Assignment #3
Due November 6th
50 points

You can write your programs in the environment of your preference (e.g. Visual Studio, or Unix). Turn in your source code files via Blackboard by submitting a single zip file `LASTNAME_assignment_3.zip`. This zip file should contain a folder containing the source code for each problem. The zip file should also contain a `LASTNAME_README.txt` file that describes how to run each program as well as what doesn't work.

All your programs should use separate files for header and implementation code.

1) Operator Overloading – 10 pts

The following is the header and implementation for a `Pair` class. The class simply stores a pair of numbers. There is an overloaded member operator **+** that adds two pairs together or adds a constant value to each number in the pair. Here is the class definition that goes in the file `Pair.h`:

```
#pragma once
class Pair
{
private:
	int num1, num2;
public:
	Pair();
	Pair(int num1, int num2);
	int get1();
	int get2();
	Pair operator+(const Pair& other);
	Pair operator+(int otherNum);
};
```

Here is the implementation that goes in the file `Pair.cpp`:

```
#include "Pair.h"

Pair::Pair() : num1(0), num2(0)
{
}

Pair::Pair(int num1, int num2) : num1(num1), num2(num2)
{
}
```

```
	int Pair::get1()
	{
		return num1;
	}

	int Pair::get2()
	{
		return num2;
	}

	// Return a new pair that adds the corresponding numbers
	Pair Pair::operator+(const Pair& other)
	{
		Pair newPair(this->num1, this->num2);
		newPair.num1 += other.num1;
		newPair.num2 += other.num2;
		return newPair;
	}

	// Return a new pair that adds otherNum to num1 and num2
	Pair Pair::operator+(int otherNum)
	{
		Pair newPair(this->num1, this->num2);
		newPair.num1 += otherNum;
		newPair.num2 += otherNum;
		return newPair;
	}
```

Here is the contents of the file `main.cpp`:

```
#include <iostream>
#include "Pair.h"

using namespace std;

int main()
{
	Pair p1(5, 10);
	Pair p2(1, 2);

	// Outputs 5 and 10
	cout << p1.get1() << " " << p1.get2() << endl;
	// Outputs 1 and 2
	cout << p2.get1() << " " << p2.get2() << endl;

	Pair p3 = p2 + p1;
	// Outputs 6 and 12
	cout << p3.get1() << " " << p3.get2() << endl;

	p3 = p3 + 2;
	// Outputs 8 and 14
	cout << p3.get1() << " " << p3.get2() << endl;
}
```

As written, the program runs as intended! But if we change the last overloaded `+` so that the `2` is the first operand then the program will not compile. In other words, we have problems if we write:

```
 p3 = 2 + p3;
```

To do and turn in:

- Put the original code into separate files, compile, and run it. You can use the programming environment of your choice. If you're using Linux and compiling from the command line, don't forget you need to compile `main.cpp` and `Pair.cpp` together unless you create a `Makefile` or do separate compilation.

- Change the line `p3 = p3 + 2;` to `p3 = 2 + p3;` and explain why it doesn't work in a comment.

- Overload the `+` operator as a friend function/operator so that all of these work: `2 + p3` or `p3 + 2` or `p1 + p2` or `p2 + p1` (the last two already work in the original code, but you should rewrite them as friends also to be consistent and for a little extra practice)

Turn in your source files, which should include your comment about why `p3 = 2 + p3` didn’t work.

2) <u>POSTNET – 15pts</u>

Prior to 2009 the bar code on an envelope used by the U.S. Postal Service represented a five (or more) digit zip code using a format called POSTNET. The bar code consists of long and short bars as shown below :

For this program, we will represent the bar code as a string of digits. The digit 1 represents a long bar, and the digit 0 represents a short bar. Therefore, the bar code above would be represented in our program as:

110100101000101011000010011

The first and last digits of the bar code are always 1. Removing these leave 25 digits. If these 25 digits are split into groups of five digits each then we have:

10100 10100 01010 11000 01001

Next, consider each group of five digits. There always will be exactly two 1's in each group of digits. Each digit stands for a number. From left to right, the digits encode the values 7, 4, 2, 1, and 0. Multiply the corresponding value with the digit and compute the sum to get the final encoded digit for the zip code. The table below shows the encoding for 10100.

Bar Code Digits	1 0 1 0 0
Value	7 4 2 1 0
Product of Digit * Value	7 0 2 0 0
Zip Code Digit	7 + 0 + 2 + 0 + 0 = 9

Repeat this process for each group of five digits and concatenate to get the complete zip code. There is one special value. If the sum of a group of five digits is 11, then this represents the digit 0 (this is necessary because with two digits per group it is not possible to represent zero).

The zip code for the sample bar code decodes to 99504. While the POSTNET scheme may seem unnecessarily complex, its design allows machines to detect if errors have been made in scanning the zip code. Write a zip code class that encodes and decodes five-digit bar codes as previously used by the U.S. Postal Service on envelopes. The class should have two constructors. The first constructor should input the zip code as an integer, and the second constructor should input the zip code as a bar code string consisting of 0's and 1's as described above. Although you have two ways to input the zip code, internally, the class should only store the zip code using one format. (You may choose to store it as a bar code string or as a zip code number.) The class also should have at least two public member functions: one to return the zip code as an integer and the other to return the zip code in bar code format as a string. All helper functions should be declared private. Embed your class definition in a suitable test program.

3) Array of Classes – 10pts

The following movies have recently come out:

1. Black Panther, PG-13
2. Avengers: Infinity War, PG-13
3. A Wrinkle In Time, PG
4. Ready Player One, PG-13
5. Red Sparrow, R
6. The Incredibles 2, G

To do:

1. Create a `Movie` class that stores the movie name and the MPAA rating (i.e. R, PG-13, etc.). Write appropriate constructors, setters, and getters.
2. In your `main` function create an array of type `Movie` that stores the movie information.
3. Write a `sort` function (it could be bubble sort, selection sort, etc., any sort of your choice) that sorts the movies alphabetically by name.
4. In your `main` function call the sort function and output the names and ratings in name-sorted order.

4) Benford's Law Re-Visited – 5 pts

Re-do the Benford Law problem from Homework #1, except this time open the file using an input stream from within the program rather than using file I/O redirection.

5) <u>Maze Class – 10pts</u>

Start with the attached solutions to the Maze Generator/Solver from Homework #2 Problem 2. Re-write the program using classes. The design is up to you, but at a minimum you should have a `Maze` class with appropriate constructors and methods. You can add additional classes you may deem necessary.