KOÇ UNIVERSITY
CMSE 501 - Intro. Computational Sci. & Eng.
Midterm I December 7, 2010
Duration of Exam: 3 hours
Instructor: Mehmet Sayar

Your final output should include only the python programs which should be named as "q#_username.py", where # is the question number and username is YOUR USER NAME. You will get **-5** points if the filenames are NOT CORRECT.

In order to get partial credit turn in your program even if it does not give correct results due to bugs. Grading will be based on not only the correctness of the code but also good programming practices and efficiency of the code.

You are NOT allowed to connect to the internet during this exam. An open internet connection will be interpreted as cheating and you will receive zero from this exam and punished by appropriate disciplinary action.

0. (2 points) You are allowed to solve only 3 out of 4 questions below. The maximum grade you can get from this exam is 102 points. Write the number of the question you are leaving blank here: ____

1. (20 points) Write a program which generates 10 numbers according to the following formula and prints them to the standard output:

$$x_k = (a\ x_{k-1} + b\ x_{k-1}^2)(mod\ M) \tag{1}$$

where a=25, b=2, M=100, and the seed $x_0 = 1$.

2. (30 points) Bubble sort, also known as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list.

   The code "bubblesort.py" contains python implementation of this algorithm.

   The bubble sort algorithm can be easily optimized by observing that the largest elements are placed in their final position in the first passes. Or, more generally, after every pass, all elements after the last swap are sorted, and do not need to be checked again.

   Modify the code "bubblesort.py" to benefit from this optimization.

3. (30 points) The program "sparsemat-buggy.py" contains an efficient implementation of the sparse matrix multiplication using dictionaries to store non-zero elements of a sparse matrix. However the function "sparsemult" contains several bugs. Debug this code.

4. (50 points)

a) Generate a class **Cone**. The constructor for the class should take two arguments from the user, named as **radius** and **height** (**h** and **r** in figure). These attributes should be equal to **1** if the user does not provide any value.

b) Add a method named as **surfarea**, which returns the surface area of the cone according to the formula in figure.

c) Add a method named as **volume**, which returns the volume of the cone according to the formula in figure.

d) Add an addition method to the **Cone** class, named as **add** which works as follows. Suppose **Cone1** and **Cone2** are objects which belong to class **Cone**. When the user issues a command as:

$$Cone3 = Cone1.add(Cone2) \tag{2}$$

the method returns a new **Cone** object **Cone3** which is also an object of class **Cone** such that: the height of **Cone3** is the average of the heights of **Cone1** and **Cone2**. The radius of **Cone3** is chosen such that its volume is equal to the sum of the volumes of **Cone1** and **Cone2**.

Test class **Cone** with a driver program.