**CpSc 1070   
Program 3   
Due Thur., March 29, 11:59:59 pm**

For this assignment, you may work with a partner. If you choose to work with a partner, **one** of you should send me an email by Friday, March 16, letting me know the name of your partner, and only **one** of you will submit the assignment.

**Purpose:**The purpose of this assignment is to give you practice in working with linked lists.

**Problem Statement:**If we need to work with an integer that is too large to be represented by any known integer type, then we need to find another method of representing the big integer. An unbounded integer can be represented by storing the integer as a linked list of its individual digits. A more efficient representation will store a larger integer in each node.

For this assignment, you are to write a program for representing and manipulating unbounded whole numbers, in which a number is represented as a linked list of 3-digit integers. Each node will hold an integer less than or equal to 999.

The number represented by the list is the concatenation of the numbers in the nodes. For example, if there are four nodes with the four integers *32  5  879  0*, then this represents the number 32,005,879,000.

Note that the number in a node is always considered to be three digits long. If the number is not three digits long, then leading zeros are inserted to make the number three digits long.

You must implement the following functions in a file named ***bigint.c***

*char \*readNumber(ifstream &in)* - reads only one number.

1. reads the length of the string from the given input file, then invokes *getc()* to consume the newline character
2. dynamically allocates memory for a numeric string (array).
3. invokes *fgets()* to read the string of digits from the given input file and returns the numeric string.

*void buildList(list\_t \*bigint, char \*numstring)*

1. converts the numeric string to 3-digit integers
2. stores each 3-digit integer in a data\_t struct
3. adds the data\_t struct to the ***front of*** the list as a (void \*).

*int size(list\_t \*bigint)* - returns the number of nodes in the list.

*void printReverse(list\_t \*bigint)* - prints the list in reverse order; thus, the number is printed in its natural order. **Note:**the code for this function is in *reverse.c*

*void print(list\_t \*bigint )* - uses an iterator\_t \* to print the list of 3-digit integers, in the order they occur in the list.

*list\_t \*add(list\_t \*bigint1, list\_t \*bigint2)* - returns the list of nodes in the sum.

1. creates an empty list for the sum.
2. uses iterators to traverse the lists until one of the lists is exhasted, retrieving the data, adding the data values together, storing the result in a data\_t struct, then putting the result node at the ***end*** of the sum list. Each node will consist of the 3-digit sum of corresponding nodes of bigint1 and bigint2.
3. continues inserting the nodes of the other list, until that list is also exhausted.

*list\_t \*subtract(list\_t \*bigint1, list\_t \*bigint2)* - similar to *add()*, but returns *bigint1 - bigint2*.

If you are feeling excited that you can add and subtract() two *bigint* numbers, and you want more excitement, then you can rack up extra credit points by implementing a *multiply()* function as described below.

**Extra Credit:**list\_t \*multiply(list\_t \*bigint1, list\_t \*bigint2) - returns a pointer to a new list\_t, which is the product of the numbers represented by bigint1 and bigint2.

**Getting the files for this assignment**

1. Use *mkdir* to create a prog3 directory for program 3. You may name your directory whatever you want.
2. **cd** to the prog3 directory.
3. Download the [prog3.tar](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog3/Public/prog3.tar) file to your prog3 directory. You can save the tar file from your browser, or you can directly copy this file over the network using the scp utility:

***scp access.cs.clemson.edu:/home/rlowe/public\_html/cs1070/programs/spr18/prog1/Public/prog1.tar .***

 Use the command below to extract the files in prog3.tar

***tar -xvf prog3.tar***

After you untar the files, you may delete prog3.tar

Your prog3 directory should contain the files *bigint.dat,  bigint.h,* *list.c,   list.h, main.c*,  *makefile,*, and nbsp; *reverse.c*.

*bigint.h* contains the function prototypes and a typedef for a *data\_t* struct, which will hold the 3-digit integer data for the list nodes.

*main.c* is a test driver for the assignment.

*makefile* builds your project.

*bigint.dat* - contains two sample *bigint* integers. The first value is the number of digits in the numeric string, followed by the numeric string (on a separate line). The file should contain two lentth, numeric string pairs. Make sure you test your code with numeric strings with different lengths.

*reverse.c* - since we have not covered recursion, *reverse.c* contains the code for *printReverse*. You may copy this code into your *bigint.c* file or write your own.

[*output.txt*](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog3/Public/output.txt) contains sample output for the two bigint numbers in *bigint.dat*.

**Compiling and Testing Your Program**

1. Use the following command to build the executable, *p3*

*make*

1. Test the program using

./p3 < bigint.dat   
where bigint.dat is the name of the file that contains the length, numeric string pairs for two bigint integers.

**Submitting your files**

Submit ***bigint.c*** electronically using [Handin](https://handin.cs.clemson.edu/)

It is your responsibility to make sure you have submitted the correct file. Check your Handin folder after submitting the file. There is a 10-point penalty for using the wrong file name and a 10-point penalty for submitting the file to the wrong folder.

**Requirements**

1. For this program, you must work only with your partner.   
   Any evidence of cheating will result in a grade of **-100** for all students involved. If you have questions, you should check with me or a lab instructor.
2. You must name your file ***bigint.c***
3. You must use the *list* module provided.
4. You must use the *bigint.h* file provided. You will need to #include the *bigint.h* in your *bigint.c* file.
5. Your program must adhere to the [program standards](https://people.cs.clemson.edu/~rlowe/cs1070/programs/standards.html) and [program requirements](https://people.cs.clemson.edu/~rlowe/cs1070/programs/requirements.shtml). Violations will lead to deductions. In particular,
   * no more than one statement may be written on a single line; this includes declarations.
   * use reasonably descriptive names for variables and functions. no abbreviations allowed, unless universally known.
   * code lines should not extend beyond column 80.
   * indenting should be consistent with logical nesting; indent 3 - 4 spaces.
   * do not use a **break** statement to prematurely exit a control structure.
   * a brief description must precede each function.
   * no function should be longer than 30 lines of code, excluding whitespace.
   * your program should compile without any warnings with gcc -Wall
   * diagnostic / debug prints should be disabled/deleted in the final submission.

If you are using debug print statements, use conditional compilation.

1. The program is due by 11:59:59 pm. Programs must be submitted on time. The late pass policy applies.
2. Programs must compile. Programs that do not compile will receive a score of **zero**.

# of digits

Numeric string

Check for divisibility by 3

List\_addToFront(list, (void \*) data)