

## Assignment 1: C++ and Array (100 pts)

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 template class `Polynomial` in polynomial.h.

### a) Skeleton files

#### polynomial.h

In this file, the polynomial abstract data type is defined. You will find function names. Actual implementation of those functions should be in the same file because you should use template. You need to implement the following functions:

- `Polynomial()` : default constructor
- `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
- `T Eval(T x)` : evaluate polynomial at x
- `void Print()` : print current polynomial
- `void CreateTerm(const float coef, const int exp)` : create a new polynomial term of  $\text{coef} \cdot x^{\text{exp}}$ . exp must be non-negative value. If the same exponent term already exists, replace its coefficient with the new one.

In addition to the functions listed above, you may add other functions if needed.

## 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.
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 (although `print()` function must print the polynomial in descending order). For example, `CreateTerm(3,1)` will create a polynomial term  $3x$ , and then you can create a higher order term by calling `CreateTerm(2,2)`, which makes the polynomial  $2x^2+3x$ .
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  $-x-1$ . You should use the machine precision to detect zero/nonzero for floating point coefficient (for example, if  $|x| < 10^{-15}$  then  $x$  is zero for double precision).
6. `print()` function will print the polynomial in descending order in the following format (also see the result of `main.cpp` below):

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

Note that the sign of the first term is printed only when it is negative (+ only appears in the middle of the polynomial).

7. Your polynomial template class must be able to handle at least double and integer types, for example, `Polynomial<double>` and `Polynomial<int>`.

8. You must submit `polynomial.h` only.

9. 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 = -4.8x^3+2.9x^2-3
g = 4.3x^4+2.2x^3-8.1
g (creating a new term) = 4.3x^4+2.2x^3+3.5x^2-8.1
h (created from f) = -4.8x^3+2.9x^2-3
h (assigned from g) = 4.3x^4+2.2x^3+3.5x^2-8.1
f + g = 4.3x^4-2.6x^3+6.4x^2-11.1
f - g = -4.3x^4-7x^3-0.6x^2+5.1
f(3.5) is -173.275
i (integer coefficients) = 2x^2+4x^1+3
j (integer coefficients) = 2x^1
i * j = 4x^3+8x^2+6x^1
i(6) = 99
```

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 and the report online. You must log in `uni06~10.unist.ac.kr` for coding and submitting the assignment. You can compile the code as follows:

```
> g++ main.cpp polynomial.h -o assign_1
```

The output executable name is `assign_1`.

You must write a report describing your work in this assignment. If you are ready to submit the code and the report, then make a zip file containing the files (`polynomial.h` and the report document file (pdf or doc)) and use the `dssubmit` script as follows:

```
> dssubmit assign1 assign1.zip
```

In order to run the `dssubmit` script, you must include the following directory into your `PATH`. You can modify `PATH` in your `.bashrc` script as follows:

```
export PATH=$MPICH_HOME/bin:$PATH:/home/cse221/bin
```

and then run `source ~/.bashrc` to apply the change.

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