CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS2360 Java Programming
Session	:	Semester A 2020/21
Time allowed	:	Two hours

This paper has NINE pages (including this cover page).

---

1. This paper consists of SEVEN questions.
2. There is a zip file containing SEVEN *incomplete* Java programs along with this paper.
3. Answer ALL the questions by *completing* the provided Java programs.
4. Name each program as QN_StudentID.java, where N is the question number. For example, **if your Student ID is 12345678, your program for question 1 should be** Q1_12345678.java and the class should be public class Q1_12345678
5. Copy all of your programs into one folder. Compress the folder in .zip format. Submit the .zip file to Canvas.
6. **Marking scheme**: The paper is graded based on the number of test cases being corrected outputted by your programs.
7. **Note**: No error checking is required unless otherwise stated in the question.

This is an ***open-book*** *examination*

Students are allowed to use the following materials/aids:

Books, lecture notes
Eclipse for Java programming

Materials/aids other than those stated above are not permitted. Candidates will be subject to disciplinary action if any unauthorized materials or aids are found on them.

Question 1 (15 marks)

The program **Q1.java** reads in two numbers, one as an integer and another one as a string. The program uses the class `MyInteger` to check whether these numbers are prime numbers.

However, the program has some syntax and logical program. **Debug Q1.java** so that it produces the desired outputs as followings.

Note: The inputs by user are underlined.
Marking scheme: 10 marks for no compilation error; 5 marks for correct outputs. Your modified program must use `MyInteger` to check whether a number is a prime number.

```
Input an integer: 1997
Input a string in digits: 2046
Is 1997 a prime number? true
Is 2046 a prime number? false
```

```
Input an integer: 8001
Input a string in digits: 101
Is 8001 a prime number? false
Is 101 a prime number? true
```

Question 2 (10 marks)

Complete Q2.java to reads 5 positive integers. The program should output true if the sum of even integers is more than the sum of odd integers.

Note: The inputs by user are underlined.

```
Enter five integers: 2 5 7 9 10
Answer: false
```

```
Enter five integers: 2 3 7 5 20
Answer: true
```

Question 3 (15 marks)

Twin prime is a pair of prime numbers that are reverse of each other. For example, (13,31) are twin primes, (17,71) are twin primes, and (37,73) are twin primes.

Complete <u>the main method</u> in the program Q3.java to display the first *N* twin primes from (13,31) using the following two methods provided in the program. The value of *N* is specified by user.

```
public static boolean isPrime(int number);
public static int reverse(int number);
```

The `isPrime` method will return true if a given number is prime, and the `reverse` will return the reverse of a given number.

Note: The inputs by user are underlined.

```
Input a value for N: 2
(13,31)
(17,71)
```

```
Input a value for N: 5
(13,31)
(17,71)
(31,13)
(37,73)
(71,17)
```

```
Input a value for N: 10
(13,31)
(17,71)
(31,13)
(37,73)
(71,17)
(73,37)
(79,97)
(97,79)
(101,101)
(107,701)
```

Question 4 (15 marks)

Complete the method `m(int i)` in the program Q4.java. The method computes and returns the value of the following series:

$$m(i) = \frac{1}{3} + \frac{2}{5} + \ldots + \frac{i}{2i+1}$$

Note: The inputs by user are underlined.

```
Input a value for i: 1
0.33
```

```
Input a value for i: 2
0.73
```

```
Input a value for i: 12
5.37
```

Remark:
The program should display only two decimal points by *truncating* the remaining decimal points.

Question 5 (15 marks)

The program **Q5.java** has an array named `database` that stores 100 distinct positive integers in ascending order.

Complete the method `miss` that returns `how many integers` from 1 to `n` (including `n`) are not stored in the `dataset`.

```
public static int miss(int[] database, int n)
```

Note: The inputs by user are underlined.

```
Input n: 4
Answer: 0
```

Remark: The integers 1, 2, 3, 4 are found in the database.

```
Input n: 10
Answer: 5
```

Remark: The integers 5, 6, 7, 8, 9 are not found in the database.

```
Input n: 25
Answer: 11
```

Question 6 (15 marks)

In **Q6.java**, there is a class name *LinearEquation*. The class is designed to find the point of intersection between two line equations, as following:

$$ax + by = e$$
$$cx + dy = f$$

And the solutions for *x* and *y* are:

$$x = \frac{ed - bf}{ad - bc} \qquad y = \frac{af - ec}{ad - bc}$$

Complete the class *LinearEquation* with the following two methods:

- A constructor with the arguments for *a, b, c, d, e* and *f*
- A method named `isSolvable()` that returns *true* if *ad-bc* is not 0
- Methods `getX()` and `getY()` that return the solutions based on the equations of *x* and *y* respectively.

Use the class *LinearEquation*, complete the main method to display the intersections of two lines as following:

```
Enter the endpoints of the first line segment: 2 2 0 0
Enter the endpoints of the second line segment: 0 2 2 0
The intersecting point is: (1.0, 1.0)
```

Note that the program reads two endpoints (*x1,y1*) and (*x2,y2*) of a line segment. In the above example, the endpoints of the first line is `(2,2)` and `(0,0)`, and the endpoints of the second line is `(0,2)` and `(2,0)`.

Given the endpoints (*x1,y1*) and (*x2,y2*), you can find the *a*, *b* and *e* of the equation $ax + by = e$ as:

$$a = y1 - y2$$
$$b = (-x1 + x2)$$
$$e = -y1(x1 - x2) + x1(y1 - y2)$$

Similarly, the *c*, *d*, and *f* of the equation $cx + dy = f$ can be found using the same formulas.

More sample outputs are listed in the next page.

Note: The inputs by user are underlined.

Note: You **MUST** use *LinearEquation* to calculate the point of intersection. Otherwise, no mark will be given.

```
Enter the endpoints of the first line segment: 2 2 0 0
Enter the endpoints of the second line segment: 0 2 2 0
The intersecting point is: (1.0, 1.0)
```

```
Enter the endpoints of the first line segment: 2 4 6 8
Enter the endpoints of the second line segment: 1 2 3 4
The intersecting point is: Not solvable
```

Remark: The equations are not solvable because *ad-bc=0*

```
Enter the endpoints of the first line segment: 1 3.5 2 2
Enter the endpoints of the second line segment: 1 -0.2 2 -0.5
The intersecting point is: (4.083333333333334, -1.1250000000000002)
```

Question 7 (15 marks)

A number is a *palindrome* if its reversal is the same as itself. For example, 404 is a palindrome because the reversal of 404 is also 404.

Complete **Q7.java** by using the class `StackOfIntegers` to store the palindrome numbers within `m` and `n` as specified by users in the stack, and then **print these numbers in decreasing order**.

The UML diagram of the Java class `StackOfIntegers` is as below:

StackOfIntegers	
-elements: int[]	An array to store integers in the stack.
-size: int	The number of integers in the stack.
+StackOfIntegers()	Constructs an empty stack with a default capacity of 16.
+StackOfIntegers(capacity: int)	Constructs an empty stack with a specified capacity.
+empty(): boolean	Returns true if the stack is empty.
+peek(): int	Returns the integer at the top of the stack without removing it from the stack.
+push(value: int): int	Stores an integer into the top of the stack.
+pop(): int	Removes the integer at the top of the stack and returns it.
+getSize(): int	Returns the number of elements in the stack.

Note: The inputs by user are underlined.

Note: No mark will be given if your program does not use `StackOfIntegers` to store the palindrome numbers.

```
Input m: 101
Input n: 151
151 141 131 121 111 101
```

```
Input m: 200
Input n: 250
242 232 222 212 202
```

- END -