PROGRAMMING ASSIGNMENT #4
CS 2223 D-TERM 2023
GAUSS-JORDAN ELIMINATION
&
DYNAMIC PROGRAMMING

ONE HUNDRED POINTS
DUE: THURSDAY, APRIL 13, 2023 11 PM

1. (5 Points) Explain why the (repaired[1]) *ForwardElimination* algorithm on page 210 of Levitin fails to provide a solution for:

$$\begin{array}{ccccccc} x_1 & + & x_2 & + & x_3 & = & 6 \\ x_1 & + & x_2 & + & 2x_3 & = & 9 \\ x_1 & + & 2x_2 & + & 3x_3 & = & 14 \end{array}$$

   despite the fact that $x = (1, 2, 3)$ or $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ can be easily verified as a solution to the system.

   How does the *BetterForwardElimination* algorithm on page 211 of Levitin remedy this?

2. (10 Points) Explain in some detail why the *BetterForwardElimination* algorithm on page 211 of Levitin fails to provide a solution for:

$$\begin{array}{ccccccc} x_1 & + & x_2 & + & x_3 & = & 6 \\ x_1 & + & x_2 & + & 2x_3 & = & 9 \\ 2x_1 & + & 2x_2 & + & 3x_3 & = & 15 \end{array}$$

   despite the fact that $x = (1, 2, 3)$ or $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ can be easily verified as a solution to the system.

   What can be done to remedy this shortcoming in the algorithm?

[1] The algorithm as printed has a serious bug. We'll remedy this in class and in a separate video. The corrected algorithm still has a significant shortcoming.

3. (35 Points) The **Gauss-Jordan elimination** method differs from Gaussian elimination in that the elements above the main diagonal of the coefficient matrix are made zero at the same time and by the same use of a pivot row as the elements below the main diagonal. Thus, the coefficient matrix is transformed into a diagonal matrix rather than an upper-triangular matrix. Furthermore, if each pivot row is "divided by" its pivot (leading non-zero entry) prior to its use as a pivot row, the coefficient matrix is transformed into the identity matrix, and the back substitution step may be dispensed with entirely. That is, the solution x is simply the last column of the (transformed) augmented system matrix.

Modify the *BetterForwardElimination* algorithm to perform Gauss-Jordan elimination to solve a system of n linear equations in n unknowns with the form $Ax = b$, where A is an $n \times n$ matrix of real coefficients[2], and b is a column vector with n real entries.

Implement your algorithm in Java to find the unique solution to the system:

$$
\begin{array}{rcrcrcrcrcrcrcrcrcrcrcr}
x_0 & + & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & + & x_8 & + & x_9 & + & x_{10} & = & 2047 \\
x_0 & + & x_1 & & & & & & & & & & & & & & & & & & & = & 3 \\
& & & & x_2 & + & x_3 & & & & & & & & & & & & & & & = & 12 \\
& & & & & & & & x_4 & + & x_5 & & & & & & & & & & & = & 48 \\
& & & & & & & & & & & & & & x_7 & + & x_8 & & & & & = & 384 \\
& & & & & & & & & & & & & & & & & & x_9 & + & x_{10} & = & 1536 \\
x_0 & & & + & x_2 & & & & & & & & & & & & & & & & & = & 5 \\
& & x_1 & & & & & + & x_4 & + & x_5 & & & & & & & & & & & = & 50 \\
& & & & & & & & & & x_5 & & & + & x_7 & + & x_8 & + & x_9 & + & x_{10} & = & 1952 \\
11x_0 & + & 10x_1 & + & 9x_2 & + & 8x_3 & + & 7x_4 & + & 6x_5 & + & 5x_6 & + & 4x_7 & + & 3x_8 & + & 2x_9 & + & x_{10} & = & 4083 \\
11x_0 & - & 10x_1 & + & 9x_2 & - & 8x_3 & + & 7x_4 & - & 6x_5 & + & 5x_6 & - & 4x_7 & + & 3x_8 & - & 2x_9 & + & x_{10} & = & 459
\end{array}
$$

[2]Note: We may revisit this algorithm later, but for now you can assume that an 11×12 array of floats will suffice.

4. (50 Points) The Dwarf King has locked the Heart of the Mountain, the jewel known as the Arkenstone, in one of the vaults in the deepest treasure room under the Lonely Mountain. The square floor of the room consists of 64 smaller squares, alternating gold and silver. (It is believed that ancient Persians, having discovered the chamber long after the age of Dwarves, were inspired by its beauty to create games played on such a surface.) Upon each square the King has arrayed a varying number of the most wondrous gemstones: emeralds, rubies, sapphires, diamonds, and more, as shown here:

	Vault 1	Vault 2	Vault 3	Vault 4	Vault 5	Vault 6	Vault 7	Vault 8
Row 8	21	95	20	82	66	52	89	35
Row 7	74	40	37	78	23	14	5	79
Row 6	63	16	4	31	25	17	59	32
Row 5	15	92	71	13	48	77	11	91
Row 4	12	67	88	22	64	47	70	56
Row 3	7	30	51	65	27	94	97	83
Row 2	93	53	24	46	86	1	41	10
Row 1	84	99	68	75	98	44	33	96

The vault containing the Arkenstone is sealed with powerful magic which can only be broken by someone who has walked the Most Precious Path. Bilbo Baggins, the Hobbit burglar, having once possessed the Arkenstone, wishes to behold it once again. Devise and implement in Java a dynamic programming algorithm which will allow Mr. Baggins to determine the Most Precious Path and thereby collect the greatest number of gemstones given that he:

- begins by collecting the gems on the square of his choice in Row 1, and then advances to the next Row by moving to
  - the square directly ahead of the one he currently occupies, or
  - the square (diagonally) ahead and to the left of the one he currently occupies, provided that he is not already against the left wall of the treasure room, or
  - the square (diagonally) ahead and to the right of the one he currently occupies, provided that he is not already against the right wall of the treasure room;
- collects the gems from the newly-visited square, and
- repeats this process until,
- he collects gems from a square on Row 8, whereupon the spell sealing the corresponding door will be broken and the vault will yield its treasure if and only if Bilbo has walked the Most Precious Path.

Your output should include:

(a) Bilbo's starting square,
(b) a representation of his path,
(c) the total number of gems collected on the way, and
(d) the number of the vault wherein the King has secreted the Arkenstone.

	Vault 1	Vault 2	Vault 3	Vault 4	Vault 5	Vault 6	Vault 7	Vault 8
Row 8	21	95	20	82	66	52	89	35
Row 7	74	40	37	78	23	14	5	79
Row 6	63	16	4	31	25	17	59	32
Row 5	15	92	71	13	48	77	11	91
Row 4	12	67	88	22	64	47	70	56
Row 3	7	30	51	65	27	94	97	83
Row 2	93	53	24	46	86	1	41	10
Row 1	84	99	68	75	98	44	33	96

Hints to follow . . .

	Vault 1	Vault 2	Vault 3	Vault 4	Vault 5	Vault 6	Vault 7	Vault 8
Row 8	1000	900	800	0	$1M	0	600	1001
Row 7	1000	900	800	0	0	0	600	1001
Row 6	1000	900	800	0	0	0	600	1001
Row 5	1000	900	800	0	0	0	600	1001
Row 4	1000	900	800	0	0	0	600	1001
Row 3	1000	900	800	0	0	0	600	1001
Row 2	1000	900	800	0	0	0	600	1001
Row 1	1000	900	800	0	0	0	600	1001

	Vault 1	Vault 2	Vault 3	Vault 4	Vault 5	Vault 6	Vault 7	Vault 8
Row 8	8000 1000	7900 900	7700 800	6700 0	1,005,700 $1M ARKENSTONE!	6606 0	7607 600	8008 1001
Row 7	7000 1000	6900 900	6700 800	5700 0 Path	4700 0	5605 0	6606 600	7007 1001
Row 6	6000 1000	5900 900	5700 800 Precious	4700 0	3700 0	4604 0	5605 600	6006 1001
Row 5	5000 1000	4900 900 Most	4700 800	3700 0	2700 0	3603 0	4604 600	5005 1001
Row 4	4000 1000 the	3900 900	3700 800	2700 0	1700 0	2602 0	3603 600	4004 1001
Row 3	3000 1000 walk	2900 900	2700 800	1700 0	800 0	1601 0	2602 600	3003 1001
Row 2	2000 1000 and	1900 900	1700 800	800 0	0 0	600 0	1601 600	2002 1001
Row 1	START 1000 HERE	900	800	0	0	0	600	1001

Note: The Most Precious Path might require Bilbo to start at any of the squares on the first row. In fact, for our problem, he will NOT start in the first column—it will be some other starting square.

This example shows how the Most Precious Path cannot (necessarily) be found with a greedy algorithm. Here the largest item in each row is ignored (and the largest sum) until the very end. You'll need a dynamic programming algorithm to find the Most Precious Path…