ESC101 : Fundamental of Computing
Lab 8 for all sections for the week 13-17 October 2008

Note :

1. This assignment is common for the entire class of ESC101. This is also perhaps the most nontrivial programming assignment given so far in this course. It is expected that you will put your sincere efforts in this assignment.

2. The assignment has to be done in a group of two, but the two students **must** be from the same section. However, if you wish, you may do it alone.

3. On your lab day, you have to be present there and explain to your TA the design of your class and the way you are going to implement its attributes and methods. You are expected to provide some justification for your design and implementation.

4. You have to email the code of the assignment to respective TA by **midnight of 18th October**. If you fail, you will get zero marks for this assignment.

5. You will have to use arrays in this assignment. But you won't require anything more than what we discussed in the last Lecture before the second mid semester exam.

The aim of this assignment is to help you use your skills to design classes for handling integers longer than `long`. As you are aware that if we add, subtract or multiply two variables of integers types (`byte, short, int, long`), then sometimes the resulting number may be incorrect due to overflow. You may recall the assignment of Lab 5 where you developed adhoc strategies and methods to avoid such overflow. In case you are unable to recall, please go through all the assignments of Lab 5. Sometimes, there are problems where such overflow occurs at an intermediate stage. One such problem which was discussed in one of the tutorial was computation of $\binom{n}{i}$. We must have some way to take care of numbers beyond the range of an integer types. In particular, we have to have a way to handle arbitrarily long integers. This assignment can be seen as a step in this direction.

You want to design a class `LargeInt` which will take care of all such overflow. For this objective, we would use a different representation for storing integers of arbitrary length and we shall use arrays for this purpose - you will store digits of a number in an array. The representation of the attributes have to be kept hidden from outside. You may use additional attributes if you wish. The constructors and methods of the class `LargeInt` should be the following.

Constructors :

- `LargeInt(long n)`
  to construct a `LargeInt` number which stores `n`.
- `LargeInt()`
  `to construct a LargeInt number which stores 0.`
- `LargeInt(LargeInt L)`
  `to construct a LargeInt number which is identical to LargeInt number referenced by L.`

Methods : You have to design the methods whose signature and explanation is given below. You have to decide whether they have to be static/non-static and what should be their access control : public/private/package ?

- `add(LargeInt L, LargeInt M)`
  return the reference to a LargeInt number which is formed by adding `LargeInt` referenced by `L` and `LargeInt` referenced by `M`.

- `add(LargeInt L, long n)`
  return the reference to a `LargeInt` number which is formed by adding LargeInt number referenced by L and `long` number n.

- `PrintLargeInt(LargeInt L)`
  to print the LargeInt number L.

- `subtract(LargeInt L, LargeInt M)`
  return the reference to a LargeInt number which is formed by subtracting `LargeInt` number referenced by `M` from `LargeInt` referenced by L.

- `subtract(LargeInt L, long n)`
  return the reference to a LargeInt number which is formed by subtracting `long` number `n` from `LargeInt` number referenced by `L`.

- `bigger(LargeInt L, LargeInt M)`
  returns true if `LargeInt` referenced by `L` is bigger than the `LargeInt` number referenced by `M`, and false otherwise.

- `bigger(LargeInt L, long n)`
  returns true if `LargeInt` number referenced by `L` is bigger than the `long` number `n`, and false otherwise.

- `equal(LargeInt L, LargeInt M)`
  returns true if `LargeInt` number referenced by `L` is equal to the `LargeInt` number referenced by `M`, and false otherwise.

- `equal(LargeInt L, long n)`
  returns true if `LargeInt` referenced by `L` is equal to the `long` number `n`, and false otherwise.

- `Multiply(LargeInt L, LargeInt M)`
  returns the reference to a LargeInt number which is formed by multiplying `LargeInt` number referenced by `L` and `LargeInt` number referenced by `M`.

- `Multiply(LargeInt L, long n)`
  returns the reference to a LargeInt object which is formed by multiplying `LargeInt` number referenced by `L` and `long` number `n`.

Using the class `LargeInt` to design the following two very short programs.

1. the first program receives two positive integers in the range of `int` from command line and prints the sum of their cubes.

2. the second program receives a positive integer `n` less than 100 from command line, and prints factorial of `n`.

Note for the students : You have to apply your analytical ability to design and implement the class `LargeInt`. You will have to give justification for the design and implementation you develop. **Note for the Tutors and TAs :** This assignment is to be graded out of 20 marks. Six marks will be based on the design of the LargeInt class and the justification given by the students for their solution during the lab hour. Each student should be able to appreciate the significance of the problem. The students have to answer your specific questions and explain their solution verbally during the lab hour. The remaining marks have to be awarded after going through the code of the

assignment and running the program. Please note that the assignments will be mailed to respective TA by 18th October midnight. Since the students may form group within a section, so in case two students belong to different TAs of a section, please make sure that the final assignment is mailed to and graded by exactly one TA.