# COSC 237

## Assignment # 1

**DUE DATE: 09/29/2022 (Thursday)**

For this assignment (review on **structured programming)**, you will work in teams of (at most) three students, but all teammates should contribute seriously (please let me know if that's not the case). Submit the source files:

1. e-mail (izimand@towson.edu) - only one e-mail for the entire team (COSC237.section, your name, Assignment#1 in the Subject box). **You must Cc** all your teammates - I may need to contact all of you as a group. **AND**

2. turn in a hard copy of the assignment with all your names on it. Please staple all pages together before coming to class (**a must**!)

In principle, you should not discuss the homework with anyone, except your partners and me. If you do discuss it with someone else (better don't), you should say this in the preamble of the assignment. You should only submit work that is completely your own and you should be able to explain your solutions if asked to do so.
Make sure you handle **invalid input, including type mismatch** - input validation is a requirement! Focus on **good design**, **good style**, and, most importantly, **reuse of code**. Start early and remember the rules about late assignments/syllabus → **no late assignments!**

**NOTE:** Try to use this quote as a guide when writing code (I do!):

*"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."*
--- Martin Fowler, Refactoring: Improving the Design of Existing Code

**HONORS STATEMENT:** "I will not cheat, fabricate materials, plagiarize, or help another to undertake such acts of academic dishonesty, nor will I protect those who engage in acts of academic dishonesty. I pledge on my honor as a student, that I have not received or provided any unauthorized help on this assignment."

---

**1.** Design a Java program to implement matrix arithmetic for square matrices (same number of rows and columns). You will need to use your math textbook or the Internet to review operations with matrices.
Make sure that your program is calling methods to perform (at least) the following operations:

**Generate**: Generate a matrix with random values 1 - 10
**Addition:** See http://mathworld.wolfram.com/MatrixAddition.html
**Subtraction:** Figure it out!
**Multiplication**
-- Multiply two matrices: See http://mathworld.wolfram.com/MatrixMultiplication.html
-- Multiply a matrix by a scalar: Equivalent to multiplication of each element by that scalar.
**Transposition:** See http://mathworld.wolfram.com/Transpose.html
**Matrix Trace:** See http://mathworld.wolfram.com/MatrixTrace.html
**Print (**use printf**)**

Have another method called **menu()**, that should list the options for matrix operations for which you designed the previous functions, and an option to exit the program. Your program should loop until the user chooses to exit. In this loop you are required to use a **switch** statement for all possible cases. Please use as a guide the sample output provided below.

**SAMPLE OUTPUT:**

```
Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 2

Enter the size of square matrices: 4
First matrix is:


   3   6   7   4
  10  10   2   5
  10   5   7   6
   3   8   7   2
Second matrix is:
   9   7   6   4
   7   5   3   4
   2   9   6   7
   2   2  10   9
The resulting matrix is:
  -6  -1   1   0
   3   5  -1   1
   8  -4   1  -1
   1   6  -3  -7
                        Command number 1 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 1

Enter the size of square matrices: 3
First matrix is:
   6   7   6
   2   1   6
   7   3   3
Second matrix is:
   3  10   7
   4   3   3
   9   9   2
The resulting matrix is:
   9  17  13
   6   4   9
  16  12   5
                        Command number 2 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
```

```
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 4

Enter the size of the square matrix: 5
Enter the multiplication constant: 4
The matrix is:
   3   9   2   5  10
  10   9   2   4   5
   5   3   1   2   7
   3   1   8  10  10
   1   9   9   8   7
The original matrix multiplied by 4 is:


  12  36   8  20  40
  40  36   8  16  20
  20  12   4   8  28
  12   4  32  40  40
   4  36  36  32  28
                        Command number 3 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 3

Enter the size of square matrices: 3
First matrix is:
   6   9  10
   4   7   8
  10   8   7
Second matrix is:
   4  10   6
   3   8   4
   8   2   5
The resulting matrix is:
 131 152 122
 101 112  92
 120 178 127
                        Command number 4 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 5
```

```
Enter the size of the square matrix: 7
The matrix is:
   4   9   4   3   7   2   6
   2   2  10   6   3   9   9
   7   4   2   8   9   9   2
   9   1   7   5  10  10   5
   8   6   8   7   7   1   8
   3   2   9   8  10   6   2
   6  10   6  10   6   6   5
The transposed matrix is:
   4   2   7   9   8   3   6
   9   2   4   1   6   2  10
   4  10   2   7   8   9   6
   3   6   8   5   7   8  10
   7   3   9  10   7  10   6
   2   9   9  10   1   6   6
   6   9   2   5   8   2   5
                        Command number 5 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
       3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 6

Enter the size of the square matrix: 5
The matrix is:
   2   2   5   9   4
   7   3   1   5   4
   8   9  10   4   4
   4   6  10   4   8
   6  10  10   9   8
The trace for this matrix is: 27
                        Command number 6 completed.

Your options are:
-----------------
        1) Add 2 matrices
        2) Subtract 2 matrices
        3) Multiply 2 matrices
        4) Multiply matrix by a constant
        5) Transpose matrix
        6) Matrix trace
        0) EXIT
Please enter your option: 0
Testing completed.
```

---

**2.** Write a Java program to solve the following problem using modularity:

- Write a method that rotates a one-dimensional array with one position to the left in a circular way, so that the first element becomes the last element (shift left). Call this method `shiftLeft`. For example, if the array is `17 34 22 55 37 10`, then after rotation the array is `34 22 55 37 10 17`
- Write a method that rotates a one-dimensional array with one position to the right in a circular way, so that the last element becomes the first element (shift right). Call this method `shiftRight`. For example, if the array is `17 34 22 55 37 10`, then after rotation the array is `10 17 34 22 55 37`

- Overload the method `shiftLeft` so that you rotate the array left by `k` positions. For example, if the array is `17 34 22 55 37 10` and `k = 2`, then after rotation the array is `22 55 37 10 17 34`
- Overload the method `shiftRight` so that you rotate the array right by `k` positions. For example, if the array is `17 34 22 55 37 10` and `k = 2`, then after rotation the array is `37 10 17 34 22 55`
- Write a method `shiftLeftK` so that you rotate the array left by `k` positions. The method should preserve the original array and save the results of the left rotation into a second array. Method signature: `public static int[] shiftLeftK(int[]list, int s, int k)`
- Write a method `shiftRightK` so that you rotate the array right by `k` positions. The method should preserve the original array and save the results of the right rotation into a second array. Method signature: `public static int[] shiftRightK(int[]list, int s, int k)`

Your program will invoke methods to initialize the array (random values between 1 and 100) print the array before the rotation, rotate the array, and then print the array (or the resulting array) after rotation. Use dynamic arrays and ask the user for the array size. Like in the previous program, you will use a method called **`menu()`**, that should print a list of options , including the option to exit the program. Your program should loop until the user chooses to exit. In this loop you are required to use a **`switch`** statement for all possible cases. Please use as a guide the sample output provided below:

**SAMPLE OUTPUT:**

```
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: fasdfdsfds
WRONG TYPE! Not an integer! REENTER: 12
INVALID! Out of range! REENTER: 3

How many elements in the list: 15
The original list is:
   51   48   13   31   99   91   21   92   53   32   47   31    7   22   42
Input k (step for shift left): 3
The second list after left rotation with 3 positions is:
   31   99   91   21   92   53   32   47   31    7   22   42   51   48   13

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 4.4
WRONG TYPE! Not an integer! REENTER: 6

How many elements in the list: 10
The original list is:
   14   71   20    9   76   54   34   45   49   13
Input k (step for shift right): 2
```

```
The second list after right rotation with 2 positions is:
   49   13   14   71   20    9   76   54   34   45

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 1

How many elements in the list: 17
The original list is:
   55   10   45   79   71   73   68   65   61   81   66   81    9   25   76   23    6
The list after left rotation is:
   10   45   79   71   73   68   65   61   81   66   81    9   25   76   23    6   55

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 5

How many elements in the list: dfrere
Not an integer! Try again!
How many elements in the list: 13
The original list is:
   95   73   19   45   27   54   79   28   77   80   42    1    4
Input k (step for shift right): 4
The original list after right rotation with 4 positions is:
   80   42    1    4   95   73   19   45   27   54   79   28   77

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 2

How many elements in the list: 9
The original list is:
   89   75   73   50   77   73   32   93   46
Input k (step for shift left): 3
The original list after left rotation with 3 positions is:
   50   77   73   32   93   46   89   75   73

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
```

```
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 4

How many elements in the list: 16
The original list is:
   63   37   74   23   66   85   62   49   27   39   91   34   60   27   92   26
The list after right rotation is:
   26   63   37   74   23   66   85   62   49   27   39   91   34   60   27   92

Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 0
Testing complete. Exit program.
```

---

**3.** Prove that the methods you wrote in the previous program are reusable. Reuse existing code! Write a second version of the previous program and read numbers from an input file. Get the file name from the user, handle the `FileNotFoundException` exception (`try/catch`) and consume unwanted input. Do multiple testing and handle all possible errors. Please use as a guide the sample output provided below:

**SAMPLE OUTPUT #1:**

```
Please input the name of the file to be opened: inputBad.txt

--- File NOT Found! ---
```

**SAMPLE OUTPUT #2:**

```
Please input the name of the file to be opened: badInput.txt

Found the file but it has no integers!
Testing complete. Exit program.
```

**SAMPLE OUTPUT #3:**

```
Please input the name of the file to be opened: input.txt

Found 16 integers in file. Start processing.
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 6
The original list is:
   23   45   67   88  231   56   78   99  145   67   33   44   55   66   77   11
Input k (step for shift right): 3
```

```
The second list after right rotation with 3 positions is:
   66   77   11   23   45   67   88  231   56   78   99  145   67   33   44   55
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 2
The original list is:
   23   45   67   88  231   56   78   99  145   67   33   44   55   66   77   11
Input k (step for shift left): 5
The original list after left rotation with 5 positions is:
   56   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: gfdsgfd
WRONG TYPE! Not an integer! REENTER: 1
The original list is:
   56   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231
The list after left rotation is:
   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231   56
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 3
The original list is:
   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231   56
Input k (step for shift left): 7.77
WRONG TYPE! Not an integer! REENTER: g
WRONG TYPE! Not an integer! REENTER: 4
The second list after left rotation with 4 positions is:
   33   44   55   66   77   11   23   45   67   88  231   56   78   99  145   67
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 4
The original list is:
   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231   56
The list after right rotation is:
   56   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231
Your options are:
```

```
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 5
The original list is:
   56   78   99  145   67   33   44   55   66   77   11   23   45   67   88  231
Input k (step for shift right): 2
The original list after right rotation with 2 positions is:
   88  231   56   78   99  145   67   33   44   55   66   77   11   23   45   67
Your options are:
-----------------
      1) Shift left
      2) Shift left (k steps)/in place
      3) Shift left (k steps)/second array
      4) Shift right
      5) Shift right (k steps)/in place
      6) Shift right (k steps)/second array
      0) EXIT
Please enter your option: 0
Testing complete. Exit program.
```

---

**NOTE:** The following files were used for testing. You shouldn't print the data/files.

**input.txt** - 23 45 67 88 ff hhh 231 nnn 56 s 78 99 145 67 33 44 bb 55 66 77 11
**badInput.txt** - sds asdfdsaf asfdasf qewq e ff 23.56 yyy rewrewrewrqwer 222.45

---