

Lab 03 -- Testing Concept 1

Include guard tutorial

<https://www.youtube.com/watch?v=RU5JUHAiR18>

https://docs.google.com/presentation/d/1tNygdwW1DoOjpVILGC5PU6OFnX8nCw6lgVlkcowzKx8/edit#slide=id.g6296a3e082_0_23

Addition

99999 * 99999

Test Cases for Bigint Addition

Folder setup:

You already have bigint folder.

Inside bigint folder , look at

after logging to unix account execute the following commands

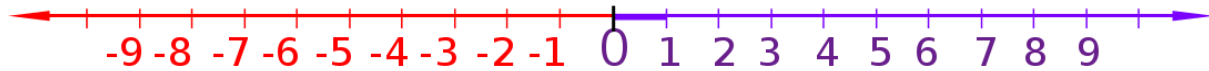
test_add.cpp → contains test cases template for testing bigint add operation
and

test_multiply.cpp → contains test cases template for testing bigint multiply operation

They are not comprehensive. Need to write more test cases.

Boundary Value Analysis

Numbers can be negative, zero or positive



(image source : obviously wikipedia)

Negative	0	Positive
Out of scope		

NEgative Bigint's are not considered for project 1. Hence, forget them for now. Let's focus on only 0 and positive numbers. And let's assume the **char array** to store bigint digits is of length 5 (Actually it's of length 200. Because your **const int CAPACITY = 200;**. I don't want to write too long bigints. But you are going to write some really bigints!!!!)

Q. What bigints can we store in a 5 length char array?

Ans.

0	Positive (assume 5 length char array for bigint)
0	1 ----- 99999
0	1---9, 10-----99, 100----999, 1000 ----9999, 10000---- 99999

So we have **5** boundaries here

We divided them in 5 equivalence classes

Add 0 with them → 6 classes

Class	Range
1	0
2	1--9
3	10---99
4	100---999
5	1000---9999
6	10000----99999

Add operation

LHS + RHS = RESULT

LHS	RHS	RESULT	Total test cases
0-999999	0-999999	0-999999	100000*100000*100000
6	6	6	$6*6*6 = 216$
6 (m dig)	6 (n dig)	2 a) max(m,n) b) Max (m,n) + 1	$6*6*2 = 72$
1+1 (m dig)	1+1 (n dig)	2 +1 c) max(m,n) d) Max (m,n) + 1	$2*2*3 = 12$

999, 9999

Q. How many test cases will be comprehensive

Ans.

6 x 6 ----> 36 test cases

For i =1 to 6:

```

Take a number x from class i
Use x as LHS of add operation
For j = i to 6
    Take a number y from class i
    Use y as RHS of add operation
    Write test case using template given by professor in test_add.cpp
    {
        //-----
        // Setup fixture
        bigint left(x);
        bigint right(y);
        bigint result;

        // Test
        result = left + right; //calculate z = x+y in paper . You will need it

        // Verify
        assert(left == x);
        assert(right == y);
        assert(result == z);
    }

```

Wait Wait Wait !!!!!

**Number of digits in Result = Max(Number of digits in LHS, Number of digits in RHS) or
Max(Number of digits in LHS, Number of digits in RHS) + 1**

Example :

5+4=9 -- result contains only 1 digit
 9+9 = 18 -- result contains 2 digit . **This has carry over!!!**
 91 + 3 = 94
 91 + 9 = 100

So for each of the 36 classes, result can be of 2 types

So in total
 $6 \times 6 \times 2 = 72$

**So, 72 test cases seems pretty good .
 But we don't want to spend so much time. Let's observe more**

Example 1 : 22 + 39 = 61

Example 2 : $222 + 339 = 561$

Example 3: $2 + 1000 = 1002$

Q. What's the similarity?

Ans.

If your code is ok for 2 digits it should be also good for 3/4/5 digits.

So, if test works only for bigint's with 3 digits and 4 digits. Combine them and if you get correct results. We are confident that, our code will work for bigints with 195 and 125 digits. Right !!!!

Doing tests for 192 and 125 digits also gives confidence for testing with 94 and 25 digits bigint.

Writing test cases: example

Example: Assume we want to write test cases for Class 4 bigints. And the operation is "ADD"

LHS (m =3)	RHS	Result
1) Any 3 digit bigint 2) 2 digit 3) Special case 0	Possible cases: 1) Any 3 digit bigint 2) Any bigint that has more than 3 digits 3) Any bigint that has less than 3 digits 4) Special Case of 0	2 possible cases 1) Result has 3 digits 2) Result 4 digits

Now let's convert that into code:

Assume we want to write test for the following selections

a) LHS is any 3 digit bigint, RHS is a 3 digit bigint and result is also a 3 digit bigint

- i) Example : $666+333 = 999$
- ii) $666+6666 = 4 \text{ dig}$
- iii) $666+555 =$
- iv) $666+0$

b)

- i) $666*666 = 443556 \text{ (2 *3)}$
- ii) $666 \times 99 = 65934 \text{ (2*3)}$
- iii) $3 \text{ dig} * 2 \text{ dig} \rightarrow (2 *3 -1)$

- c) LHS is any 3 digit bigint, RHS is a bigint that has less than 3 digits and result is a 4 digit bigint
- 1) Example : $999+22 = 1021$

Some other important points

1. I have discussed for only 5 digit bigint. But your **const int CAPACITY = 200;**
2. Make variation on LHS and RHS digits and repeat them.
 - a. Example:
 - i. 3 digits on LHS, 4 digits on RHS
 - ii. 4 digits on LHS, 3 digits on RHS
3. Make sure you are testing for carry!!
4. For really large bigints use char array constructor
 - a. Example : bigint ("99999999999999.....9999"). Assume you have hundred and twenty five 9 in a bigint You can't use **int constructor**. You have to use **char array constructor**
- 5.

Make FILE modification is necessary

Test Cases for Multiplication

Same as addition. Only change the operation from **Add(+)** to **Multiply(*)**

Pro Tips !!!!

1. Always study existing codes.
2. Look at test_multiply.cpp
3. Almost everything discussed upto now is provided as an example in the file.