CIS 275 Discrete Math I Summer 2019

Quiz 03

Name:

Time: Complete and submit to the instructor

Evaluation:

- As described in the syllabus, the Quiz is 20% of the overall grade.

Exercise 1: Find the complexity of the below program:

```
(A)

function(int n)
{
    if (n==1)
       return;
    for (int i=1; i<=n; i++)
    {
        for (int j=1; j<=n; j++)
        {
            printf("*");
            break;
        }
    }
```

Exercise 2:

Are each of the following true or false?

a) (a) 3 n^2 + 10 n log n = O(n log n)

b) (b) 3 n^2 + 10 n log n = Omega(n^2)

c) (c) 3 n^2 + 10 n log n = Theta(n^2)

d) (d) n log n + n/2 = O(n)

e) (e) 10 SQRT(n) + log n = O(n)

f) (f) SQRT(n) + log n = O(log n)

g) (g) SQRT(n) + log n = Theta(log n)

Exercise 1 : Find a theta notation for each expression

h) 6n + 1

i) $3n^2 + 2n \lg n$

j) 2 + 4 + 6 + … + 2n

k) $2 + 4 + 8 + 16 + \ldots + 2^n$

Exercise 2 : Find a theta notation for the number of times the statement x = x +1 is executed

a) for i = 1 to 2n

```
x = x + 1
```

b) for i = 1 to 2n

```
for j = 1 to n
    x = x + 1
```

c) for i = 1 to n

```
for j = 1 to n
  for k = 1 to n
    x = x + 1
```

Exercise 3: Find the greatest common divisor of each pair of integers:

a) 20, 40

b) $3^2 \times 7^3, 3^2 \times 7^3$

c) 0, 17

Exercise 4 : Find the least common multiple of each pair of integers :

a) 20, 40

b) 5, 25

Exercise 5: Express each binary number in decimal.

a) 100000

b) 1001

<u>Exercise 6:</u> Express each decimal number in binary.

a) 43

<u>Exercise 7:</u> Add the binary numbers.

a) 101101 + 11011

<u>Exercise 8:</u> Express the hexadecimal number in decimal.

a) B4F

<u>Exercise 9:</u> Add the hexadecimal numbers.

b) F0BA + E9AD

Formula

ANALYSIS OF ALGORITHMS

Analysis an algorithm refers to the process of deriving estimates for the time and space needed to execute the algorithm.

- To derive a theta notation, you must derive both big oh and omega notation.
- An other way to derive big oh, omega and theta estimations is to use known results:

Expression	Name	Estimate
$a_k n^k + a_{k-1} n^{k-1} +$,,, $+ a_1 n + a_0$	Polynomial	$\mathbf{\Theta(n^k)}$
$1 + 2 + \ldots + n$	Arithmetic Sum (Case k = 1 for Next entry)	$\mathbf{\Theta(n^2)}$
$1^k + 2^k + \ldots + n^k$	Sum of Powers	$\mathbf{\Theta(n^{k+1})}$
lg n!	Log n Factorial	$\mathbf{\Theta(n \lg n)}$

Definition: Let f and g be functions with domain {1, 2, 3, …}

We write $f(n) = O(g(n))$

We define f(n) is of order at most g(n) or f(n) is big oh of g(n) if there exists a positive constant C1 such that

$$|f(n)| \leq C1\,|g(n)|$$

For all but finitely many positive integers n.

We write $f(n) = \Omega(g(n))$

We define f(n) is of order at least g(n) or f(n) is omega of g(n) if there exists a positive constant C2 such that

$$|f(n)| \geq C2\,|g(n)|$$

For all but finitely many positive integers n.

We write $f(n) = \Theta(g(n))$

We define f(n) is of order g(n) or f(n) is theta of g(n) if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- ***By replacing O by Ω and "at most" by "at least", we obtain the definition of what it means for the best-case, worst-case, or average-case time of an algorithm to be of order at least g(n). If the best-case time required by an algorithm is O(g(n)) and Ω(g(n)), we say that the best-case time required by an algorithm is Θ(g(n)) .***

Introduction to number of theory

- Let m and n be integers with n and m different to zero. A *common divisor* of m and n is an integer that divides both m and n. **The *greatest common diviso***r, written **gcd(m, n)**

- Let m and n be positive integers. A *common multiple* of m and n is an integer that divides by both m and n. **The *least common divisor***, written **lcm(m, n)**

` lcd(m, n) is the smallest common multiple of m and n.

- ***<u>The decimal number system</u>****:* it represents integers using 10 symbols.

- ***<u>The binary number system</u>****:* it represents integers using bits (a bit is a binary digit, that is a 0 or a 1).

- ***<u>The hexadecimal number system</u>****:* it represents integers using 16 symbols

- The ***<u>Euclidean algorithm</u>*** is an odd, famous, and efficient algorithm for finding the greatest common divisor of two integers, The Euclidean algorithm is based on the fact that if *r = a mod b*, then **gcd(a, b) = gcd(b, r)**

Hexadecimal to Decimal Conversion Chart

Hexadecimal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15