination before continuing to the next page.

"I signify that the work shown in this examination booklet is my own and that I have not received any assistance from other students nor given any assistance to other students."

_____(Signature)

Q1. Answer the following questions. **(45 points total)**

(a) **(10 points)** Write a recurrence equation that describes the number of lines of output generated by COMPOSE(n). Do not forget the base case.

```
COMPOSE (n)
1. for i ← 1 to n
2.       do for j ← 1 to 1/√n          ▷ you can assume 1/√n is an integer
3.             do write( i , j, n)     ▷ print line with i, j, and n
4. if n > 0
5.    then for i ← 1 to 100
6.             COMPOSE (n/10)
```

Solution:

$$T(n)=\begin{cases}0 & if\ n=0,\\ 100T(\frac{n}{10})+\sqrt{n} & if\ n>0.\end{cases}$$

In case, assumption on step 2 is taken then the following recurrence is also valid

$$T(n)=\begin{cases}0 & if\ n=0,\\ 100T(\frac{n}{10})+n & if\ n>0.\end{cases}$$

(b) **(15 points**) Find the complexity of above recursive code using the Master Theorem if applicable. Otherwise use an alternate method to find the complexity using Θ-notation.

Solution: For

$$T(n)=\begin{cases}0 & if\ n=0,\\ 100T(\frac{n}{10})+\sqrt{n} & if\ n>0.\end{cases}$$

Since $a = 100$, $b = 10$, $n^{\log_{10}100}=n^2$, $f(n)=n^{\frac{1}{2}}$, and $f(n)=0(n^{2-\epsilon})$ for $0 < \epsilon \leq$ 1.5, by case 1 of the Master Theorem, $T(n) = \Theta(n^2)$.

Solution: For

$$T(n)=\begin{cases}0 & if\ n=0,\\ 100T(\frac{n}{10})+n & if\ n>0.\end{cases}$$

Same as above except that $f(n) = n$, and $f(n) = 0(n^{2-\epsilon})$ for $0 < \epsilon \leq 1$, by case 1 of the Master Theorem, $T(n) = \Theta(n^2)$.

(c) Order the following functions by order of growth. If one is asymptotically larger than the other state the order; if they are in the same class, that is $f(n) = \Theta g(n))$, then state that fact. Make sure to prove your answer. **(10 points total)**

i. $2^{\lg n}$ versus $3^{lg\, n}$ **(5 points)**

Solution :

Note that $2^{\lg n} = n^{\lg 2} = n$ and $3^{\lg n} = n^{\lg 3} > n^{1.5}$

$$\lim_{n\to\infty} \frac{n}{n^{\lg 3}} = 0$$

Hence, $3^{\lg n}$ grows asymptotically faster than $2^{\lg n}$.

ii. $\left(\sqrt{2}\right)^{\lg n}$ versus $\sqrt{n}$ **(5 points)**

Solution:

$$\left(\sqrt{2}\right)^{\lg n} = 2^{\frac{1}{2}\lg n} = 2^{\lg\left(\sqrt{n}\right)} = \sqrt{n}.$$

(d) (10 points) A dice with *m* faces is rolled *k* times. On the average how many time the number, *p*, will show up?

Solution: k/m

(see page 109 of text)

Q 2. (20 points) The following function (Polly) returns the value of the polynomial

$$a_n x^n + a_{n-1}x^{n-1} + \mathrm{L} \; + a_1 x + a_0$$

at a given value of *x*. Prove the function (Polly) is correct by finding and proving the appropriate loop invariant Q and evaluating Q at loop termination.

Polly(real a_n; K ; real a_0; real x)
Local variables:
integers *i,j*
  $i = n$
  $j = a_n$
  while $i \neq 0$ ***do***
    $j = j * x + a_{i-1}$
    $i := i - 1$
end while

//j now has value of the polynomial evaluation (comment line)
return *j*
end function Polly

Solution: The loop invariant Q is:
Q: $j = a_n x^{n-i} + a_{n-1}x^{n-i-1} + \mathrm{L} \; + a_i x^0$

Q(0): $j_0 = a_n x^{n-i_0} + a_{n-1}x^{n-i_0-1} + \mathrm{L} \; + a_{i_0} x^0$ true since $j = a_n$ and $i = n$ before loop is entered, so equation becomes

$a_n = a_n x^{n-n}$, or $a_n = a_n$ a

Assume Q(k): $j_k = a_n x^{n-i_k} + a_{n-1}x^{n-i_x-1} + \mathrm{L} \; + a_{i_k} x^0$

Show Q(k+1): $j_{k+1} = a_n x^{n-i_{k+1}} + a_{n-1}x^{n-i_{k+1}-1} + \mathrm{L} \; + a_{i_{k+1}} x^0$

$$j_{k+1} = j_k * x + a_{i_k-1} = (a_n x^{n-i_k} + a_{n-1}x^{n-i_k-1} + \mathrm{L} \; + a_{i_k} x^0)x + a_{i_k-1}$$
$$= a_n x^{n-i_k+1} + a_{n-1}x^{n-i_k} + \mathrm{L} \; + a_{i_k} x^1 + a_{i_k-1}$$
$$= a_n x^{n-i_{k+1}} + a_{n-1}x^{n-i_{k+1}-1} + \mathrm{L} \; + a_{i_k} x^1 + a_{i_{k+1}}$$

At termination,
$j = a_n x^{n-i} + a_{n-1}x^{n-i-1} + \mathrm{L} \; + a_i x^0$ and $i = 0$ so
$j = a_n x^n + a_{n-1}x^{n-1} + \mathrm{L} \; + a_0 x^0$

Q3 (25 points total)

(a) **(10 points)** One can generalize the definition of binary-heap (the one we studied in the class) to ***d***-heap in which every node has at most d children. Describe a labeling scheme for such a heap. In other words, if such a heap is stored in an array, for an entry located at position i of the array, find the indices for its parent and children nodes? Assume the first element of the array has index 1.

Solution: A d-ary heap can be represented in a 1-dimensional array as follows. The root is kept in A[1], its d children are kept in order in A[2] through A[d + 1], their children are kept in order in A[d + 2] through A[$d^2 + d + 1$], and so on. The two procedures that map a node with index i to its parent and to its j^{th} child (for $1 \le j \le d$), respectively, are:

D-ARY-PARENT(*i*)

return $\left\lfloor \frac{i-2}{d} + 1 \right\rfloor$

D-ARY-CHILD(*i,j*)

return $d(i-1) + j + 1$ where $1 \le j \le d$

To convince yourself that these procedures really work, verify that
D-ARY-PARENT(D-ARY-CHILD(i,j)) = i, for any $1 \le j \le d$.
Notice that the binary heap procedures are a special case of the above procedures when $d = 2$.

(b) **(15 points)** Modify the HEAPIFY algorithm (discussed in the class) for a d-heap, assuming a max heap. A precise pseudo code will suffice

Solution:

Notations:
← Assignment, ↔ Swap (or exchange)

```
d-HEAPIFY(A, i)
    largest ← i
    for j ←1 to d
    do   child_index ← d * (i – 1) + j + 1
         if  child_index > heap-size[A]
         then break
         if  A[largest] < A[child_index]
         then  largest ← child_index
    endfor
    if i ≠ largest
    then A[largest] ↔ A[i]
         d-HEAPIFY (A, largest )
```