Data Structures 2020

Homework #1

- (i) Make class *MyIntVector* similar to class *vector<int>* in STL, and also (ii) write a test program to check that all the member functions/operators of your class *MyIntVector* work correctly.

```
class MyIntVector {
        public:
                …
        private:
                int *data;
                …
};
```

Constraints

- Use a “pointer to an int” variable and dynamic memory allocation by the *new* operator.
  - int *data; // private member

- Do not use any static array!

Members to be Implemented

- Default constructor
  - MyIntVector( );
- Copy constructor for deep copy
  - MyIntVector(const MyIntVector& v);
- Destructor
  - ~MyIntVector( );
- Assignment operator (=) for deep copy
  - Chaining assignment should be possible.

Members to be Implemented

- Operator: +=
  - Appends RHS object to LHS one.
- Operator: [ ]
  - Returns a reference to the element at the requested position in the vector container.
  - If the requested position is out of range, it should output some messages and terminate the program.

Members to be Implemented

- (Binary) operator: +
  - Returns an object that is a vector-sum of the two operand objects.
- (Binary) operator: -
  - Returns an object that is a vector-difference of the two operand objects.
- (Binary) operator: *
  - Returns the scalar product (or dot product) value of the two operand objects.

Note that the above three operators are applicable only when the sizes of the two operands is the same.

Members to be Implemented

- (Unary) operator: -
  - Returns an object of which each element is the unary negation of the corresponding element in the operand object.
- (Binary) operator: ==
  - Returns whether or not the two operand vectors are the same. (You should check if their sizes are the same. Do not need to check their capacities.)
- (Unary) operator: ( )
  - Makes every element of this object be the value of the integer-valued (int-typed) operand.

Members to be Implemented

- void pop_back( );
  - Removes the last element in the vector, effectively reducing the vector size by one and invalidating all references to it.
- void push_back(int x);
  - Adds a new element at the end of the vector, after its current last element. The content of this new element is initialized to a copy of *x*.
- size_t capacity( ) const;
  - Returns the size of the allocated storage space for the elements of the vector container.

Members to be Implemented

- size_t size( ) const;
  - Returns the number of elements in the vector container.
- void reserve(size_t n);
  - Requests that the capacity of the allocated storage space for the elements of the vector container be at least enough to hold *n* elements.

Note that size_t is defined in the library cstdlib.

Members to be Implemented

- bool is_empty( ) const;
  - Returns whether the vector container is empty, i.e., whether or not its size is 0.
- void clear( );
  - All the elements of the vector are dropped: they are removed from the vector container, leaving the container with a size of 0 and a default capacity.