Program #1 **CS 501X**
J. Dichter **University of Bridgeport**

General Instructions:
Write a program which will add digit strings. The input strings will be read *Byte* objects. The program will evaluate the sum of each byte pair. The program will write each input pair as well as the results to an output file according to the format specified below. The results will include meaningful messages about erroneous string input or addition overflow when applicable. The program will continue to read and evaluate strings until the end of the input file is reached. You will need to define *Byte* as a class. The obvious overloaded operators will be addition and output, i.e. `+()` and `<<()`. The input format is given at the bottom of the page.

Specific Instructions:
This assignment should be written as a two-module C++ program. You will develop a class (module) called *Byte*, with header (.h) and implementation (.cpp) files. The main program should be in its own module, and be about 20 lines long. You are to handle additions of both positive and negative binary strings with the following assumptions: the first string is stored using two's complement and the second using 8-bit bias notation. The result should be a bit string in two's complement (including a decimal interpretation of the result). To keep track of overflow keep the following scenarios in mind:

1) Adding a positive and a negative cannot create overflow.

2) Adding two positives can create overflow, indicated when the highest order bit of the result is 1 after the operation.

3) Adding two negatives can create underflow, indicated when the highest order bit of the result is 0 after the operation.

Sample:	00010110	2's complement		22
	+ 10000011	biased notation	+	3
	----------			---
	00011001	2's complement		25
	10000001	2's complement		-127
	+ 00000000	biased notation	+	-128
	----------			---
	underflow			-255

Input Data:

11111111 00000000
00000011 10000111
00001111 10000111
10000000 11111111
11110000 10001000
10000001 00000001
01111111 00000000
01110101 11010001
00000000 10000000
00001111 11110000

It is recommended that you define the following Byte class similar to the following declared class. Note that some methods or operators should be members of the class, others should be declared *friend* to *Byte*. A suggested Byte declaration is given below.

```
#include <iostream>
using namespace std;

#ifndef __BYTE
#define __BYTE

class Byte {
public:

	enum { UNDER, OVER, OK, BYTESIZE = 8 };
	Byte() { error = false; }
	Byte(char *);
	int read(istream & = cin);
	Byte add(Byte);

	// switch high-order bit only
	Byte biasTo2sComplement();

	// apply 2s complement conversion for magnitude evaluation
	Byte to2sComplement();

	friend ostream & operator << (ostream &, const Byte &);
	int toInteger();
	const char * getErrorMessage() const;
	bool hasError() const;

private:
	char byte[BYTESIZE];
	bool error;
	const char * errorMessage;
	int magnitude();
};

#endif
```