**Name:** Dauren Abdykaparov

**Student ID:** 20142001

**Course number:** CSE22101

**Data Structures.**

**Assignment #2. Report.**

**Plan:**

In the given problem we were not allowed to use STL vector, therefore I decided to write an own one. I planned to write `Vector` class and inherit `Polynomial` from it, but had a problem with type parameter. As a result, code of the `Vector` class was simply copied into `Polynomial` class.

**Implementation:**

The polynomial is stored in array. An element represents coefficient and its index represents the exponent.

**buffer:** is an array

**\_capacity:** is an actual size of `buffer` array

**\_size:** is a currently used size

**Polynomial():** empty constructor. All values set to zero.

**Polynomial(size):** constructor with given size. All array elements are set to zero.

**Polynomial(size,value):** constructor with given size and initial value. All array elements are set to `value`.

**Polynomial(source):** copy constructor

**~Polynomial():** destructor. Array is deleted.

**reserve(capacity):** increases the capacity of array. Doubles the array’s capacity until it is sufficient.

**resize(value):** calls **reserve()** and increases the size.

**= operator(source):** almost same as copy constructor

**[] operator (index):** if the `index` is out of range, reserve it. Return `buffer[index]`.

**capacity():** returns the capacity

**size():** returns the size

**empty():** returns true if size is zero and false otherwise.

**begin():** returns pointer to the beginning of the array.

**end():** returns pointer to the end of the array.

**front():** returns the first element.

**back():** returns the last element.

**push\_back():** reserves and adds element to the end of array.

**pop\_back():** erases the last element and decrements the size

**print():** prints the polynomial as written in the problem.

**+ operator(source):** creates a new `Polynomial` and fills it with the summation of elements of \*this and source.

**- operator(source):** creates a new `Polynomial` and fills it with the subtraction of elements of \*this and source.

**\* operator(source):** creates a new `Polynomial` and fills it with the multiplication of elements of \*this and source.

**Eval (x):** evaluates the polynomial for the given `x` using the `pow()` function from the `cmath` library.

**normalize():** erases the consequent zero-valued elements from the beginning.