Lab 3 : Programming with Decisions and Loops – Selected Solutions

Exercise 5: 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.

```
1  import java.util.Scanner;
2
3  public class EvenNoModulus {
4     public static void main(String[] args) {
5        Scanner keyboard = new Scanner(System.in);
6        System.out.println("Enter a number:");
7        int input = keyboard.nextInt();
8
9        // Method 1:
10       // Check if dividing the number by integer 2 is the same as
11       // dividing it by floating point 2.0. If the result is the
12       // same, the number must be even as integer division did not
13       // lose any precision.
14       boolean isEven = ((input / 2) == (input / 2.0));
15
16       // Method 2:
17       // Check if doing integer division then multiplying the
18       // result back will keep the same number
19       isEven = ((input / 2 * 2) == input);
20
21       if(isEven) {
22          System.out.println("The number is even.");
23       } else {
24          System.out.println("The number is odd.");
25       }
26    }
27 }
```

Exercise 6: Create a `LeapYear` class to read a year from the **command-line arguments**, and output if this year is a leap year. To determine whether a year is a leap year, follow these steps:

Step 1: If the year is divisible by 4, go to step 2. Otherwise, it is not a leap year.
Step 2: If the year is divisible by 100, go to step 3. Otherwise, it is a leap year.
Step 3: If the year is divisible by 400, it is a leap year. Otherwise, it is not a leap year.

```
1  public class LeapYear {
2      public static void main(String[] args) {
3        int year = Integer.parseInt(args[0]);
4        boolean isLeap;
5        //Method 1:
6        //step 1
7        if (year % 4 == 0) {
8           //step 2
9           if (year % 100 == 0) {
10             //step 3
11             if (year % 400 == 0){
12                isLeap = true;
13             } else {
14                isLeap = false;
15             }
16          } else {
17             isLeap = true;
18          }
19       } else {
20          isLeap = false;
21       }
22
23       //Method 2 (compressed version):
24       if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)) {
25          isLeap = true;
26       } else {
27          isLeap = false;
28       }
29
30       if(isLeap) {
31          System.out.println(year + " is a leap year!");
32       } else {
33          System.out.println(year + " is not a leap year!");
34       }
35     }
36 }
```

Exercise 7: 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. If the user has not entered any positive numbers, `NumberCrunch` should tell the user to input at least one positive number.

Exercise 8: Part 'B' 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.

Exercise 9: Part 'C' If the user has entered two or three positive numbers, `NumberCrunch` should print out the largest of the input numbers.

```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) {
         System.out.println("You entered 1 positive number.");
         // Part 'B': printing factors of 1 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) {
         System.out.println("You entered 2 positive numbers.");
         // Part 'C': largest of 2 numbers
         System.out.print("The largest number is ");
         if(n1 > n2) {
            System.out.println(n1 + ".");
         } else {
            System.out.println(n2 + ".");
         }
         return;
      }

      System.out.println("You entered 3 positive numbers.");
      //Part 'C': largest of 3 numbers
      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 1: 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!

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

```
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 numbers.");
         printFactors(n1);
         return;
      }

      int n3 = keyboard.nextInt();
      if(n3 < 0) {
         System.out.println("You entered 2 positive numbers.");
         printMax(n1, n2, -1);
         return;
      }

      System.out.println("You entered 3 positive numbers.");
      printMax(n1, n2, n3);
   }

   public static void printFactors(int a) {
      // .. code from Part B ..
   }

   public static void printMax(int a, int b, int c) {
      // .. code from Part C, largest of 3 numbers ..
   }
}
```