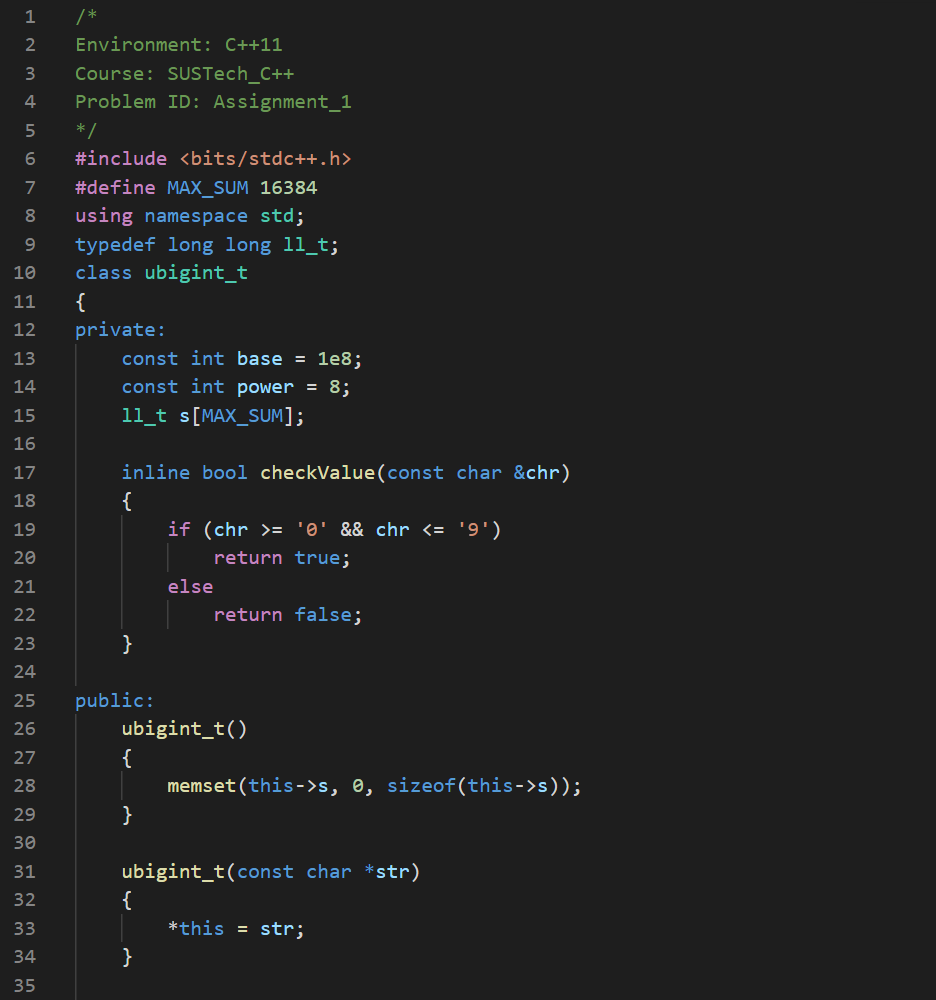
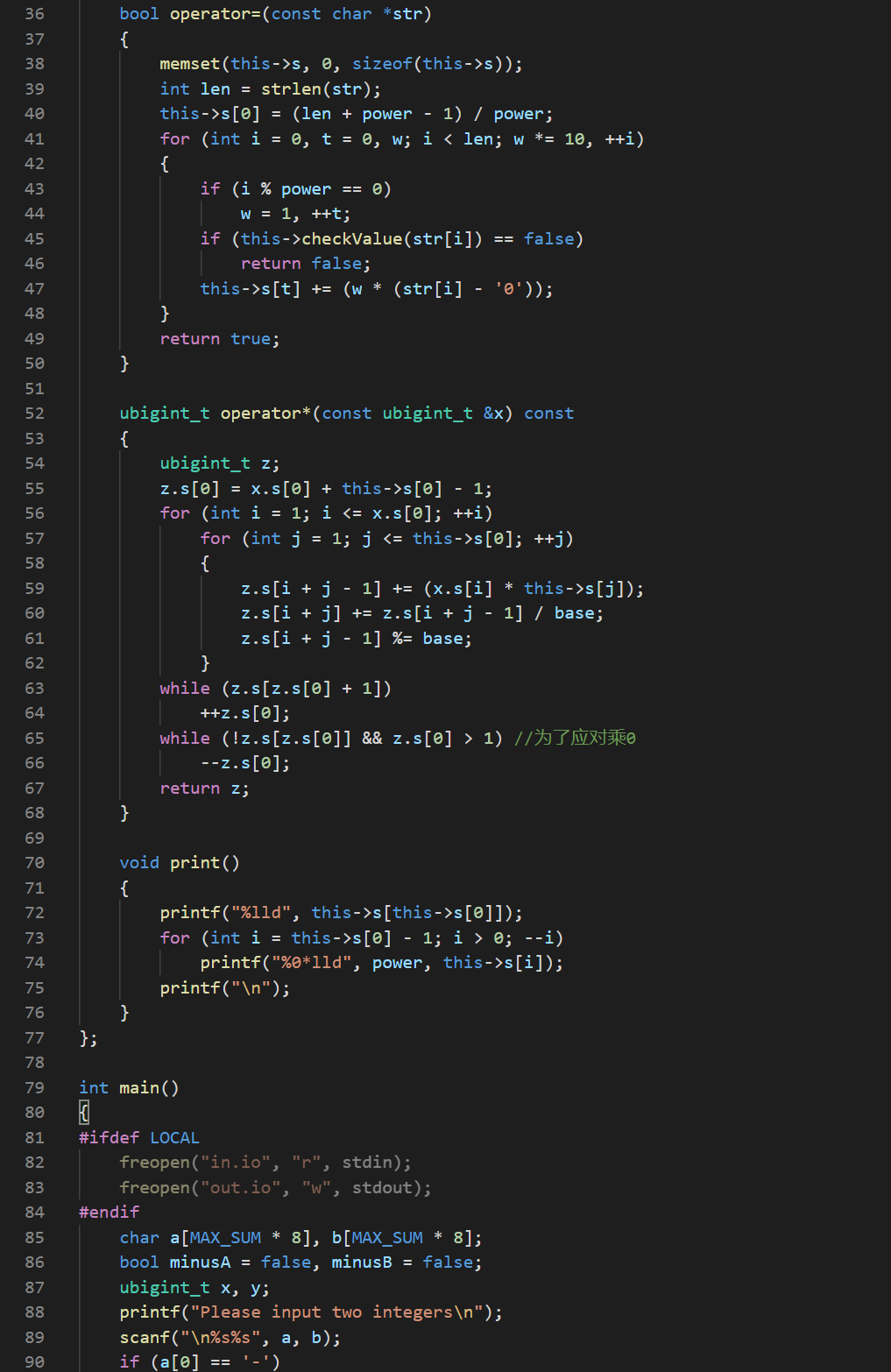
**CS205 C/ C++ Program Design**

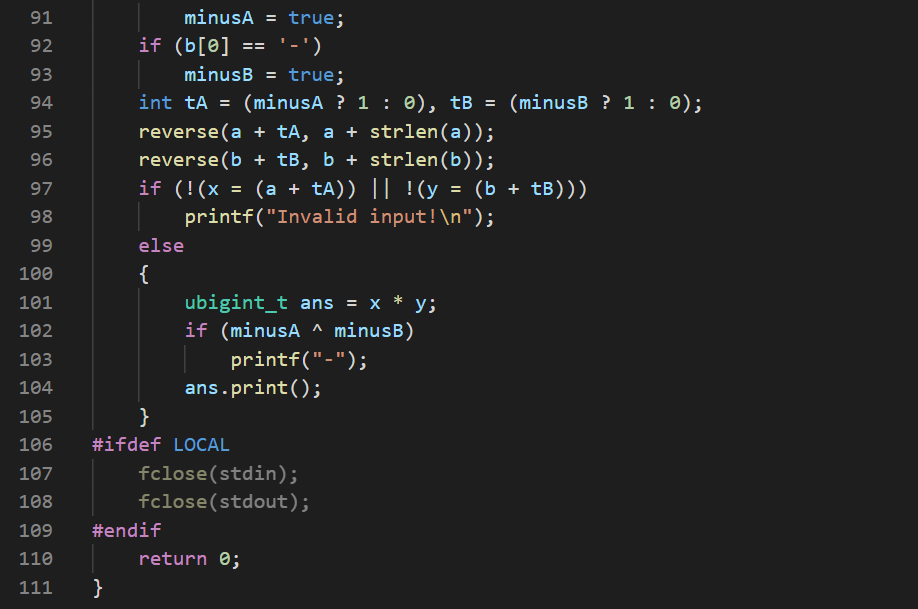
**Assignment 1**

**Name**: 黄彦淞, **SID**: 11912824

## Part 1. Source Code







注：代码中的#ifdef与#endif命令仅在本地执行，完整代码文件请见另一附件。

## Part 2. Result & Verification

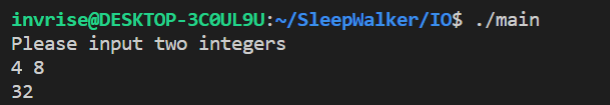
In this part, you should present the result of your program by listing the output of test cases and optionally add a screen-shot of the result.

**Test case #1:**

Input: 4 8

Output: 32

Screen-short for case #1:



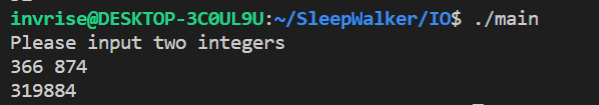
其中./main为编译后的可执行文件。

**Test case #2:**

Input: 366 874

Output: 319884

Screen-short for case #2:



## Part 3. Difficulties & Solutions, or others

**Point #1: 超长整数乘法**

由于C++中64位整数型可存储的数据范围仅有-264~264-1，对于超过这一范围的整数以及可能的结果，使用基本数据类型将不能正确计算。

解决方案：使用压位高精度整数结构，将原整数进行拆分，得到多个处于int可表示的范围内的数字，利用类似竖式乘法的过程进行计算。

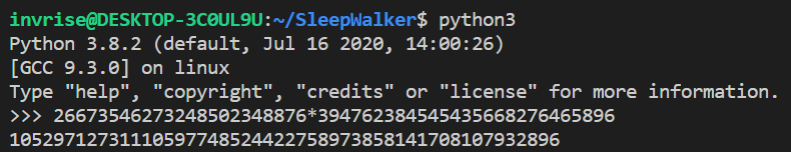
代码对应：ubigint\_t类，理论最高支持两个8192\*8=65536位整数的乘法运算，通过调节代码中MAX\_SUM宏的值，可以继续提高这一上限。

**Test case #3:**

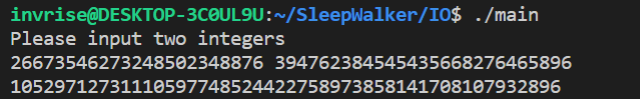
Input: 26673546273248502348876 394762384545435668276465896

Output: 10529712731110597748524422758973858141708107932896

Python验算：



Screen-short for case #3:



**Point #2: 非法输入**

在输入的整数中，可能出现各种导致运算无法正常进行的异常因子，为了不使得程序在运行中崩溃，需要在输入时检测出这些错误。

解决方案：使用字符串对数字进行读入，并在扫描形成可处理的ubigint\_t对象时检测每一位的字符是否合法。

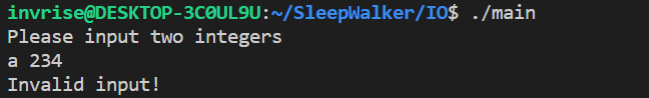
代码对应：程序入口函数与ubigint\_t类中的赋值号重载函数。

Test case #4（一般错误）:

Input: a 234

Output: Invalid input!

Screen-short for case #4:

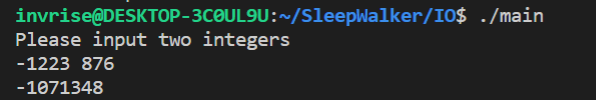


Test case #5（负号误判的避免）:

Input: -1223 876

Output: -1071348

Screen-short for case #5:

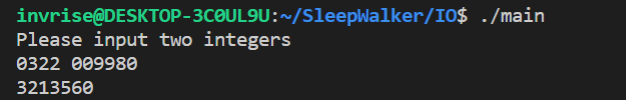


Test case #6（前导零的去除（不视为八进制））:

Input: 0322 009980

Output: 3213560

Screen-short for case #6:



Test case #7（中段错误）:

Input: 24373&%4 213

Output: Invalid input!

Screen-short for case #7:

