## Programming 2 – Laboratory 10

Your task is to implement class **Polynomial** derived from class **Array**. Class **Array** contains dynamically allocated array of integer numbers.

### Part 1 (2 points)

In class Array define:
- private method `void init(int, int*);` which allocate and initialise memory. First parameter indicate size of memory to allocate, second is an address of array which contain values that should be used to initialise newly allocated memory. If latter is not given all elements are set to `0`
- constructor,
- copy constructor,
- operator= ,
- destructor.

Constructors and operator must use the method init()!

### Part 2 (2 points)

Define for class **Array:**
- operator+ : add two objects of **Array** of the same size,
- operator- : subtract two objects of **Array** of the same size,
- operator*= : multiply all elements of **Array** by integer number.

### Part 3 (1 point)

In class **Polynomial :**
- derive class **Polynomial** from class **Array** through private inheritance,
- add member field degree – degree of polynomial stored ,
- define constructor,
- define copy constructor, operator= and destructor (if needed),
- define operator<<.

### Part 4 (1 point)

Define for class Polynomial
- operator+ : add two **Polynomials** of the same sizes,
- operator- : subtract two **Polynomials** of the same sizes,
- operator*= : multiply **Polynomial** by integer number.

### Part 4 (2 points)

Define for class Polynomial
- public method derivative() – it return **Polynomial** which is derivative of the object **Polynomial**,
- operator()(double) – it calculate value of polynomial at point given by parameter.

Use Horner's method in order to calculate polynomial value at point . Eg.:

$$W(x) = 3x^3 - 2x^2 + x + 4 = x(x(3x - 2) + 1) + 4$$

Implement function **poly_root(**`const Polynomial &w, double x, int& it`**)** which calculates polynomial root using Newton's method.

Function **poly_root ()** returns the root or print out information and return given `x` value if algorithm was not convergent. **poly_root** takes three parameters: polynomial object `w` (in), start point `x` (in) and set number of performed iterations `it` (out by reference).

Newton's method:

Start from point $x_0$ (the function parameter). Consecutive approximations are calculated as:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Algorithm iterate until $|x_k - x_{k+1}| < eps$ or it reach local extremum $|f'(x)| < eps$.

Example main() function output:

```
****** Part 1 - (2 points ) *****

Empty Array
+0  +0  +0
+1  +2  +3
+1  +2  +3
+1  +2  +3

****** Part 2 - (2 points) *****

Empty Array
+3  -1  +4
+2  +4  +6

****** Part 3 - (1 point) *****

Empty polynomial

+3x^2+2x^1+1
+3x^2+2x^1+1
+3x^2+2x^1+1
Dynamic allocation :
+3x^2+2x^1+1
+1x^2-2x^1-8

****** Part 4 - (1 point)  *****

+6x^2+4x^1+2
Empty polynomial
+12x^2+8x^1+4

****** Part 5 - (2 points)  *****

p5 = +1x^2-3x^1+2
p5' = +2x^1-3
p5(+0.5) = +0.75
p5(+1) = +0
p5(+1.5) = -0.25
root p5(x0=0) = +1
root p5(x0=1.3) = +1
root p5(x0=1.6) = +2
Press any key to continue . . .
```