

MCS 360 Project Three : adding polynomials
due Monday 25 October at noon

The goal of this project is to add polynomials over any number type using the STL list class in the implementation of a class Poly. A test program `use_poly.cpp` is shown below.

```
#include "poly.h"
#include <iostream>
using namespace std;

int main()
{
    Poly<int> a(5,3);
    cout << "a = " << a.to_string() << endl;
    Poly<int> b(-2,2);
    cout << "b = " << b.to_string() << endl;
    Poly<int> c = a + b;
    cout << "c = a + b = " << c.to_string() << endl;
    Poly<int> d(1,0);
    cout << "d = " << d.to_string() << endl;
    Poly<int> e = c + d;
    cout << "e = c + d = " << e.to_string() << endl;
    Poly<int> f(2,2);
    cout << "f = " << f.to_string() << endl;
    Poly<int> g = e + f;
    cout << "g = e + f = " << g.to_string() << endl;
    Poly<int> h = e + g;
    cout << "h = e + g = " << h.to_string() << endl;
    return 0;
}
```

Running `use_poly` at the command prompt `$` produces the following output:

```
$ use_poly
a = 5*x^3
b = -2*x^2
c = a + b = 5*x^3-2*x^2
d = 1*x^0
e = c + d = 5*x^3-2*x^2+1*x^0
f = 2*x^2
g = e + f = 5*x^3+1*x^0
h = e + g = 10*x^3-2*x^2+2*x^0
$
```

The class `Poly` exports three methods:

1. constructing a polynomial from a coefficient and degree,
2. the operator `+` to add two polynomials, and
3. a method `to_string()` for a string representation.

In the internal representation, the monomials need not be sorted along degrees, but if the addition results in a zero coefficient, then the monomial must be removed from the list of monomials.

Some important points:

1. You may develop your solution with any C++ compiler, but your solution will be tested with the `g++` compiler. It is recommend that before submission, you check your program on a computer in lab SEL 2263 on campus.
2. Your solution *must* use the STL list class. Correct programs that do not use the STL list class will receive only half of the points.
3. The class `Poly` is a template class so the code should work for any number type.
4. Every method in your program must have appropriate documentation. In particular, the documentation describes the purpose of every parameter.
5. Submit two files: the header file `poly.h` and the file `poly.tc`.
6. Handing in an incomplete but working program is better than handing in a program that crashes or does not run at all.
7. The first line of your C++ program must be

```
// MCS 360 Project Three by <Author>
```

where you replace the `<Author>` by your name.

8. Collaborations are not allowed. You must solve the project on your own.
9. Email your solution to the project to `jan@math.uic.edu` before noon on Monday 25 October so the date of the email is proof of an on time submission. As a backup, bring also a printed version of your solution to class.

If you have questions or difficulties with the project, feel free to come to my office for help.