

# Project 1: Big Integer

CMPE 250, Data Structures and Algorithms, Fall 2016

Instructor: H. L. Akın, A. T. Cemgil

TA: Çağatay Yıldız

SA: Rıza Özçelik

Due: TBA

## Description

Welcome to the most entertaining class in department and, of course, welcome to your projects. As your first project, since you are engineers and the word engineer means "who does Math" in Turkish, you will be doing Math! You are going to be implementing addition and multiplication for very big integers that are represented as a `LinkedList`.

To do this you will be provided some code written by assistants and you are going to implement the missing part.

The files you will be provided are: `Node.h`, `Node.cpp`, `LinkedList.h`, `LinkedList.cpp`, `BigInteger.h` and `main.cpp`

The file you should fill in is: `BigInteger.cpp`

In other words, you are expected to implement the methods, constructors and destructor, which are prototyped in `BigInteger.h`. So, for this project **any code written outside `BigInteger.cpp` will be useless.**

## Details of the Given Code

### Node

This class consists of an integer data field and a **Node pointer** to the next node which will be used to construct `LinkedList`. So the data structure is the same as you learned back in CMPE160. The copy constructor and destructor and some operator overloading is done by us, so you can use help while implementing yours.

## LinkedList

This class consists of two `Node` instances that represent heads and tails of the list and an integer to keep the length of the list. Constructors, destructor and push methods are implemented for you so you can make use of them too.

## BigInteger

This is the class you will be implementing and contains a pointer to a `LinkedList` object, which will be used to represent the number read. `BigInteger.h` is already written for you and you are responsible for implementing `BigInteger.cpp`. To implement the project successfully, you are going to

- overload `+` and `*` operators
- write copy constructors
- overload assignment operator
- implement destructor

You can of course implement additional helper functions but make sure that you do not modify header files as any other code written outside `BigInteger.cpp` is meaningless for grading. Remember, you can implement methods whose signature do not contain scope operator (`::`) without updating header.

## main.cpp

This is the class where you will be handling the input and output. Most of the I/O stuff has already been implemented. You are allowed to modify this class as we will replace yours with the original. Note that when you execute the main after removing `/*` and `*/` signs, your output must be the same as the corresponding result file.

# Input & Output

Your code must be compiled by the simple `cmake CMakeLists.txt` and `make` commands and your code must read the arguments from the command line. Your code will be tested with the command:

```
./project1 inputFile outputFile
```

## Input

Your program should read the input from the file which is specified while executing the program from the terminal. Input file will have the following format:

```
operand type
number1
number2
```

The first line in the file contains one char either '\*' or '+' which will determine the type of operation you will be applying to the numbers. The following two lines contain, not surprisingly, the two numbers you will be dealing with.

## Output

Similar to the input, your program will output to the file specified from the terminal. And the only thing you will output is the result of the operation with no new line at the end. Note that this may result in wrong evaluations since it will be done automatically.

Some example cases are as follows:

Table 1: Sample Test Cases

Content of Input File	Content of Output File
+ 123456789987654321 9999999999	123456799987654320
* 8974561 4131175414	37075485754643254
* 2 5	10
+ 987 5413	6400

## Warnings:

1. You are not allowed to use any library or object provided by C++ for multiplying and adding numbers. **You will get zero in this case.**
2. You must read the numbers as a string and transfer it to a Linked List. You are free to determine the number of digits will be held in one node. In other words, you can split the number digit by digit while pushing to the linked list or 5-digit by 5-digit.
3. You are expected to implement copy constructor and destructor, and overload assignment operator. Otherwise, when extensively tested, your code will very likely give segmentation fault and crash even though the addition and multiplication operations are just fine.
4. Make sure that an executable is created after executing `cmake CMakeLists.txt` and `make` commands. Otherwise, no executable will be created and your code will fail in grading.

## Submission Details

You are supposed to use the Git system provided to you for all projects. No other type of submission will be accepted. Also pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code.
- In our case, you are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
- Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.