Lab 3 : Programming with Decisions and Loops – Selected Solutions

Exercise 3: Write a program that determines whether an input integer is even — but you can't use the modulus operator `%`: for this one you'll have to use *casting*. There are several ways to do this.

It may be helpful to draw a diagram of the logic part of this exercise to see if you can figure out how to do it.

```java
import java.util.Scanner;

public class EvenNoModulus {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter a number:");
      int input = keyboard.nextInt();

      // Method 1:
      // Check if dividing the number by integer 2 is the same as
      // dividing it by floating point 2.0. If the result is the
      // same, the number must be even as integer division did not
      // lose any precision.
      boolean isEven = ((input / 2) == (input / 2.0));

      // Method 2:
      // Check if doing integer division then multiplying the
      // result back will keep the same number
      isEven = ((input / 2 * 2) == input);

      if(isEven) {
         System.out.println("The number is even.");
      } else {
         System.out.println("The number is odd.");
      }
   }
}
```

Exercise 4: Part 'A' Create a `NumberCrunch` class, and make it read in **up to** three integers from the user. As soon as the user inputs a negative number, the program should stop trying to read numbers. Display to the user how many positive numbers were read in.

Exercise 5: Part 'B' If the user has not entered any positive numbers, `NumberCrunch` should tell the user to input at least one positive number.

Exercise 6: Part 'C' If the user has entered exactly two positive numbers, `NumberCrunch` should print out the product of the two numbers, as well as their relationship (equal, greater than, less than).

Exercise 7: Part 'D' If the user has entered exactly three positive numbers, `NumberCrunch` should print out the largest of the three numbers.

Exercise 8: Part 'E' If the user has entered exactly one positive number, `NumberCrunch` should print out all factors of that number. Remember: y is a factor of x if, when you divide x by y, there's no remainder. You should use the *modulus* operator `%` for this.

You will have to use a loop for this exercise.

```java
import java.util.Scanner;

public class NumberCrunch {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please enter up to three positive numbers:");

        int n1 = keyboard.nextInt();
        if(n1 < 0) {
            System.out.println("You have not entered any positive numbers. " +
                  "Please input at least one positive number.");
            return;
        }

        int n2 = keyboard.nextInt();
        if(n2 < 0) {
            // Part 'E': printing factors of 1 number
            System.out.println("You entered 1 positive number.");
            System.out.print("The factors of " + n1 + " are: ");
            // print all factors except n1 itself on one line
            int factor = 1;
            while(factor < n1) {
                if(n1 % factor == 0) {
                    System.out.print(factor + ", ");
                }
                factor++;
            }
            // print final factor, which is always the number itself.
            System.out.println(n1 + ".");
            return;
        }

        int n3 = keyboard.nextInt();
        if(n3 < 0) {
            // Part 'C': product and relationship
            System.out.println("You entered 2 positive numbers.");
            System.out.print("Their product is " + (n1 * n2) + " and ");
            System.out.print(n1);
            if(n1 < n2) {
                System.out.print(" is less than ");
            } else if(n1 > n2) {
                System.out.print(" is greater than ");
            } else {
                System.out.println(" is equal to ");
            }
            System.out.println(n2 + ".");
            return;
        }

        //Part 'D': largest number
        System.out.print("The largest number is ");
        if(n1 > n2 && n1 > n3) {
            System.out.println(n1 + ".");
        } else if(n2 > n1 && n2 > n3) {
            System.out.println(n2 + ".");
        } else {
            System.out.println(n3 + ".");
        }
    }
}
```

Extensions

Extension: Modify your `NumberCrunch` program so that when the user inputs a single number, the program prints out all **prime** factors of that number. Prime factorisation is not as simple as regular factorisation!

```java
import java.util.Scanner;

public class NumberCrunch {
   public static void main(String[] args) {
      // ... code from previous version ...
      if(n2 < 0) {
         System.out.println("You entered 1 positive number.");
         System.out.print("The prime factors of " + n1 + " are: ");

         boolean firstOutput = true;
         int input = n1;
         while(input > 1) {

            int factor = 2;
            boolean isPrime = false;
            // Find the next prime factor
            while(!isPrime) {
               // Find the next factor
               while(input % factor != 0) {
                  factor++;
               }
               // Check if it is prime
               isPrime = true;
               int i = 2;
               while(i <= Math.sqrt(factor)) {
                  if(factor % i == 0) {
                     isPrime = false;
                     break;
                  }
                  i++;
               }
            }
            // Output the factor and reduce the value by
            // the appropriate amount
            if(firstOutput) {
               firstOutput = false;
            } else {
               System.out.print(", ");
            }
            System.out.print(factor);
            input /= factor;
            factor = 2;
         }
         System.out.println(".");
         return;
      }
      // ... code from previous version ...
   }
}
```

Note: there are much more efficient ways to do this, using arrays and sieves (such as the Sieve of Eratosthenes), however this method uses simple loops.

Extension: Modify your `NumberCrunch` program so not all the functionality is built in the `main` method.

```java
import java.util.Scanner;

public class NumberCrunch {
   public static void main(String[] args) {
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Please enter up to three positive numbers:");

      // .. code from Part B ..

      int n2 = keyboard.nextInt();
      if(n2 < 0) {
         printFactors(n1);
         return;
      }

      int n3 = keyboard.nextInt();
      if(n3 < 0) {
         printProductAndRelationship(n1, n2);
         return;
      }

      printMax(n1, n2, n3);
   }

   public static void printFactors(int a) {
      System.out.println("You entered 1 positive number.");
      // .. code from Part E ..
   }

   public static void printProductAndRelationship(int a, int b) {
      System.out.println("You entered 2 positive numbers.");
      // .. code from Part C ..
   }

   public static void printMax(int a, int b, int c) {
      System.out.println("You entered 2 positive numbers.");
      // .. code from Part D ..
   }
}
```