COSE213 Data Structures (Spring 2021)
Instructor: Prof. Won-Ki Jeong
Due date: March 28, 2021, 11:59 pm.

Assignment 1: C++ and Array (100 pts) (ver1.2, 2021.3.26)

In this assignment, you will implement the polynomial abstract data type (ADT) using C++ and array. The skeleton files – polynomial.h and main.cpp, are provided. You are required to complete the implementation of the class `Polynomial` in `polynomial.cpp`. This class definition is based on the version 3 discussed in the class.

a) Skeleton files

polynomial.h

In this file, the polynomial abstract data type (class) is defined as follows.

- `Polynomial(const Polynomial& source)` : copy constructor
- `~Polynomial()` : destructor
- `Polynomial& operator = (const Polynomial& source)` : Assignment operator
- `Polynomial operator+(const Polynomial& source)` : Sum of polynomials
- `Polynomial operator-(const Polynomial& source)` : Subtraction of polynomials
- `Polynomial operator *(const Polynomial& source)` : Multiplication of polynomials
- `Polynomial Derivative()` : compute the derivative of polynomial
- `float Eval(float x)` : evaluate polynomial at x
- `void CreateTerm(const float coef, const int exp)` : create a new polynomial term of coef*x^(exp). exp must be non-negative value. If the same exponent term already exists, replace its coefficient with the new one.

There are several functions pre-defined in the header files (you are not allowed to change the implementation of these functions), such as

- `Polynomial()` : default constructor
- `Print()` : print the current polynomial.
- `Capacity()` : maximum number of terms this polynomial can store
- `Terms()` : returns the number of non-zero terms currently stored in this polynomial

- `GetTerm(int x)` : get the access of a term (x is the index of that term).

polynomial.cpp

This is the actual implementation of the functions defined in the header file. You need to fill in // ToDo part.

Note

1. Make sure you implement the copy constructor and assignment operator correctly so that assigning one polynomial to the other using = operator or creating a new polynomial from an existing polynomial object work correctly.

2. Make sure that `CreateTerm()` resizes the array if more terms are created than the current object can handle. Stress tests will be conducted.

3. When a new term is added using `CreateTerm()`, if there is no term existing in the current polynomial of a given exponent, you need to create a new term. If the same exponent term exists, then you need to replace its coefficient with the new one.

4. `CreateTerm()` does not need to be called the descending order of exponent term, but the termArray must be arranged in descending order. For example, if you call `CreateTerm(3,1)` to create a polynomial term `3x`, and then you create a higher order term by calling `CreateTerm(2,2)`, then the resulting polynomial should be `2x^2+3x` (not `3x+2x^2`).

5. If a term has a zero (or close to zero) coefficient, then you need to remove that term from the polynomial. For example, if `p1=2x^2+1` and `p2=2x^2+x+1`, then `p1-p2` will make the coefficient of `x^2` term zero, so the result should be `-x-1`.

6. We provided `print()` function that prints out the given polynomial as the following form.

   $$c_n x^n + c_{n-1} x^{(n-1)} + \ldots + c_1 x^1 + c_0$$

   Do not change print() function because this will be used to check your code. You must manage your polynomial terms in a descending order because print function does not sort the order of terms.

7. You must submit `polynomial.cpp` only.

8. You are not allowed to use STL library.

main.cpp

This is the driver program to test your Polynomial class. This file contains testing code to evaluate the correctness of your implementation. If your implementation is correct, the result should be as follows:

```
f = -4x^3+2.3x^2-3
g = 3x^4-4x^3-8
g (creating a new term) = 3x^4-4x^3+5x^2-8
h (created from f) = -4x^3+2.3x^2-3
h (assigned from g) = 3x^4-4x^3+5x^2-8
f + g = 3x^4-8x^3+7.3x^2-11
f - g = -3x^4-2.7x^2+5
f(3.5) is -146.325
Derivative of f = -12x^2+4.6x^1
```

This is just a simple test code. When we grade your code, we will conduct more rigorous testing to check whether every function is working correctly and robustly.

b) Compile and submit

You must submit the code (`polynomial.cpp` only) online via blackboard. You can compile the code as follows:

```
> make
```

The output executable name is `assign_1`. You can run your code by simply type in this name in the terminal.

```
> assign_1
```

Good luck, and ask TAs and professor if you have any question.