American University of Armenia
CS 121 Data Structures A

Homework Assignment 1

1. (2 points) The number of operations executed by algorithms A and B is $100n^3$ and 5×2^n, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.

2. (2 points) Order the following functions by asymptotic growth rate.

$13n + 2\log n$	$5n^3$	$3n + 2n\log n$
3×2^n	$15n$	$10 + \log n$
$2n^2 + 4n^3$	1250×2^{45}	$2n^2$

3. (2 points) Give a big-Oh characterization, in terms of n, of the running time of the method shown below. Implement an alternative method *alt_example* that performs the same task in $O(n)$.

Java:

```
1  /** Returns the number of times second array stores sum of prefix sums from first. */
2  public static int example(int[] first, int[] second) { // assume equal-length arrays
3    int n = first.length, count = 0, total = 0;
4    for (int j=0; j < n; j++)                  // loop from 0 to n-1
5      for (int k=0; k <= j; k++)               // loop from 0 to j
6        total += first[k];
7    for (int i=0; i < n; i++)                  // loop from 0 to n-1
8      if (second[i] == total) count++;
9    return count;
10 }
```

C++:

```
1  /* Returns the number of times second array stores sum of prefix sums from first. */
2  int example(int first[], int second[], int n) // assume equal-length arrays
3  {
4    int count = 0, total = 0;
5    for (int j=0; j < n; j++)                  // loop from 0 to n-1
6      for (int k=0; k <= j; k++)               // loop from 0 to j
7        total += first[k];
8    for (int i=0; i < n; i++)                  // loop from 0 to n-1
9      if (second[i] == total) count++;
10   return count;
11 }
```

4. (2 points) Perform an experimental analysis that compares the relative running times of the two methods (given method *example* and your implementation *alt_example*) from the previous task. Briefly discuss your findings.

5. (2 points) Show that $f(n)$ is $O(g(n))$ if and only if $g(n)$ is $\Omega(f(n))$.

6. (2 points) Write a Java/C++ program that given a sorted array a of integers and a separate integer b, for each of the factors of b checks whether a contains that factor. In your implementation you should have a separate method that constructs the factors of a given number and another method that checks if a given array contains a given integer. What is the complexity

of each of those methods? Is it optimal? Test your program on a few inputs of growing size and make a record of the running times. How do these times reflect the asymptotic complexity of your methods?

7. (3 points) Assume that you are writing an application that needs to work with the following types of shapes:

- rectangle (which is defined by it width and height),
- square (which is defined by the length of its side),
- ellipse (which is defined by its semi-major and semi-minor axes), and
- circle (which is defined by its radius).

Design and implement a set of Java/C++ classes that represent these shapes. Your design should support the following operations:

- For each shape, it should be possible to retrieve and change its size.
- Given a set of shapes, it should be possible to identify all squares.
- Given a set of shapes, it should be possible to identify all circles.
- For each shape, it should be possible to get and print its area.

Note that your implementation should follow the principles of good object-oriented design. Include a test program that shows how your design works.