
# FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE, CAIRO UNIVERSITY

## CS213: Programming II
## Year 2022-2023
## First Semester

## Assignment 2 – Version 2.0

**Course Instructors:**
Dr. Mohammad El-Ramly

**Revision History**

| | | | |
|---|---|---|---|
| **Version 1.0** | By Dr Mohammed El-Ramly | 26 Oct.2022 | Main Doc |
| **Version 2.0** | By Dr Mohammed El-Ramly | 26 Oct.2023 | Updated for 22/23 |

**CS213:** Object Oriented Programming
Assignment 2 (11 marks + (1.5 + 1.5) bonus) – Version 2.0

Cairo University, Faculty of Artificial Intelligence

## Objectives

This assignment aims to help you learn OOP concepts in C++ and how to use OOP modeling and design with C++ to build systems with intermediate complexity.

## Instructions

1. **Your Learning Is YOUR Responsibility. أنت المسؤول عن تعلمك و عن نجاحك**
2. **Deadline for part 1 is 1 of November 2023 @ 11:00 pm (Tasks 1 & 2)**
3. **Deadline for part 2 is 10 of November 2023 @ 11:00 pm (All tasks and bonus)**
4. Students will forms teams of **three** students **from the same lab/section.**
5. Please submit **only work that you did yourself**. If you copy work from your friend or book or the net or AI **you will fail the course**. تسليم حلول منقولة من أى مصدر يؤدى إلى الرسوب فى هذا المقرر فلا تغش الحل أو تنقله من أى مصدر و اسأل فى أى شئ لا تفهمه و لا تنقل الحل من النت أو زملائك أو أى مكان

## Task 0 (0 marks)

1. Review OOP C++ concepts and syntax.
2. Review the slides and code examples uploaded in the class and understand them very well.
3. Read this quick tutorial https://www.codesdope.com/cpp-introduction/
4. Create a **private GitHub** repo for the project. **Use it for development.** Everyone should use it.

## Task 1 (4 marks – Individual Work) – C++ Concepts and Problem Solving

Students will individuals solve the problem in Sheet 1 individually. Student with the smallest ID will solve problems 1, 4, 7, 10. The next will solve 2, 5, 8, 11. And the student with the largest ID will solve problems 3, 6, 9, 12. Code should be in **standard C++** not using third-party libraries.

Each file will be named A2_SectionNum_StudentID_ProblemNum.cpp.

Each file should have header similar to this:

```
// File: ……..cpp
// Purpose: ……..
// Author: ……..
// Section: ……..
// ID: ……..
// TA: ……..
// Date: 12 Oct 2023
```

## Task 2 (3 marks – Group/Individual Work) - Classes, objects & operator overloading

Students should divide the work as suggested below and then **they should integrate their code** together and make sure it works properly.

Different variations of types **`int`** and **`float`** exist in C++ and other languages. They are limited by minimum and maximum values depending on the number of bytes used to store the number. We need versions of these types with unlimited bounds. Java solves this problem by providing



**`BigInteger`** and **`BigDecimal`** classes. In this problem it is required to develop a a class **`BigReal`**. An object of this class represents a real number of any arbitrary length. The public interface of your class will be as follows. For functions where default implementation of C++ is enough and works fine, you do not need to re-implement them. You are free to store the parts of the real number in any way like string, array or vector. **The team should decide and agree on the data members of the class.** You should have suitable data members to store the integer and the fraction parts, the sign, the place of the decimal point (if needed) and the size (if needed) and any other info.

**Break your code into: header file, implementation file and application file**.

```
class BigReal {
  private:
        ...
        bool isValidReal (string realNumber) // True if correct real
  public:
        BigReal (double realNumber = 0.0);   // Default constructor
        BigReal (string realNumber);         // Initialize from string
        BigReal (const BigReal& other);      // Copy constructor
        BigReal& operator= (BigReal& other); // Assignment operator
        // Depending on how you store data, default can be enough
        void setNum (string realNumber)
        int size();
        int sign();

        BigReal operator+ (BigReal& other);
        BigReal operator- (BigReal& other);

        bool operator<  (BigReal anotherReal);
        bool operator>  (BigReal anotherReal);
        bool operator== (BigReal anotherReal);
        friend ostream& operator << (ostream& out, BigReal num);
};
```

It is required to separate your design into a header file and an implementation file. You should also validate that the passed parameter in case of string is a valid real number. Note that **1.** and **.1** are valid real numbers but **1.1.1** is not valid real number.

Your new class will support writing application programs like this:

```
BigReal n1 ("11.900000000000000000000000000000001");
BigReal n2 ("2333333333339.1134322222222292");
BigReal n3 = n1 + n2;
cout << n3;
n3 = n3 + BigReal (0.9);
```

Test your class with a small application and 12 test cases that test all these functions.

Each team member will take one of the parts of the class **after designing it carefully together. Work must be integrated in one working program.**

**CS213:** Object Oriented Programming
Assignment 2 (11 marks + (1.5 + 1.5) bonus) – Version 2.0

Cairo University, Faculty of Artificial Intelligence

**What to deliver?**

1- **Write code in standard C++ not using third-party libraries. Put in a separate directory.**
2- Name your file **A2_Task2_YourGroup_YourIDs.cpp** or **.zip** (if more than one file)
3- Code must have header like the one in Task 1 and detailed comments inside.
4- In the pdf report write the algorithms of +, -, <, > and isValidReal.
5- Add a **table showing who did which part.**

1. Please note the following
   - Using data encapsulation. Decide on a suitable container for storing the content of the number. Study the different options and find the ***efficient*** and ***elegant*** one to use in terms of (1) memory used and (2) suitable attributes and (3) speed of developing the program. Possible options include string, dynamic array, vector, or other containers in STL. These are details that are not important to the user of your class. This is called ***encapsulation* or *data hiding*.** You will need to build + and – operations that work on the representation you chose. The user does not care about which algorithm you use as long as it works. This is called ***algorithm hiding.***
   - Signs might be added to the number at initialization, but only –ve sign is printed. You may like keep this information about the sign in a separate attribute.
     - +111111111111111111.00000000000000000000000011111111111 // valid
     - -.000000000000009999999999999991111111111111111111999999 // valid
     - + 00099.999 // invalid
     - +-999000000 // invalid
     - 99900.011111 // valid
   - Both addition and subtraction should consider +v end –ve cases.
   - **See this https://www.geeksforgeeks.org/copy-constructor-vs-assignment-operator-in-c/**
   - **See https://ecomputernotes.com/cpp/classes-in-c/returning-object-from-function**
   - Team should help each other and support in each.
   - Team should integrate the work together and submit one working program.
   - Team should test the entire program and make sure that it works correctly in full.
   - All team members **must fully understand all parts of the program.**
   - **Use separate compilation**

**CS213:** Object Oriented Programming
Assignment 2 (11 marks + (1.5 + 1.5) bonus) – Version 2.0

Cairo University, Faculty of Artificial Intelligence

## Task 3 (4 marks - Group) - Classes, objects, abstraction and composition

Students should divide the work **as they like** and then **should integrate their code** together and make sure it works properly.

### Task 3.1 (0 marks – Individual) Review Vole Machine

Refer to CS111 materials and "CS an Overview Book" to review the Vole machine and language.

### Task 3.2 (0 marks – All the students in OOP class together) Machine Simulator Design

Model and design a simulator for the Vole machine and its language that is capable of simulating its operation and running the program. This design will be in the form of a **UML class diagram** that shows different classes and their relations (DO NOT SHARE CODE). The simulator will:

1- Offer a menu of choices
2- Allow the user to load a new program from a file
3- Fetch instructions a step by step to IR and validate it is a valid step and execute it
4- Allow the user to display the status of the registers, PC, IR, memory and screen at the end of program execution or after each step in a suitable format. (In text format)

All CS213 class work together designing the simulator with OOP concepts like classes, inheritance, encapsulation, abstraction and polymorphism. Probably, the model will have classes like Machine, Instruction (can be multiple types of it), Register, Memory or Memory Manager, etc. Classes will have suitable attributes and operations and also inheritance or association relations between them.

### Task 3.3 (4 marks – Group) Machine Simulator Development

Each team will implement its own version of the simulator. Data is taken as a file of space separated hex numbers representing the instructions, e.g. 0x1 0x0 0xFF means load R0 with memory content in location $255_d$. It is required to support these instructions only. Test your program on real Vole programs.

**Machine Language**

| Op-code | Operand | Description |
|---|---|---|
| 1 | RXY | LOAD the register R with the bit pattern found in the memory cell whose address is XY.<br>*Example:* 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4. |
| 2 | RXY | LOAD the register R with the bit pattern XY.<br>*Example:* 20A3 would cause the value A3 to be placed in register 0. |
| 3 | RXY | STORE the bit pattern found in register R in the memory cell whose address is XY.<br>*Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1. |
| **3** | **R00** | **STORE to location 00, which is a memory mapping for the screen. Writing to 00 is writing to screen.** |
| 4 | 0RS | MOVE the bit pattern found in register R to register S.<br>*Example:* 40A4 would cause the contents of register A to be copied into register 4. |
| 5 | RST | ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R.<br>*Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7. |
| 6 | RST | ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R.<br>*Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3. |
| B | RXY | JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the program counter during the execute phase.)<br>*Example:* B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the pattern 3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence. |
| C | 000 | HALT execution.<br>*Example:* C000 would cause program execution to stop. |

**CS213:** Object Oriented Programming
Assignment 2 (11 marks + (1.5 + 1.5) bonus) – Version 2.0

Cairo University, Faculty of Artificial Intelligence

**What to deliver?**

1- **Working code in standard C++ not using third-party libraries in a separate directory.**
2- Name your file **A2_Task3_YourSection_YourIDs.cpp** or **.zip** (if more than one file)
3- In the pdf report, include the detailed class diagram showing classes, relations and attributes.
4- In the report, include a **work break-down table** showing who did which part.

## Group Bonus 1: Task 4 (1.5 mark) – Static Code Analysis and Code Quality

Software is expensive to develop and lives for years. Many people over long time may work on the same program. It is important, **ethical**, **professional** and **economical**, to develop high quality code.

In this task, it is required for **all team** to cooperate in using one of the static analysis and code quality tools to analyze their solution for tasks 2 and 3. They can use student version of PVS studio https://pvs-studio.com/en/order/for-students/ or anther tool they have access to, e.g. SonarCube. (A video on PVS is here but there are many https://www.youtube.com/watch?v=vYW6TOwFK2M)

These tools look for weak and vulnerable parts of the code, bad practices, misuse of language constructs **and can enforce the application of coding style.** (to some extent)

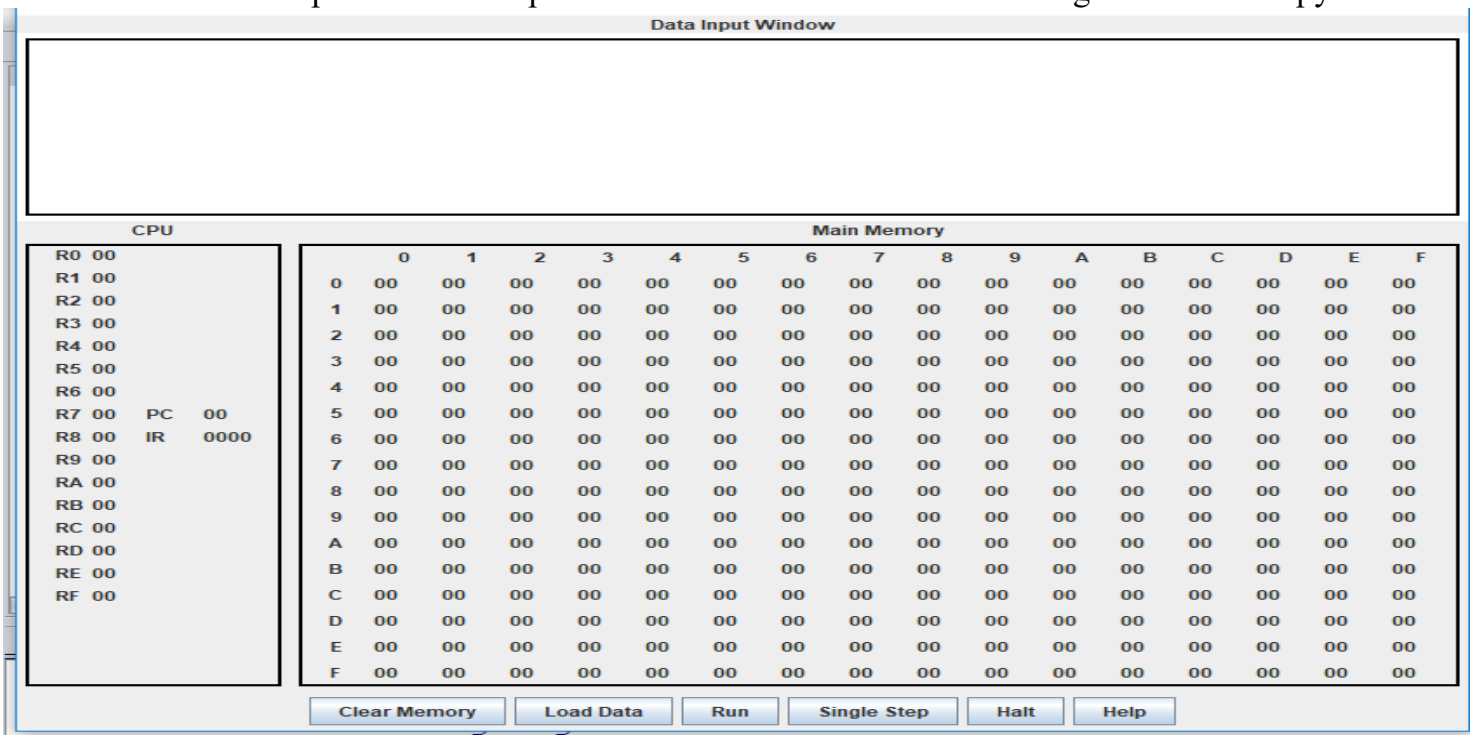
Analyze the code with tool, find all important quality issues & then fix them & produce better code.

**What to deliver:**

1- A written report of the experience with the tool and how useful it is and easy / hard to use.
2- A list of the most important issues found (they can be 100s, only report the important ones)
3- A modified and clean version of the code.
4- A **work break-down** table explaining who did what in this part.

## Group Bonus 2: Task 5 (1.5 mark) – GUI for the Simulator

1- Team will use a GUI library for C++ and develop a GUI for the simulator.
2- Simulator will support the full list of instructions of the Vole machine.
**3-** Be creative and develop something that is (1) Super easy to use and (2) Intuitive to understand. Here is a sample included of previous work but it is not the best design. DO NOT copy it.



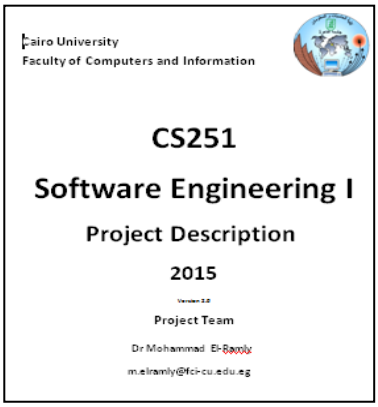
# **CS213:** Object Oriented Programming
# Assignment 2 (11 marks + (1.5 + 1.5) bonus) – Version 2.0

Cairo University, Faculty of Artificial Intelligence

**What to deliver:**

1- Snapshots of the GUI of the simulator
2- Modified Version of Simulator named **A2_Bonus2_YourSection_YourIDs.cpp** or **.zip**
3- A **work break-down** table explaining who did what in this part.

## Submission Instructions




**Team will submit into classroom the following:**

1. A zip file with the following components.
2. A pdf report with the following items.
   - The document should have a cover page like this.
   - A screen shot for GitHub for shared projects.
   - The required class diagrams, work break-down tables and reports needed in bonus tasks.
3. Team will create a project in GitHub to upload code there.
4. The source code of each program in a separate folder with suitable name for the folder. For student should put individual solutions for individual problems in a folder with his name and ID.
5. Each team member will work individually on his part. **But the team must provide ONE integrated and working program and report.**
6. Team members are expected to help each other but not do work of others.
7. Team members are responsible of testing all the programs and making sure they work.
8. **All team members must understand the details** of all programs and be able to explain it or even modify it if needed. TA can ask any team member about any of the programs developed**.**
9. **Ask your TA** about the discussion time of your work.

## Marking Criterion

1. 4.0 1 x 4 = 4 for developing a solution for the individual problems (Task 1).
2. 3.0 For correct and working BigReal implementation. If one part is missing, all team loses -1
   -1.0 If parts are not integrated in a working program.
3. 4.0 For a correctly working simulator that can load and run programs correctly.
4. 1.5 **Group Bonus1** for checking code quality by the tool and producing a high quality report
5. 1.5 **Group Bonus2** for producing a GUI simulator that supports all instructions.
6. -1.0 For not using GitHub

## Marking Instructions for TA

1. **Code that does not run does not exist. Ensure IT IS THEIR OWN CODE.**
2. Be merciful, kind and polite with the students.
3. Appreciate their effort and use marking as learning opportunity to educate them.
4. Ask every student in all code not only on what s/he did.
5. Ensure students fully understand the code, can explain it, and can rewrite it.
6. Be generous in marks. Always smile even if you are frustrated.