Imperial College
London

Software Engineering 2: Object Oriented Software Engineering

Week 5 – Dynamic memory, composition and inheritance – Lab [*]

Sahbi Ben Ismail (s.ben-ismail@imperial.ac.uk)

Our complete `polynomial`

Complete the `polynomial` class:

- Constructor(s)
- Member function `at(int)` to get the value of the element (i.e coefficient) at index `i` (and potentially overwrite it)
- Getter for `degree`
- Destructor
- Copy constructor
- Assignment operator
- Overloaded subscript operator `operator[]` (two versions, with and without const overloading)
- Overloaded insertion operator `operator<<` performing an appropriate display of a `polynomial`

Write a main testing all the functions, including the copy constructor and the assignment operator.

[*]Lab content originally written by Max Cattafi.

Try also the following template:

```
//Test 1) at(int) member function
polynomial p(2);
cout << p << endl;
p.at(0) = 2; p.at(1) = 3; p.at(2) = 1;
cout << p << endl;

//Test 2) copy constructor
polynomial p1(p);
cout << p1 << endl;

//Test 3) subscript operator[]
p[0] = 5; p[1] = -4; p[2] = 3;
cout << p << endl;

//Test 4) assignment operator
polynomial p2 = p;
cout <<p2 << endl;

p = p; //self assignment
polynomial p3 = p2 = p1; //chain assignment

cout << p1 << endl;
cout << p2 << endl;
cout << p3 << endl;
```

Classes relationships and introduction to inheritance

Without using inheritance from class `point`, write a class `labeled_point` representing a `point` with:

- Two coordinates `x` and `y` (or type `double`) and a `label` (of type `string`).
- Constructors with parameter lists: `()`, `(string)`, and `(double, double, string)`
- Getters and setters for `x`, `y`, `label` (and `distance_to_origin`)
- Member function `to_symmetric()`

- Member functions `distance_to(point)` and `distance_to(labeled_point)`
- Member functions `translate(point)` and `translate(labeled_point)`
- Overloaded insertion operator `operator<<`
- Overloaded comparison operators `operator==` and `operator<`

(Optional) Is it possible to compare an object instance of `point` with an object instance `labeled_point`? What should be added to classes `point` and/or `labeled_point` in order to make this code compile:

```
point p(1, 2);
labeled_point lp(2, 3, "A");

cout << "p == lp? " << (p == lp) << endl;
cout << "lp == p? " << (lp == p) << endl;

cout << "p < lp? " << (p < lp) << endl;
cout << "lp < p? " << (lp < p) << endl;
```