HW #5 (Overloading: BigInt & MyVector)

Part 1:

- Here we introduce a BigInt class which can represent integers much larger than the maximum bound for *integer* type in computers.
- You will deal with only **positive** BigInt in this problem. The BigInt class uses a dynamic array of **char** to represent the “big” integers. Each of the elements in the array can only be one of ‘0’, ‘1’, ‘2’, …, ‘9’, i.e., each array element stores just *one* digit of the “big” integer.
- The BigInt.h file defines the BigInt class as follows:

HW #5 (2)

```
class BigInt {
public:
    // constructor
    BigInt () {
            num = NULL;
            size = 0;
    };
```

HW #5 (3)

```
// convert an array of integral digits by tmp to BigInt
// tmp: pointer to the array
// length: the number of digits in the array
BigInt (const int* tmp, int length);

BigInt (const BigInt &); // copy constructor

// Assignment
const BigInt & operator=(const BigInt &);
```

HW #5 (4)

```
// destructor
~BigInt() {
    if (num != NULL) delete [ ] num;
};

char & operator[ ] (int index);

int length() const { return size; };

char* getNum() { return num; };
```

HW #5 (5)

```
private:
    char* num; // the big integer in char
    int size;      // number of digits in the big integer
};
```

HW #5 (6)

1A:

- Implement the overloading of the operator [ ], so that it returns the digits in *char* of a particular index. For example, if A is a BigInt object, we can get the 5th digit of A from the left in *char* by calling A[4] (digits are indexed 0, 1, 2, … from the left).

```
char & BigInt::operator [ ] (int index) {
    assert (index >=0 && index < size);
    // your code below
```

HW #5 (7)

1B:

- Implement the constructor which takes in an integer array of digits and converts it to a BigInt object. You may assume that the digits in the array are all integers between 0 and 9.
- **Hint:** in order to convert a digits to a *char* type, you can add (‘1’ – 1) to it. For example, if you want to convert the digit 3 to a *char* in the form of ‘3’, you can use 3 + (‘1’ – 1).

```
// convert an integer array of digits (0-9) to BigInt
// tmp: pointer to the array
// length: the number of digits in the integer array
BigInt (const* tmp, int length) {
```

HW #5 (8)

1C:

- In addition to the above, you need to implement prefix increment of BigInt, i.e., you want to support ++*bi* for a BigInt *bi*. Write the prototype of the member function in the BigInt class, and its implementation outside the class.

1D:

- You also want to implement postfix increment which supports *bi*++ for a BigInt *bi*. Write the prototype of the member function in the BigInt class, and its implementation outside the class.

HW #5 (9)

Part 2:

- Design a class MyVector with two **private** data members: *int length* and a *pointer to double* for memory dynamic allocation.
- Include the following methods in the **public** section:
- two constructors: one default without parameters and one constructor initializer with one parameter of *int* type which will be used for specifying the requested vector length, the other parameter of *pointer* type for specifying that vector.
- a copy constructor;

HW #5 (10)

- destructor;
- assignment operator;
- overload the output operator <<;
- operator* for calculating inner product of two vectors;
- operator* for vector multiplication with a constant;
- operator+ for adding two vectors.

- Write a test program which implements all of the above methods.