University of Colombo School of Computing

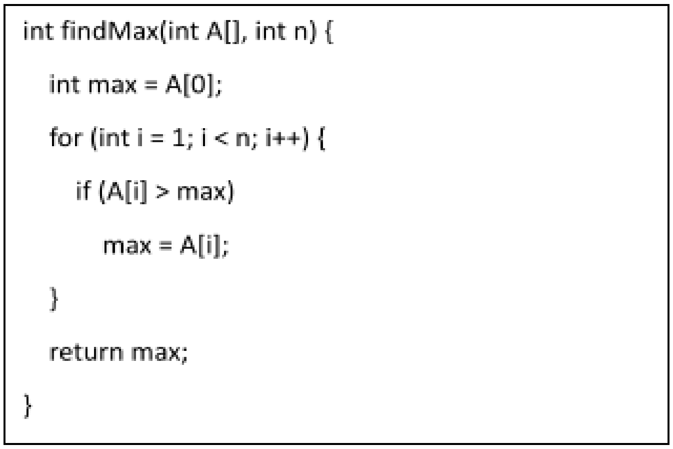
SCS 1308 – Foundations of Algorithms

Take – Home 01

Instructions

- Try the following questions and upload your answer script as a zip file to the given link in the UGVLE on/before 16th November at 10 PM.
- It's an individual task. The deadline is 10:00 PM. Submissions will be accepted until midnight with a late penalty of 5% deducted for each hour past the deadline.
- Note: Rename your zip file as indexNo_1_last_namefname.zip.

01. Define what an algorithm is and explain whether this program meets the definition of an algorithm.



```
int findMax(int A[], int n) {
   int max = A[0];
   for (int i = 1; i < n; i++) {
      if (A[i] > max)
         max = A[i];
   }
   return max;
}
```

Analyze the efficiency of this algorithm and express its time complexity using Big notation.

02. Define time complexity and explain why it is independent of machine hardware.

03. Given two algorithms with time complexities $T_1(n)=10n$ and $T_2(n)=n^2$, determine which is faster for small n and which for large n, explaining with rate-of-growth reasoning.

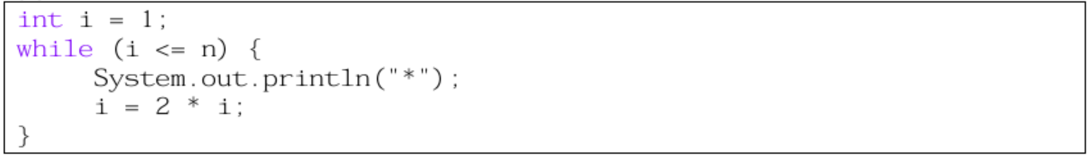
04. Remember, Big-O time complexity gives us an idea of the growth rate of a function. In other words, for a large input size N, as N increases, in what order of magnitude is the volume of statements executed expected to increase?

05. Rearrange the following functions in increasing order of their big-O complexity:

$4n^2$	$\log_3 n$	$20n$	$n^{2.5}$
$n^{0.00000001}$	$\log n!$	n^n	2^n
2^{n+1}	2^{2n}	3^n	$n \log n$
$100\ n^{2/3}$	$\log [(\log n)^2]$	$n!$	$(n-1)!$

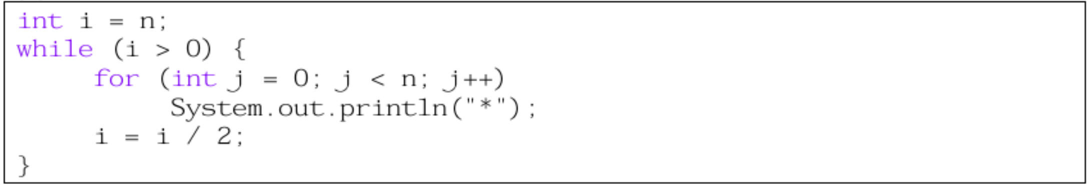
06. Find the big-O time complexity of each of the following code fragments:

(a)



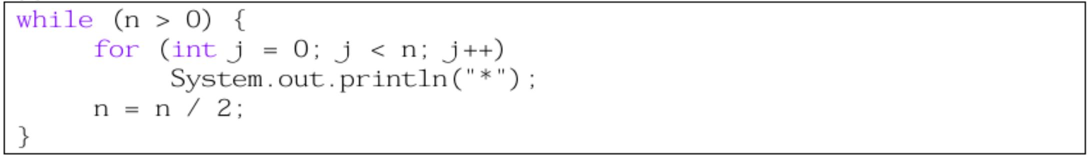
```
int i = 1;
while (i <= n) {
      System.out.println("*");
      i = 2 * i;
}
```

(b)



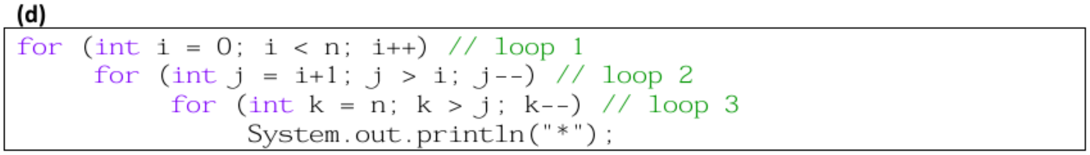
```
int i = n;
while (i > 0) {
      for (int j = 0; j < n; j++)
            System.out.println("*");
      i = i / 2;
}
```

(c)



```
while (n > 0) {
      for (int j = 0; j < n; j++)
            System.out.println("*");
      n = n / 2;
}
```

(d)



```
for (int i = 0; i < n; i++) // loop 1
      for (int j = i+1; j > i; j--) // loop 2
            for (int k = n; k > j; k--) // loop 3
                  System.out.println("*");
```

07. Draw a line from each of the five functions in the center to the best big-Ω value on the left, and the best big-O value on the right.

$\Omega(1/n)$		$O(1/n)$
$\Omega(1)$		$O(1)$
$\Omega(\log\log n)$		$O(\log\log n)$
$\Omega(\log n)$		$O(\log n)$
$\Omega(\log^2 n)$		$O(\log^2 n)$
$\Omega(\sqrt[3]{n})$		$O(\sqrt[3]{n})$
$\Omega(n/\log n)$	$1/(\log n)$	$O(n/\log n)$
$\Omega(n)$	$7n^5 - 3n + 2$	$O(n)$
$\Omega(n^{1.00001})$	$(n^2 + n)/(\log^2 n + \log n)$	$O(n^{1.00001})$
$\Omega(n^2/\log^2 n)$	$2^{\log^2 n}$	$O(n^2/\log^2 n)$
$\Omega(n^2/\log n)$	3^n	$O(n^2/\log n)$
$\Omega(n^2)$		$O(n^2)$
$\Omega(n^{3/2})$		$O(n^{3/2})$
$\Omega(2^n)$		$O(2^n)$
$\Omega(5^n)$		$O(5^n)$
$\Omega(n^n)$		$O(n^n)$
$\Omega(n^{n^2})$		$O(n^{n^2})$