Swinburne University Of Technology

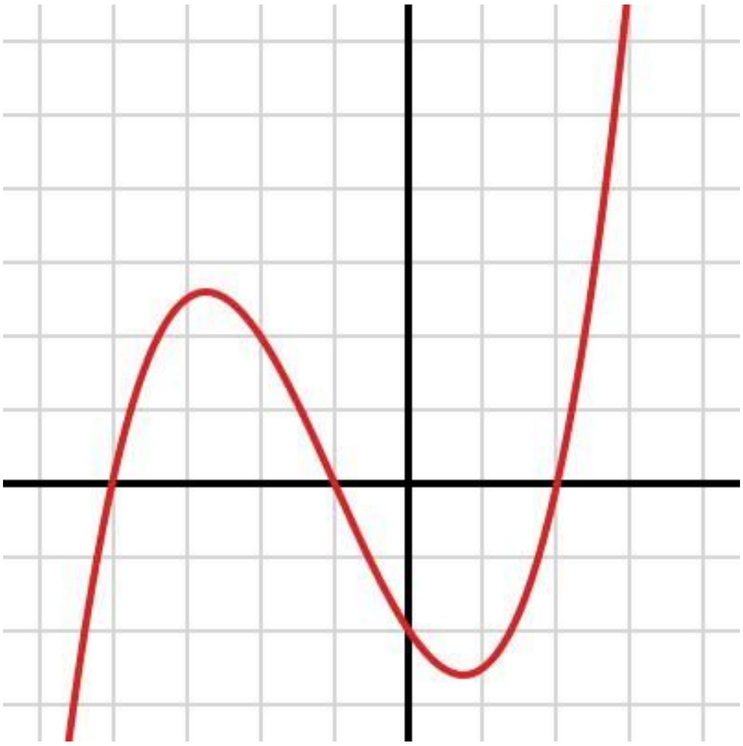
Faculty of Science, Engineering and Technology

LABORATORY COVER SHEET

Subject Code:	COS30008
Subject Title:	Data Structures and Patterns
Lab number and title:	3, Solution Design in C++
Author:	Dr. Markus Lumpe
Edited by:	Carmen Chai

However difficult life may seem, there is always something you can do and succeed at.

Steven Hawking



Solution Design in C++

The goal of this laboratory session is to build a C++ console application, called `Polynomials`, that allows users to specify the degree and coefficients of simple polynomials, multiply two polynomials, and output a human-readable representation.

A polynomial with a single variable x can be written in the form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x^1 + a_0$$

where $a_0, \ldots, a_n$ are floating-point numbers, and x is the variable of the polynomial. A polynomial can be expressed more concisely by using summation notation, which allows for a straightforward mapping to a standard for-loop in C++:

$$\sum_{i=0}^{n} a_i x^i$$

That is, a polynomial can be written as the sum of a finite number of terms $a_i x^i$. Each term consists of the product of a number a_i, called the coefficient, and a variable x raised to integer powers – x^i. The exponent i in x^i is called the degree of the term $a_i x^i$. The degree of a polynomial is the largest degree of any one term with a non-zero coefficient. For example

- $5x^0$ is a constant polynomial with degree 0,
- $2x^2 + 5x^1 + 3x^0$ is a polynomial of degree 2, that is, a quadratic function.

For the purpose of this tutorial, we limit the maximum degree of user-specified polynomials to 10.

In addition to representing polynomials, we also wish to support polynomial multiplication. Given two polynomials

$$\sum_{i=0}^{n} a_i x^i \quad \text{and} \quad \sum_{j=0}^{m} b_j x^j$$

the product is defined as

$$\sum_{i=0}^{n} a_i x^i * \sum_{j=0}^{m} b_j x^j = \sum_{i=0}^{n} \sum_{j=0}^{m} a_i b_j x^{i+j}$$

In order words, the product of two polynomials can be realized as a nested for-loop that aggregates the respective i^th and j^th polynomial terms. The maximum degree of the resulting polynomial is i+j. Since we allow 10 as the maximum user-specified degree for polynomials, our implementation must support polynomials up to degree 20 = 10 + 10.

Here's a video about Polynomials: https://www.youtube.com/watch?v=ffLLmV4mZwU

To facilitate the implementation, we shall use fixed-size arrays of double values to represent polynomials. All elements in the array have to be initialized to 0.0. For all non-zero coefficients a_i the array contains at index i the value a_i. As a result, the array arranges a given polynomial from right to left, that is, in increasing degree order.

The application should consist of two parts: a class `Polynomial` that implements the desired functionality and a `main` function that declares, reads, multiplies polynomials, and outputs the results to the Console. The specification of class `Polynomial` is shown below:

```
#pragma once

#include <iostream>

#define MAX_DEGREE 20+1     // max degree = 10 + 10 + 1, 0 to 20

class Polynomial
{
private:
        int fDegree;  // the maximum degree of the polynomial
        double fCoeffs[MAX_DEGREE]; // the coefficients (0..10, 0..20)

public:

        // the default constructor (initializes all member variables)
        Polynomial();

        // binary operator* to multiple to polynomials
        // arguments are read-only, signified by const
        // the operator* returns a fresh polynomial with degree i+j
        Polynomial operator*( const Polynomial& aRight ) const;

        // input operator for polynomials
        friend std::istream& operator>>(std::istream& aIStream,
                Polynomial& aObject);

        // output operator for polynomials
        friend std::ostream& operator<<(std::ostream& aOStream,
                const Polynomial& aObject);
};
```

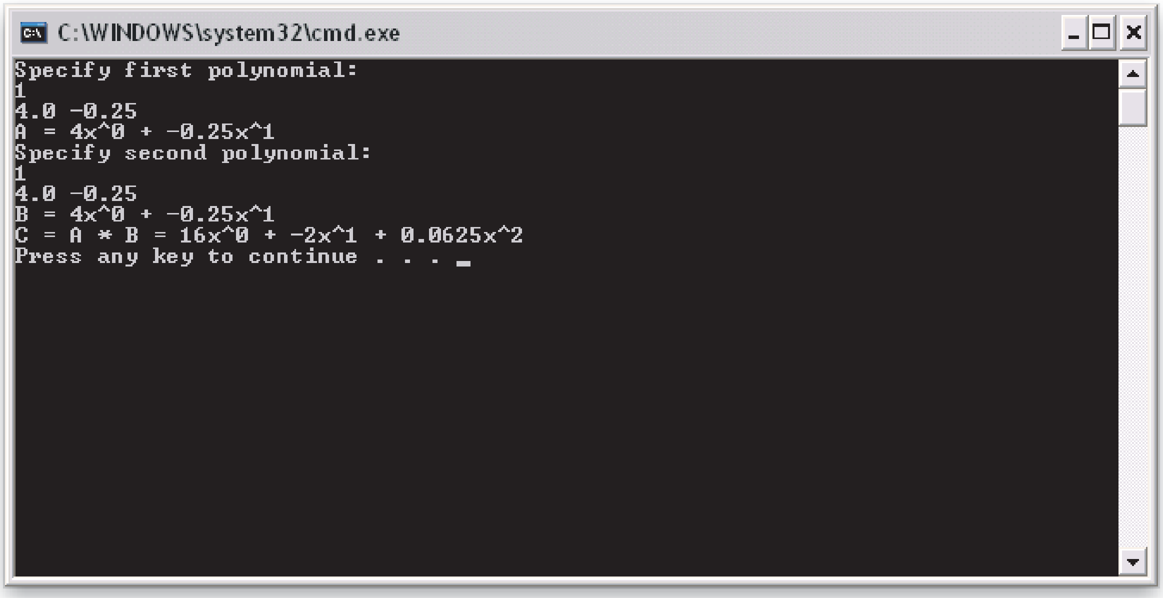
To implement the class `Polynomial` follow the process outlined in the lecture notes. First implement the constructor. Then implement `operator>>` and `operator<<`. The input operator requires two types of information: the degree (an integer value) and the corresponding number (degree+1) of coefficients (floating-point values). The output operator should only print the polynomial terms with non-zero coefficients. Finally, define the multiplication of polynomials.

(You may use this as reference: https://www.geeksforgeeks.org/multiply-two-polynomials-2/)

Use as main program the following code:

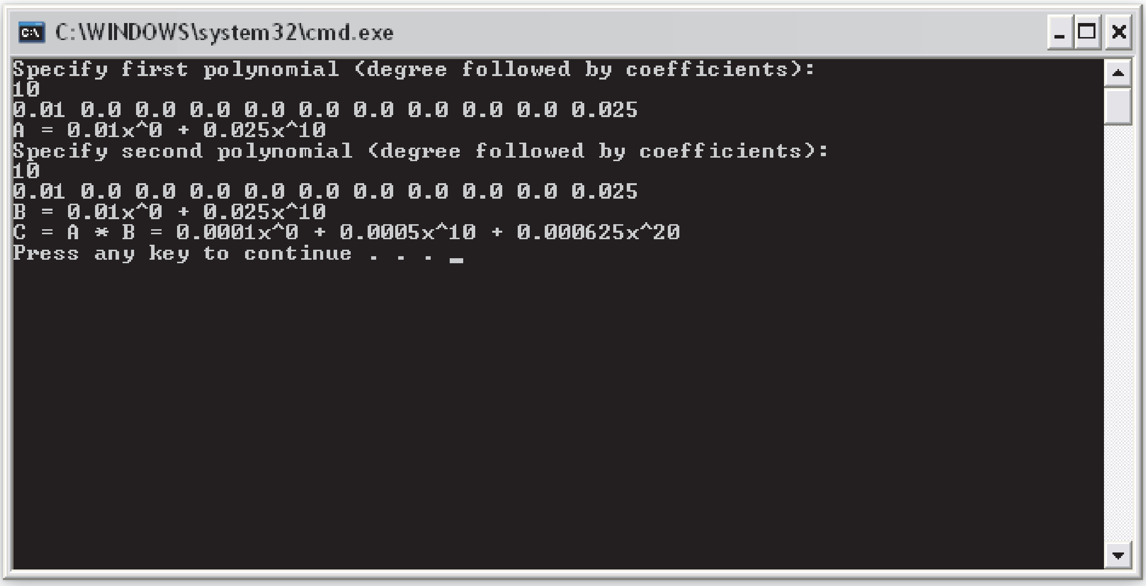
```
#include <iostream>
#include "Polynomial.h"
using namespace std;

int main()
{
        Polynomial A;
        cout << "Specify first polynomial (degree followed by coefficients):" << endl;
        cin >> A;
        cout << "A = " << A << endl;

        Polynomial B;
        cout << "Specify second polynomial(degree followed by coefficients):" << endl;
        cin >> B;
        cout << "B = " << B << endl;

        Polynomial C = A * B;
        cout << "C = A * B = " << C << endl;

        return 0;
}
```

Naturally, you can comment-out parts that you have not yet implemented. Once the implementation is complete, test your code as shown below (e.g., `-0.25x + 4.0`):



```
Specify first polynomial:
1
4.0 -0.25
A = 4x^0 + -0.25x^1
Specify second polynomial:
1
4.0 -0.25
B = 4x^0 + -0.25x^1
C = A * B = 16x^0 + -2x^1 + 0.0625x^2
Press any key to continue . . .
```

Your solution must support polynomials up to the 10th degree. For example, the polynomial $0.025x^{10} + 0.01$ must produce a result as show below:



```
Specify first polynomial (degree followed by coefficients):
10
0.01 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.025
A = 0.01x^0 + 0.025x^10
Specify second polynomial (degree followed by coefficients):
10
0.01 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.025
B = 0.01x^0 + 0.025x^10
C = A * B = 0.0001x^0 + 0.0005x^10 + 0.000625x^20
Press any key to continue . . .
```

You need to input:

```
10
0.01 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.025
```

The result of the multiplication is a polynomial of the 20th degree: $0.000625x^{20} + 0.0001$.

The solution requires 60-100 lines of low density C++ code.