COE528 (W 2022)
Lab2

Duration: one week.

Note:

1. Every lab assignment must be done individually.
2. When you name a folder or a file, you **should** avoid spaces in those names. For example, if you need to name a folder as **GreenApple**, you should name it as **GreenApple** instead of **Green Apple**

Objective

- Design and implement procedures, based on their given specifications.

The specification for a procedural abstraction contains:

(a) **Requires:** This clause states any constraints under which the procedure will work. This is an optional clause.

(b) **Modifies:** This clause lists the names of any inputs that are modified by the procedure. This is an optional clause.

(c) **Effects:** This clause describes the behaviour of the procedure for all inputs that are not ruled out by the Requires clause.

Ex1: Implementation of `reverseFactorial` procedure

Design and implement a procedure named **`reverseFactorial`**. This procedure should take one integer parameter `x`. When `x` is a positive integer, this procedure should return the smallest positive integer `n` for which `n!` (i.e. 1*2*3*…*n) is greater than or equal to `x`. For example:
`reverseFactorial(24)` should return 4 since (1*2*3*4) = 24 but (1*2*3) < 24;
`reverseFactorial(119)` should return 5 since (1*2*3*4*5) > 119 but (1*2*3*4) < 119.

```
//Requires: None
//Modifies: None
//Effects: Returns the smallest positive integer n for which n!
//         (i.e. 1*2*3*…*n) is greater than or equal to x, for positive
//         integer x. Otherwise returns 1.

public static int reverseFactorial(int x) {
    //write the code here



}
```

Ex2: Implementation for `isMatrixNice` procedure

Design and implement a procedure named **`isMatrixNice`**. A matrix is called **Nice** matrix if it satisfies the following properties:

- The matrix is a square matrix. For example 2x2 or 3x3. And,
- The sum of the integers in each row, each column and both the diagonals are same.

For example, $\begin{matrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{matrix}$ is a **Nice** matrix. However, $\begin{matrix} -3 & 1 & 0 \\ 4 & -3 & 4 \\ 7 & -9 & 0 \end{matrix}$ is not a **Nice** matrix.

The procedure **`isMatrixNice`** should take a matrix (i.e. a two-dimensional array) of integers, `arr`, as its only parameter. If `arr` is a **Nice** matrix, the procedure should print the sum and return true. Otherwise returns false.

```
//Requires: None
//Modifies: None
//Effects: If the matrix arr satisfies Nice property, prints the sum and
//         returns true. Otherwise returns false.

public static boolean isMatrixNice(int[][] arr) {
    //write the code here



}
```

Step 1: *Create a Netbeans project*

1. Create a Netbeans project called `coe528Lab2`
2. Create a Java class called `ProceduralAbstraction.` Set the package to `lab2`
3. Implement the `reverseFactorial` procedure inside `ProceduralAbstraction` class as described in Ex1.
4. Implement the `isMatrixNice` procedure inside `ProceduralAbstraction` class as described in Ex2.

Step 2: Submit your lab

Deadline: See announcement in D2L for deadline.

Create a folder. Name it as YourLastname_YourFirstname_coe528_Labnumber.
Example: If student name is John Smith, the name of folder should be
Smith_John_coe528_Lab2.

Copy your Netbeans project folder in the above folder.

The above folder must also contain a duly filled and signed standard
cover page. The cover page can be found on the departmental web site:
Standard Assignment/Lab Cover Page

Compress the above folder as a single zip file that is named
according to the following rule:
YourLastname_YourFirstname_coe528_Labnumber.zip.
Example: Smith_John_coe528_Lab2.zip.

Upload the above zip file on D2L through the "Assessment" > "Assignments" link.

Within the deadline, you may re-submit (i.e. re-upload) the aforementioned zip file. However, note that your latest submission will always overwrite your previous submission.

Submission by email is NOT accepted.

Note: If the code does not compile, the submission will receive a ZERO mark.