Assignment 9: Computations for Large Integers

EC602 Design by Software

Fall 2021

Contents

1 Introduction

1.1 Assignment Goals

The assignment goals are to help you learn about

- arithmetic operators
- number systems and bases
- sequences

1.2 Group Size

The maximum group size is 2 students.

1.3 Due Date

The assignment is due 2021-11-22 at 23:59:59

1.4 Assignment Value

This assignment is worth 5 points.

1.5 Late policy

Late assignments will be accepted until the beginning of the lecture immediately following the due date, or for 48 hours, whichever is less.

If the *natural grade* on the assignment is g, the number of hours late is h, and the number of hours between the assignment due time and the next class is H, the new grade will be

```
(h > H) ? 0 : g * (1- h/(2*H))
```

If the same assignment is submitted ontime *and* late, the grade for that component will be the maximum of the ontime submission grade and the scaled late submission grade.

1.6 Submission Link

You can submit here: bigint hw9 submit link

2 Background: Polynomials

The standard way to write a polynomial in x containing only positive powers of x is

$$p(x) = \sum_{i=0}^{N} c_i x^i = c_N x^N + c_{N-1} x^{N-1} + \cdots + c_1 x^1 + c_0$$

If negative powers of x are involved, then you can write

$$p(x) = \sum_{i=-M}^{N} c_i x^i = c_N x^N + c_{N-1} x^{N-1} + \cdots + c_1 x^1 + c_0 + c_{-1} x^{-1} + \cdots + c_{-M} x^{-M}$$

The coefficents can be integer, real, or complex. The two most common cases we will see are integer and real. The polynomial variable x can also be integer, real, or complex (there are also more general types of polynomials on other algebraic systems).

2.1 Representing Numbers as Polynomials.

When we write an integer, say 451, or a real number, say 31.4159, we are actually using a shorthand for a polynomial in 10, as follows:

$$451 = 4x^2 + 5x^1 + 1x^0$$

where $x = 10$

and

$$31.4159 = 3x + 1 + 4x^{-1} + 1x^{-2} + 5x^{-3} + 9x^{-4}$$

where $x = 10$ again.

For convenience, to establish uniqueness, and by convention, when we write a number this way we also restrict the coefficients to be *integers* in the range

$$0 \leq c < x$$

Numbers can also be written in base 2 (binary), which is important because computer memory is binary.

For example, 16.125 can be represented as

$$16.125 = 1x^4 + 1x^{-3}$$

where $x = 2$

and so $16.125 = 1000.001_2$

3 Background: Sequence Types

3.1 Sequences in C++

In C++, the most commonly used sequence types are:

- C-style array
- vector
- string
- array

Here is an example of defining each of these.

```
#include <string>
#include <vector> // vectors are indexable and variable size.
#include <array>  // arrays are indexable and fixed size.
```

```
string s;
array<int,3> position;
vector<int> v;
int a[6];   // this is a built-in "C" array.
```

In C++, every element of a sequence must be the same type.

4 The assignment: `bigint`

Write a C++ library file `bigint.cpp` to perform arbitrary precision integer multiplication in C++.

Here is the function signature:

```
typedef string BigInt;
BigInt multiply_int(const BigInt &a,const BigInt &b);
```

This code is contained in the header file, which must be in your directory when building your code: bigint header file

The values will be stored as the base-10 representation of the number, i.e. as strings of ASCII characters `'0'` to `'9'`.

So, for example,

```
string b = "111111";
string c = "1111111";
string d = multiply(b,c);
cout << d << endl;
```

The output should be 123456654321

You will need to include `<string>` in your program, and you are allowed to include `<vector>` but you must not include any other library.

Since your program will be compiled with my version of `main()` you must not have a `main()` function in this file.

Your program will be tested using

```
g++ bigint.cpp bigint_example_main.cpp
```

You can write your own main program to test your functions, or you can start with the example I wrote.

Here is an example: bigint main example

The filename of the program submitted must be `bigint.cpp`

4.1 Checker

There is a checker available here:

hw9_bigint_check.py

The latest version number is 2.2

4.2 Brackets

Brackets are not allowed.