Assignment 1: Polynomial Multiplication

Overview

The purpose of this project is to give you practice in designing and implementing classes, and using existing class templates. You will design and implement a class to represent polynomials and perform polynomial multiplication. You will also write a main program that is a client of the class. The main program will read a sequence of polynomial definitions and operations from a file, and will create the specified polynomials and perform the specified operations. If you need to brush up on polynomial arithmetic, refer to any elementary algebra textbook.

Polynomial Arithmetic

A ***polynomial*** (in one variable) is a function of the form

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

where n is a non-negative integer, $c_n \neq 0$, and the other coefficients c_k may or may not be zero. The degree of a polynomial $p(x)$ is its largest exponent, n. No exponent in a polynomial is allowed to be negative. In this assignment, all coefficients are assumed to be integers. (In this case it is called a polynomial in an integer domain.) Note that if the degree of the polynomial is 0, then the polynomial is just a constant. For the remainder of this specification, I write p as a shorthand for $p(x)$.

Given two polynomials p and q, the following binary operation is defined:

```
p * q   is defined to be the product of polynomials p and q
```

Example

If $p(x) = 2x + 3$ and $q(x) = -x + 1$ then

$$p * q = (2x + 3) * (-x + 1) = -2x^2 - x + 3$$

Two polynomials are ***equivalent*** if they represent the exact same function. For example $p(x) = 2x^2 + 3$ and $q(x) = 2x^2 + 0x + 3$ are equivalent.

Project Details

The Class Interface

You are to design the interface to a `Polynomial` class. The class must contain the following member functions.

Name	Description
`eval(double x)`	Evaluate the polynomial using argument `x` and return the value. This should be implemented as an overloaded function call `operator().`
`operator*`	Given polynomials `p` and `q`, return `p * q.`
`operator<<`	Given an output stream `out` and polynomial `p`, write `p` in symbolic form on `out.`

In addition, the class needs

- a default constructor that creates a ***zero polynomial***, which is defined to be the number 0 (meaning that it has one term with $c = 0$ and $e = 0$);

- a constructor that takes a coefficient `c` and an exponent `e` and constructs a polynomial with the single term cx^e, e.g., `Polynomial p(c,e);`
- a copy constructor that takes a polynomial `q` and makes a new polynomial that is a copy of `q`, e.g. `Polynomial p(q);`
- an `operator=` for the class that will (copy) assign a polynomial to an existing polynomial;
- a destructor, which deletes the polynomial.

Note that this description makes no mention of the private part of the class. That is up to you. You will probably find it necessary to implement other private members of the `Polynomial` class.

Design and Implementation Requirements

A polynomial must be implemented using a C++ `list` object. The *C++* language includes a `list` class template, and you *must* use this class template instead of writing your own list class. That is my intention in giving you this assignment.

The `list` provides all of the functions that you will need. I suggest that your list nodes be ***terms***. Each term is completely defined by its coefficient and exponent. No two terms can have the same exponent; i.e., every node must have a unique exponent. *The list should not store any nodes whose coefficients are zero unless it is the zero polynomial.* It is a good idea to keep the nodes sorted by exponent value.

When two polynomials are added, some terms may cancel. For instance, if $p = 2x + 1$ and $q = -2x + 2$, and we let $r = p + q$, then $r = 3$ because the terms $2x$ and $-2x$ are canceled. Because multiplication requires adding polynomials, the product of two polynomials might have fewer terms than either polynomial. This implies that you need to check when a coefficient of the result is zero, and if so, delete that node.

The main program, class implementation, and class interface must each be in its own file. For this project there is no need to have more than three source code files.

Input and Output

The program expects a single command line argument that specifies a file to be opened for reading. If no file is specified, it is an error and the program must write an appropriate and meaningful error message onto the standard error stream, after which it must exit. If the file that is specified does not exist or cannot be opened for some reason, the program must write an appropriate and meaningful error message onto the standard error stream and then exit.

Assuming there were no usage errors, the program should allocate a vector that contains at least 100 zero polynomials. I will call it `Poly` here. It should then read input from the text file whose name was specified on the command line. Each line of the file will be in one of the following four forms:

1. `k` : c_1 e_1 c_2 e_2 ... c_n e_n
2. `n = m * k`
3. `eval n ( 6 )`
4. `print n`

There will be no empty lines. There will be whitespace separating all tokens on all lines. In the first form, the number before the colon, `k`, is the index in the vector where the polynomial should be stored. The line will be well-formed, so you do not have to do input validation. This means that for the first form of input line, there will be a set of pairs of numbers after the colon, each representing a term of the form $c_k x^k$. and `k` will be non-negative. The terms are not sorted by increasing or decreasing exponents. If the index `k` is greater than the largest index in the program's vector, the vector must be increased to a size at least as large as `k+1`. Examples of lines in the first form are:

```
3 : 6  5  -4  3  -2  1  2  0
4 : 1  2   3  4   -9  1
```

which define `Poly[3]` as $6x^5 - 4x^3 - 2x + 2$ and `Poly[4]` as $3x^4 + x^2 - 9x$.

The second form is an instruction to compute a product. It states that `Poly[n]` should contain the product of `Poly[m]` and `Poly[k]`, deleting anything that might have been in `Poly[n]` previously. If any of `m`, `k`, or `n` are larger than the largest index in `Poly`, then the vector is expanded so that all of `m`, `k`, and `n` are all valid indices. I.e., the vector is expanded to $max(k, m, n) + 1$. Remember that the vector elements that have not been explicitly assigned contain zero polynomials, so the product is always defined and may produce another zero polynomial.

The `eval` instruction specifies that the polynomial `Poly[n]` is to be evaluated with argument 6 and its value displayed on the standard output stream. The argument can be any floating-point number, such as `3.2`. The output should be something like

```
Polynomial[4](1.5) = 3.9375
```

The last form is an output instruction. The specified polynomial, `Poly[n]`, must be written onto the standard output stream in the format below. It does not need to be wrapped onto a new line if it seems long. The stream will do whatever wrapping needs to be done.

`Poly[n] =` a_1 `x^`e_1 `+` a_2 `x^`e_2 `+ ... +` a_m `x^`e_m

where the terms should be displayed in decreasing order of exponent and the letter `x` should be used as the term's variable. If a coefficient is zero, the associated term should not be displayed.

A small sequence of lines in the file might look like

```
0 : 1 2 2 1 1 0
1 : 1 1 1 0
3 : 4 5 1 2 3 0
2 = 0 * 1
4 = 3 * 2
6 = 5 * 0
print 2
eval 4 ( 5.5 )
0 = 1 * 2
1 = 0 * 2
print 1
```

Testing Your Program

You should design your own input files and test your program using your own input. You should carefully check that the output of your program is correct for the inputs you give to it. Try it on the simplest of polynomials to be sure. Try it on constants, on polynomials that cancel terms when multiplied, and so on.

Programming Rules

Your program must conform to the programming rules described in the Programming Rules document on the course website. It is to be your own work alone.

Grading Rubric

The program will be graded based on the following rubric, based on 100 points.

- A program that cannot run because it fails to compile or link on a `cslab` host receives only 20%. This 20% will be assessed using the rest of the rubric below.
- Meeting the requirements of the assignment, including performance requirements: 60%
- Design (modularity and organization): 15%

- Documentation: 20%
- Style and proper naming: 5%

This implies that a program that does not compile on a `cslab` host cannot receive more than 20 points.

Submitting the Assignment

This assignment is due by the end of the day (i.e. 11:59PM, EST) on February 27, 2017. Create a directory named named *`username`*`_project1`. where *username* is to be replaced by your CS Department network login name. Put all project-related source-code files into that directory. ***Do not place any executable files, data files, or object files into this directory.*** You will lose 1% for each file that does not belong there, and you will lose 2% if you do not name the directory correctly[1].

Next, create a zip archive for this directory by running the zip command

```
zip -r username_project1.zip ./username_project1
```

This will compress all of your files into the file named `username_project1.zip`. Do not use the tar compress utility.

There is a command named `submit_cs335_assignment` that I have written and installed on all `cslab` hosts. You must use this command to deposit your zip file into the right directory with the right name and permissions. To make it easier to do this, you should modify your `.bashrc` file by adding the following lines to the end of it:

```
# Add path to csci 335 bin directory
if [ -e /data/biocs/b/student.accounts/cs335_sw/bin ] ; then
    PATH=$PATH:/data/biocs/b/student.accounts/cs335_sw/bin
    export PATH
fi
```

After modifying `.bashrc` remember to source it by typing

```
. .bashrc
```

This will allow you to type the command `submit_cs335_assignment` without typing its entire pathname.

The command requires two arguments: the number of the project and the pathname of your file. Thus, if your file is named `username_project1.zip` and it is in your current working directory you would type

```
submit_cs335_assignment 1  username_project1.zip
```

The program will copy your file into the `project1` subdirectory

```
/data/biocs/b/student.accounts/cs335_sw/projects/project1/
```

and if it is successful, it will display the message, "`File ...  successfully submitted.`"

You will not be able to read this file, nor will anyone else except for me. But you can double-check that the command succeeded by typing the command

```
ls -l /data/biocs/b/student.accounts/cs335_sw/projects/project1
```

and making sure you see a non-empty file there.

If you put a solution there and then decide to change it before the deadline, just replace it by the new version. Once the deadline has passed, you cannot do this. I will grade whatever version is there at the end of the day on the due date. You cannot resubmit the program after the due date.

[1] I have scripts that process your submissions automatically and misnamed files force me to manually override them.