**New York University**

**School of Professional Studies**

**Information Technologies Institute**

# C++ Part I

**Take Home Final Exam**

**Fall 2014 Yedidiah Solowiejczyk**

**INFO1\_CE9264-01 Wed 6:00 – 9:30 PM**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write C++ programs for each of the following questions. The **source code** and **execution output** should be provided separately. The exam is due **December 3, 2014**.

Write a C++ programs for each of the following questions. The source code and execution output should be provided separately.

1. Polynomials appear in a wide variety of areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary [word problems](http://en.wikipedia.org/wiki/Word_problem_(mathematics_education)) to complicated problems in the sciences; they are used in [calculus](http://en.wikipedia.org/wiki/Calculus) and [numerical analysis](http://en.wikipedia.org/wiki/Numerical_analysis) to approximate other functions; they are used in **error-correction algorithms** in communications systems. A typical polynomial can be described as follows:

**P(x) = 2x4 + 3x3 – 12x2 + x – 19 (4th order polynomial)**

or

**P(x) = 2x7 + 5x5 – 7x2 + x – 19 (7th order polynomial)**

**Where the coefficients for the first and second equations can be described by the following array of integers**

**Coeff1[ ] = {-19, 1, -12, 3, 2}**

**Coeff2[[ ] = {-19, 1, -7, 0, 0, 5, 0, 2}**

**Design and code a polynomial class in C++ that has the following properties:**

**class Poly{**

**private:**

**int order; //order of the polynomial**

**int \*coeff; // pointer to array of coeff on the heap**

**// size of coeff array predicated on (order + 1)**

**public:**

**Poly( ); //default constructor – order=0 & coeff[0] =1**

**Poly(int Order , int Default = 1) ;// creates Nth order poly**

**// and inits all coeffs**

**Poly(int Order, int \*Coeff); //creates an Nth polynomial & inits**

**~Poly( ); // destructor**

**::::::: // copy constructor**

**//mutators & accessors**

**void set( ){// Query user for coefficient values);**

**void set(int coeff[ ], int size); // input coeffs via external coeff vector**

**int getOrder( )const; // get order of polynomial**

**int \* get( ); //returns pointer to coeff array**

**//overloaded operators**

**Poly operator+( const Poly &rhs); // add two polynomials**

**Poly operator-( const Poly &rhs); // subt two polynomials**

**Poly operator\*( const int scale); // scale a polynomial**

**Poly operator\*(const Poly &rhs); // mult two polynomials**

**bool operator==(const Poly &rhs); // equality operator**

**const int & operator[ ](int I)const; // return the Ith coefficient**

**int & operator[ ](int I); // return the Ith coefficient**

**int operator( )(int X); // evaluate P(x) according**

**Poly & operator=(const Poly & rhs);**

**friend ostream & operator<<(ostream & Out, const Poly &rhs);**

**//other member functions**

**};**

**Demonstrate the following operations for the following Polynomials:**

**P1(x) = 2x4 + 3x3 – 12x2 + x – 19 (4th order polynomial)**

**P2(x) = 2x7 + 7x5 – 6x2 + x – 19 (7th order polynomial)**

**//display the following results for the polynomials defined above**

* **P3 = P1 + P2;**
* **P3 = P2 – P1;**
* **P3 = P1\*10;**
* **P3 = 10\*P1;**
* **P3 = P1\*P2;**
* **bool flag = (P1==P2);**
* **P1[3] = P2[5]; // assign the 5th coefficient of P2 to 3rd coefficient of P1**
* **int Z = P1(int X = 5); // evaluate Polynomial for input X**

**// suggest using Horner’s method**

* **The displayed polynomial for P2 should be printed as follows**

**2X^7 + 7X^5 – 6X^2 + 1X – 1**

1. **A class *Point* is defined by the following attributes and member functions:**

* **int x, y; // location on an x-y coordinate system**
* **default constructor(write message – dc)**
* **parameterized constructor(write message – pc)**
* **destructor**
* **copy constructor (write message – cc)**
* **overloaded assignment operator=()**
* **overloaded operator==()**
* **reset\_location(int X, int Y)**
* **getX( )const**
* **getY( )const**
* **overloaded int operator-(const Point &rhs) //distance between points**
* **overloaded operator\*(int scalar) // rescale P2=P1\*3;**
* **friend operator\*(int, Point &); //rescale P2=3\*P1;**
* **overloaded operator<<(ostream &os, Pointe &X); // prints coord of Point**

**Demonstrate the following cases:**

* **Point X(3,4), Y(10,40), Z;**
* **int distance = Y – X;**
* **Z = X\*10;**
* **Z = 10\*X;**
* **int x = X.getX();**
* **int y = X.getY();**
* **cout << “X = “, << X << endl;**
* **demonstrate void foo(Point & X, Point Y); // pass Point objects by reference and value that prints its x,y coordinates**

1. Create a C++ program that contains the following classes

**class Date{**

private:

int day, month,year;

**static const int daysMonth =**

**{31, 28,31, …………30, 31};**

**static const monthsYear = 12;**

public:

//constructors &destructors

// mutators & accessors

// calculate the number of days between

int **calcDiff**( const Date &Today));

};

**class Bond{**

private:

char \*name; // name of bond

float value;

const Date purchaseDate;

const Date maturityDate;

Date Today;

public:

//Constructors & use initialization list

//Destructors

//Accessors & Mutators

setToday( Date & **hoy**);

// Print function

//Methods

int daysToMaturity(Bond & rhs );

}; //end of class

b.

c. Demonstrate that the above class supports the following operations in the main program

* Date Today(4,18,2012);
* Date Maturity(12,31,2025);
* Date Purchase(2,28, 2012);
* Bond X; // forbidden
* Bond Y(“NYC\_Obligation”, 885.0, Purchase, Maturity);
* Bond \*bond\_ptr = new Bond(“GW\_Bridge\_Obligation”, 895.0, Purchase, Maturity)
* int days = Y.calcDaysMaturity(Today);

**Develop the code for the above class, debug and test.**

**Good Luck…….!!!!**

**Feel free to email me if you have any problems or questions. I encourage everyone to start looking at the problems this week.**

**ysolowie@yahoo.com**