CS 3520 - Programming Language Structures

Program: 1
Points: 40 points (20 points for Ruby version + 20 points for Java version)
Due Time: September 27, 11:59 pm

The objectives of this program are to enable students to practice how to code in Ruby, a newly learned scripting programming language, and gain a better understanding of the differences between Java and Ruby through solving the same problem.

For this assignment, you are going to create a program that does fixed point arithmetic, which is sometimes used in systems that need to perform floating point calculations, but the microprocessor chosen doesn't have a floating-point capability. The idea is that integer arithmetic is performed with an implied "binary" point. The details are given below, and you will need to create two classes: FixedPointNumber and FixedPointList. Prog1Java.java and prog1_ruby.rb will be provided.

Class FixedPointNumber

As we know, computers store integers as binary numbers. Assume an integer takes 4 bytes, that's 32 bits. The "left most" bit is the sign bit. For simplicity, in this program, we only consider **non-negative** integers and will ignore the sign bit. So, we assume an integer has 31 bits and ignore overflows.

For this program, you will represent a Fixed-point number as an integer value and a Q value. The Q value specifies where the implicit binary point is. For example, if Q = 2, then for the 31 bits of the integer value, the low 2 bits are the "fractional" part and the remaining 29 bits are for the integer parts. The 29 bits represent numbers from 0 to 536870911 (2 to the 29th power - 1). The 2 fractional bits can have values 0,1,2,3 which represent 0/4, 1/4, 2/4 and 3/4. If Q is 10, then the integer part has 21 bits (0 to 2097151) and the fractional part goes from 0/1024 to 1023/1024.

To convert a floating point number such as a double to a Fixed-point number with a given q, multiply the number by (2 to the qth power) and then truncate it as an int. The FixedPointNumber class can only have two pieces of data and no others, and the uncommented FixedPointNumber class in Java starts out as:

```
public class FixedPointNumber
{
   private int intVal;
   private int qVal;

   public FixedPointNumber(double x, int q)
   {
      intVal = (int)(x * Math.pow(2, q));
      qVal = q;
   }
}
```

Constructors

- One constructor in Java is given to you. No methods inside the class is allowed to use the constructor; only the user of the class can use the constructor. You can also use the constructor when testing the class.
- You need to make another constructor in Java that takes two integers and just assigns one integer to intVal and the other to qVal.
- Note that you cannot have multiple constructors in Ruby. You need to figure out a way to handle the different types of parameters in one single constructor `initialize`.

Methods

The following method signatures are given in Java. You need to have the corresponding methods in the Ruby version.

- **public double toDouble()**
  The method does the opposite of the first constructor: It divides intVal by (2 to the qVal power) and returns the resulting double. Inside the class, only the toString method is allowed to call the method.
- **public FixedPointNumber toQVal(int q)**
  It is critical to have a method to convert a FixedPoint number to another FixedPoint number with a different qVal. You use the shift operators << and >>, which shift the bits of an integer left or right. For example, suppose that a FixedPoint number has qVal 12. It will be a FixedPoint number with qVal 15 after intVal << 3, since 15-12 = 3; It will be a FixedPoint number with qVal 8 after intVal >> 4, since 8-12 = -4.
- **public String toString()**
  You must override the toString method, which must produce a string in the following form:

  ```
  intVal,qVal: toDouble_representation
  such as:
  50544,12: 12.339844
  ```

  where the float number is printed to 6 decimal places. You can do that using String.format, which allows you to format a string with variable arguments. The specifier %d means printing the argument as an integer and %f means as a floating point number. So toString can be:

  ```
  return String.format( "%d,%d: %.6f", intVal, qVal, toDouble() );
  ```

  In Ruby, you need to override to_s and may use `"%.6f" % [number]` for formatting.

- **public boolean equals (Object p)**
  You must override the equals method of the Object class. Two FixedPoint numbers are equal if their qVal and intVal are equal. In Ruby, you need to override ==.

- **public FixedPointNumber plus (FixedPointNumber p, int resultQ)**
  The method returns the sum of this object and p, and resultQ is the qVal of the sum. You should convert the two numbers to FixedPoint numbers with qVal being resultQ, then add their intVal to get the intVal of the sum.
  Note that the method must create a new FixedPoint number to return and is not allowed to change this object. So use temporary variables to do the conversions.
- Overflow could occur when performing fixed point arithmetic. For simplicity, we just ignore it.
- You can have some other private methods to be used inside the class.
- Only the toString method of the class can return a String. Callers of the toString method can only use it to print it.
- This class cannot read or write anything.

Class FixedPointList

The FixedPointList class is a growable container that stores FixedPoint numbers.

- The class must maintain a growable list of FixedPoint numbers. You can use Vector or ArrayList in Java.
- The class should have the following public methods. Remember to comment them according to the formatting rules.

```
// Handle input and output
public void run()
```

The commands you are to implement are:

```
Command     Meaning

A dbn     Add the fixed-point number for dbn to the end of the list
D dbn     Remove the first occurrence of dbn from the list
P         Print all fixed-point numbers in the list
Q num     Change the current Q value to the specified integer num
S         Sum up the numbers in the list and print the result
X         Normal termination
```

The Ruby version used by the Grader doesn't recognize `casecmp?` well. So find another way to check equality of strings.

```
// Allows duplicates
public void add(FixedPointNumber p)

// Deletes the first occurrence of a repeating element
public boolean delete(FixedPointNumber p)

// Returns the sum of all FixedPoint numbers in the list
// with the current qVal
```

```
// Returns 0.0 with qVal q when empty
public FixedPointNumber sum(int q)

// Returns "" when empty
// Otherwise, one line for each element with the string
// from toString followed by a new line character
@Override
public String toString()
```

- The class should maintain a current Q value to be used by the commands. The initial value is 12.
- For the A and D commands, dbn will be a double number in Java. You will make a FixedPoint number for dbn, and the Q value is the current Q value.
- The result of the S command must be in the current Q value.
- A command could be a single char.
- You must handle bad commands.
- The class should have a number of other methods to process the commands, and all of them must be private.
- See the sample input/output for the exact wordings. Note how for different Q values, the same number is stored with different accuracy!

Additional Notes:

- This is an individual assignment and you must complete it by yourself. Moderate help from others is accepted, but you have to make sure you understand the concepts.
- You must comment your code appropriately.
- You will lose points if you submit your program to the grader too many times:
  - -1 if you submit more than 10 times
  - -2 if you submit more than 15 times
  - -4 if you submit more than 20 times
- You will lose points if your final submission is after the due time:
  - -6 if your final submission is by 11:59 PM, September 28
  - -10 if your final submission is by 11:59 PM, September 29
  - -15 if your final submission is by 11:59 PM, September 30
  - No credit after 11:59 PM, September 30
- For all programs in this class, you must turn in a running solution in order to pass the course. The solutions do not have to work correctly for all test cases but must solve a significant portion of the problem.
- Match the output exactly and follow the Ground Rules below.

Ground Rules for this program:

- For Java code, follow camelCase for identifiers.
- For Ruby code, variable and method names are in lowercase and use underscores between words in a name.
- Use spaces between operators.

- For Ruby code, indent two spaces in all the normal places (if, while, for, etc.). For Java code, intent four spaces
- Comment at the top of each file saying what the class does and your name as the author.
- One- or two-line comment for each method.
- Methods must be less than 30 lines long.

Sample Input:

```
P
S
Q 9
A 123.46
a 123.49
Q 17
a 123.4
P
S
D 123.46
Q 9
A 123.49
P
D 123.49
P
q 15
A 123.5
P
S
A 43.3101
A 12.5
A 1.2
A 43.31
P
S
C 3.520
X
```

Sample Output:

```
All fixed-point numbers in the list are:
The sum is 0, 12: 0.000000.
Current q_value was changed to 9.
63211, 9: 123.458984 was added to the list.
63226, 9: 123.488281 was added to the list.
Current q_value was changed to 17.
16174284, 17: 123.399994 was added to the list.
All fixed-point numbers in the list are:
63211, 9: 123.458984
63226, 9: 123.488281
```

```
16174284, 17: 123.399994
The sum is 48542156, 17: 370.347260.
No value equal to 16182149, 17: 123.459999 in the list.
Current q_value was changed to 9.
63226, 9: 123.488281 was added to the list.
All fixed-point numbers in the list are:
63211, 9: 123.458984
63226, 9: 123.488281
16174284, 17: 123.399994
63226, 9: 123.488281
63226, 9: 123.488281 was deleted from the list.
All fixed-point numbers in the list are:
63211, 9: 123.458984
16174284, 17: 123.399994
63226, 9: 123.488281
Current q_value was changed to 15.
4046848, 15: 123.500000 was added to the list.
All fixed-point numbers in the list are:
63211, 9: 123.458984
16174284, 17: 123.399994
63226, 9: 123.488281
4046848, 15: 123.500000
The sum is 16182387, 15: 493.847260.
1419185, 15: 43.310089 was added to the list.
409600, 15: 12.500000 was added to the list.
39321, 15: 1.199982 was added to the list.
1419182, 15: 43.309998 was added to the list.
All fixed-point numbers in the list are:
63211, 9: 123.458984
16174284, 17: 123.399994
63226, 9: 123.488281
4046848, 15: 123.500000
1419185, 15: 43.310089
409600, 15: 12.500000
39321, 15: 1.199982
1419182, 15: 43.309998
The sum is 19469675, 15: 594.167328.
C is not a valid command!
Normal termination of program1.
```