**Cairo University**

**Faculty of Computers and Artificial Intelligence**

# CS213

# Object Oriented

**Programming** **Assignment 3 Report**

**2022**

**Group: B**

**Muhannad ID: 20210411**

# Mina Albert ID: 20210417

Dr. Mohammad El-Ramly

## m.elramly@fci-cu.edu.eg

**Task 1 (mmVector Class):**

1- mmVector (int):a constructor takes an integer and initialize itsSize and itsCapacity to it and allocating array with that integer

2- mmVector (object\*, int ): a constructor takes array as a first parameter and its size and initialize our itsSize and itsCapacity to the array’s size and our mmVector object with elements of the array

3- mmVector(initializer\_list<object>) : a constructor takes initialize\_list as a parameter so we can write this : mmVector<int> temp = {1,2,3,4};

4 - mmVector (const mmVector&): a constructor that constructs a copy .

5- ~mmVector(): the destructor that deletes all allocated memory locations

6- mmVector &operator=(const mmVector&) : copies all data in the rhs mmvector object to the lhs(\*this)

7- mmVector &operator=(mmVector&&): moving resources from the rvalue object to the lhs object

8- object& operator[](int): returns the object at a specific index b reference

9- const object& operator[](int) const : same as 8 but for const objects

10- void push\_back(object): put the object passed as a parmeter at the end of our vector if the capacity is bigger than the size , if not we will increase the capacity of our vector to the (2\*size) and then putting our object at the end

11- object pop\_back(): returns the last element in the mmvector and decreases the size by one

12- void erase(iterator): erasing the element at a specified iterator by setting that element by shiting the elements that on the right side of that iterator to the left and then decrease the size by 1.

13- iterator begin(): returns an iterator to the first element in mmVector object

14- iterator end(): returns an iterator to the last element in mmVector object

15- void erase(iterator, iterator): erasing the elements between two iterators by shifing the elements after the second iterator to the left to the first iterator   
and then decreases the size by the difference between the two iterators

16- void clear(): setting the size to 0 and every element in mmVector to a constructed object by its type e.g: array[index] = data-type();

17- void insert(iterator, const object&): inserting an object to the specified iterator, first get the index of the location to insertion(by subtracting the iterator from the begin itreator) and then push back an object of data-type that our mmVector stores and the shifting the elements of the right to the specified index to the right .

18- bool operator==(const mmVector<object>&): returns true if both size and elements of each mmVector objects are equal otherwise, returns false

19- bool operator< (const mmVector<object>&): if size of first mmVector is less than second mmVector returns true otherwise, returns false.  
if sizes are equal, comparing each element.

20- int size() const: return the size of mmVector.

21- int capacity() const: returns the capacity of mmVector

22- void resize(int ) : if the new size of mmVector is negative, an exception is thrown.  
if new size is bigger than the capacity then we will increase the capacity by (2 \* old size) and then initializing elements that their indices from new size and bigger to a default value of the data-type our mmVector stores

23- bool empty(): return true if the size is 0, otherwise false.

24- void print() const: printing the elements of the mmVector

25- friend ostream& operator << (ostream& out,const mmVector<t>&rhs): looping over mmVector and printing the elements

**Task 1 (mmVector Class):**

Work Breakdown Table:

**Muhannad’s part**:

* mmVector(int);
* mmVector(object\*, int);
* mmVector(const mmVector&);
* ~mmVector();
* object& operator[](int);
* void push\_back(object);
* object pop\_back();
* iterator begin();
* iterator end();
* void clear();
* bool operator==(const mmVector<object>&);
* bool operator<(const mmVector<object>&);
* friend ostream& operator<<(ostream& out, const mmVector<t>& rhs);

**Mina’s part**:

* mmVector(initializer\_list<object>);
* mmVector &operator=(const mmVector&);
* mmVector &operator=(mmVector&&);
* const object& operator[](int);
* void erase(iterator);
* void erase(iterator, iterator);
* void insert(iterator, const object&);
* const int& size()const;
* const int& capacity() const;
* bool empty();
* void print()const;

**Task 2 (Individual Problems):**

Mina: 4,5,6

Muhannad: 1,2,3

**Group Bonus: Task 3(Doxygen) :**