Homework 4

Structured Data

The following problem set is worth 100 points. Please submit a zip file containing all source code I need to run your assignments using **QtSPIM**. Code will be graded on elegance and correctness.

Problem 1: Complex Program (30pts)

Create a structure that holds a **complex number**. Recall complex numbers have the form:

```
realPart + imaginaryPart * i
```

Where $i = \sqrt{-1}$.

In your structure use **double** for each of the two components.

With your structure created, create the following functions:

Add	adds two complex numbers (passed as arguments $a0 and $a1) and returns a new structure in $v0
Subtract	subtracts two complex numbers in arguments $a0 and $a1 and returns a new structure in $v0
Print	prints a complex number in the form **(a, b)** with the structure passed as an argument in $a0

In your main method, ask the user for double prompts to collect data that creates two complex numbers. Call both add and subtract with the input and output the two resulting structures (calling the print method you created above in the process). Store your code in **complex.asm**.

Problem 2: Rational Problem (30pts)

Create a structure that holds a rational number. A rational number is a number that can be expressed in the form $\frac{a}{b}$. With **a** and **b** being stored as integers in your structure.

With your structure created, add the following functions in your code:

Add	Adds two rational numbers (input $a0 and $a1) and returns the result ($v0) in reduced form.
Subtract	Subtracts two rational numbers (input $a0 and $a1) and returns the result ($v0) in reduced form.
Multiply	Multiplies two rational numbers (input $a0 and $a1) and returns the result ($v0) in reduced form.

Divide	Divides two rational numbers (input $a0 and $a1) and returns the result ($v0) in reduced form.
Print	Prints the rational number input ($a0) in form A/B
PrintFloat	Prints the rational number input ($a0) in floating-point format.

In your main method, prompt the user for data that creates two rational numbers. Call each of your functions on the input and output the results. Store your code in **rational.asm**.

Problem 3: Overflow Problem (40pts)

In programming languages with static types, integer types are bound by a specified number of bytes. For instance, in Java, an integer is represented with 32 bits. If you perform arithmetic operations on two integers in java and it goes outside the bounds, this operation is considered an overflow (or underflow) operation. Java compensates for this limitation by providing classes such as **BigInteger** and **BigDecimal** which allow developers to work with unbounded number data.

With this stated, create a structure that represents a big integer. This structure can hold data for integers up to **40 digits in length**. Create the following functions with take two big integers as arguments (in $a0 and $a1):

Add	Returns a new big integer where the inputs are added together
Subtract	Returns a new big integer where the inputs are subtracted from one another
Eq	Returns a Boolean comparing the two arguments for equality
Ne	Returns a Boolean comparing the two arguments for inequality
Ge	Returns a Boolean comparing $a0 > $a1
Geq	Returns a Boolean comparing $a0 >= $a1
Le	Returns a Boolean comparing $a0 < $a1
Leq	Returns a Boolean comparing $a0 <= $a1

In your main prompt the user for digits for each of two big integers (prompting one digit at a time in a sentinel-controlled loop). Call each of your functions and output the results. Store your code in **bigInteger.asm**.

Submission Requirements

Please submit a zip file containing all source code I need for testing before the closing of the Canvas drop box.