DS8001 Designs of Algorithms and Programming for Massive Data
Midterm Test 1 held on Monday, October 31, 2016
Duration: 120 minutes

Student Name	
ID	

Make sure that your examination booklet has 7 pages (including this one).

Write your answers in the space provided.

Notes:

1. **This is a closed book quiz.**
2. **No aid allowed.**
3. **All communications between students are considered harmful.**
4. **Read carefully, if in doubt, raise your hand and the instructor will come to help.**
5. **Read all questions, answer the easiest ones first.**
6. **Do not explain the concepts unless explicitly asked for, in all cases answer briefly and to the point.**

Good luck!

Short Answer (4 questions, 13 points in total)	True/False (4 questions, 1 points each)	TOTAL (max 17)

Short Answer Questions (6 questions at 4 points for each correct answer) – Answer the question briefly and to the point. Do not explain the concepts unless explicitly asked to do so. Some points may be awarded for partially correct answers. Some points will be taken off for "rambling" and for incorrect or irrelevant statements.

1. [4 points] Given the following pseudocode:

```
Input: n   // n > 1 and is an integer

z = 10
for ( i = 1 to n):
    j = 1
    while(j < i):
        j = j + 1
```

a. [2 points] Compute $T(n)$. Show your work, do not unravel the sums.

```
                               Cost       Times
z = 10                         c1         1
for ( i = 1 to n):             c2         n+1
    j = 1                      c3         n
    while(j < i):              c4         $\sum_{i=1}^{n}\sum_{j=1}^{i} 1$
        j = j + 1              c5         $\sum_{i=1}^{n}\sum_{j=1}^{i-1} 1$
```

$$T(n) = c_1 + c_2(n+1) + c_3 n + c_4 \sum_{i=1}^{n} \sum_{j=1}^{i} 1 + c_5 \sum_{i=1}^{n} \sum_{j=1}^{i-1} 1$$

In principle, it can be simplified to

$$T(n) = c_1 + c_2(n+1) + c_3 n + c_4(n^2+n)/2 + c_5(n^2-n)/2$$

b. [1 point] What is $\Theta(T(n))$? You can skip formal derivation via inequalities.

Given that we have double-sums, the leading term is quadratic and $\Theta\big(T(n)\big) = n^2$

c. [1 point] What is $\Omega(T(n))$? You can skip formal derivation via inequalities.

$\Omega(T(n))$ is a subset of $\Theta\big(T(n)\big)$. Thus, $\Omega(T(n)) = n^2$

2. [4 points] You are given a subset sum problem. That is, given a set of integers of length n, is there a subset of integers that sums up to 0?

   Example

   Set: {5, 2, 1, -7, 3}
   Subset: 5+2-7 = 0
   Answer: Yes

   a. [2 points] Assume that the worst-case scenario is the situation when you have to iterate through all possible subsets. What is the worst-case time complexity of the solution (if you solve it naïvely)? Justify your answer (at a high level, skip $T(n)$ and inequalities).

      Naïve solution needs to iterate over $2^n - 1$ subsets and perform up to n summations. Thus, the complexity, expressed in O-notation will be $O(2^n n)$.

   b. [2 points] What is the verification time of the solution? Justify your answer.

      Verification will require checking a single solution, which will require, at worst, $n - 1$ summations. Thus, the complexity of verification is $O(n)$.

3. [2 points] You need to solve a system of equations, but the problem is ill-conditioned (condition number $\kappa = 10000$). Name *two* approaches you can use to get more accurate results. *Why* would you use these approaches?

   Note that you are not limited to R language and can use the power of other languages, tools, and techniques.

   1. [1 point] If we are using floating point number – increase the precision (e.g., from float to double) or use numbers with arbitrary precision.
   2. [1 point] Resort to exact computations using computer algebra tools.

4. [3 points] You are given a Bloom filter, which is based on a bit array of length *m*. The algorithm can be summarized as follows:

```
Create bit array A of length m and populate it with 0s

add_element(element):
  // Apply k different hash functions to an element
  // Each hash function returns a value between 1 and m
  for i  = 1 to k:
    A [ $hash_i$( element ) ] = 1

is_element_present(element):
  //if at least one element in A is 0 – the element is absent
  for i  = 1 to k:
    if( A [ $hash_i$( element ) ] == 0 ): return false
  return true
```

a. [1 point] What will happen if, after a number of insertions to the bit array, all *m* bits are set to 1?

The `is_element_present` function will always return true, thus the algorithms becomes not usable from practical perspective.

b. [1 point] What can you do to prevent it from happening?

Increase the value of *m,* as it will reduce probability of filling up *A*.

c. [1 point] Why do you need to use different hash functions?

The same hash function will always map *element* to the same cell in *A*. Using multiple hash functions will increase the number of permutations, reducing the probability of incorrect results.

True/False Questions (4 questions at 1 point each) – *circle the correct answer, True or False.*

True	False	Massive/ Big data is a moving target, the threshold for what is considered massive is changing with time
True	False	Based on Gödel’s theorem, there exists more problems than solutions
True	**False**	NP class is defined as follows: If a solution resulting in “no” answer is given to me, then I can verify it in a polynomial time
True	**False**	Probabilistic algorithm will always produce correct result

[Scrap paper, do not remove]