CMPS 101-02 Spring 2018
Programing Assignment #2
Arbitrary Precision Arithmetic

April 26, 2018

Goal

The goal of this assignment is to learn how to use C to design an ADT for arbitrary precision integers. The assignment is almost the same as Programming Assignment 1, except that there are less requirements.

Program Specification

In this assignment you will implement an ADT in C to support the numeric type of *arbitrary precision integers*. Arbitrary precision means that there is **no fixed limit to the size of the numbers**. The sizes are limited only by the amount of memory available. Specifically, from the users point of view, the `apint` type is to be a base 10 number made up of an arbitrary number of digits.

Requirements

For the `apint` type you need to provide

- a default constructor
- a constructor which uses a string, made up of optional {+,-} followed by a string of characters from {0,1,2,3,4,5,6,7,8,9} as an input argument.
- a constructor for conversion of ints to apints.
- a method for printing.
- methods for addition, subtraction, and multiplication.

Testing Write test programs to verify that your ADT implementations are working correctly. You should have a test for every bullet point in the above list.

Direction and hints

- Use the name `apint` for your type and name your print method `print`. This is mandatory to facilitate testing and grading.
- Use good coding style. See `http://www.soe.ucsc.edu/~sbrandt/105/coding.html` for guidelines.

Submission instructions

- Create directory named `CMPS101S18PA<PROGRAMMING ASSIGNMENT NUMBER>`, e.g. `CMPS101S18PA2` for Programing Assignment 2.
- In the program assignment directory put in the following files
  - `README` - a short file which lists all the files in the directory and describes what they are.
  - `NoteToGrader` - a short note in which you describe your approach.
  - The `<SOURCEFILES>` - the Java files which you wrote in this assignment.
  - The `<TESTFILES>` - the test files which you used to ensure that your program works correctly.
- Go to your directory, and invoke the `script` command. For a tutorial on how to use this command see `http://www-users.cs.umn.edu/~gini/1901-07s/files/script.html`. In your case, you should invoke the following commands:
  - `script pa1submissionfile.txt`
  - `pwd` - this will show us which directory you are in.
  - `ls -l` - this will list the contents of the directory.
  - `cat README`
  - `cat NoteToGrader`
  - `cat <SOURCEFILES>` - this will print the contents of the source files to the screen. We want you to run this command because this is the only way in which we will see the code you wrote. Run this command on every source files you are using, but DO NOT RUN IT ON YOUR BINARY FILES!
  - `<commands to compile/link the program>` - run the commands which build your source code
  - `ls -l` - this will list the contents of the directory again, showing us that there are binaries which resulted from the previous step.

  - `cat <TESTFILES>` - this will print the contents of the test files to the screen. As before, we want you to run this command because this is the only way in which what kind of tests you ran to ensure that the source files are doing what they should be doing. Run this command on every test file you are using, but DO NOT RUN IT ON YOUR BINARY FILES!
  - `<commands to execute required tests>` - run the commands which test your binaries against specific files. Make sure that the results of these tests are printed to screen.
  - `exit` - this will exit the `script` command and produce a plain text called `pa1submissionfile.txt`.

- Take the plain text file which you created in the previous step and paste its contents to a `.pdf` file. If you've never created a `.pdf` file we have a guideline for this on canvas. This `.pdf` file is the only document you will submit on canvas.

- Do not submit source files or binaries!

- Do not run any editing commands between the `script` command and the `exit` command. It will produce a mess in the plain text file.