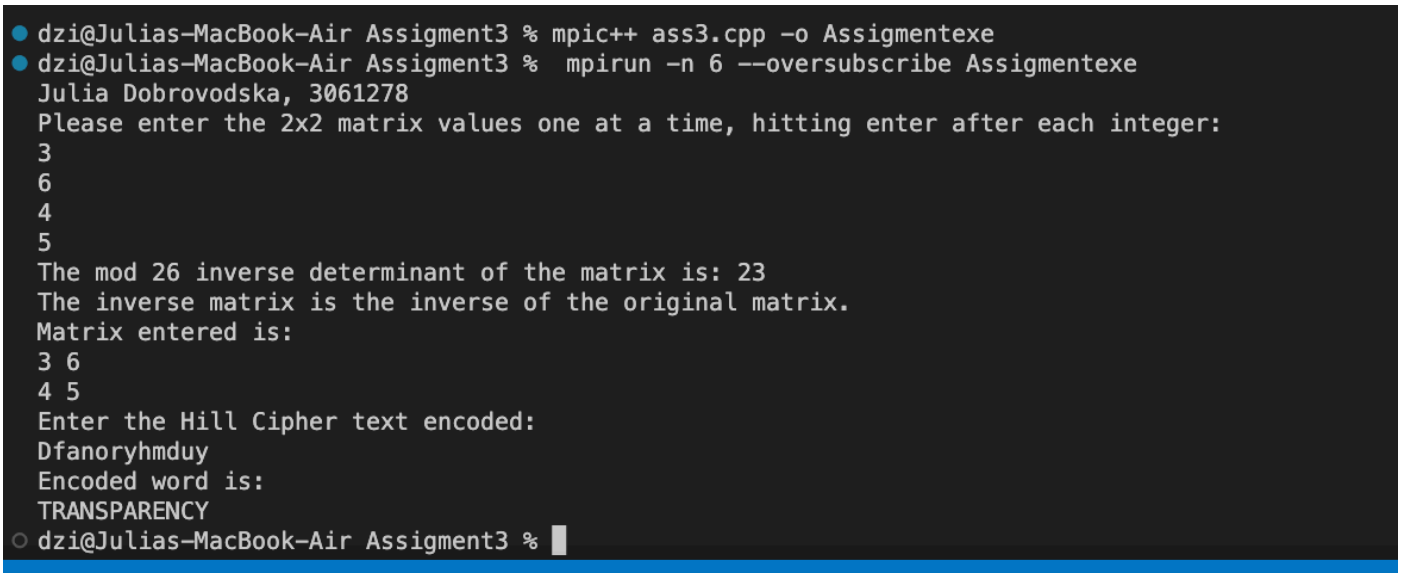
Julia Dobrovodska – 3061278
Documentation Report
Assignment 3: Parallel Hill Cipher Decryption

Answer the questions below, relating your answers to the program you submitted for Part 1.

1. Provide a screenshot of the command used to run your code for this assignment, including all of the output that is generated when you run this script.



2. In your own words, explain how the Hill Cipher works.

Explanation:

The Hill Cipher is a encryption/decryption technique that uses matrix operations to convert plaintext into ciphertext and vice versa. A key matrix is selected, typically 2x2 or 3x3 in size, and multiplied by a block of plaintext characters. The resulting matrix is then scaled down and converted into ciphertext. Decryption requires knowledge of the key matrix, and involves multiplying the ciphertext matrix by the inverse of the key matrix, scaling the result modulo the alphabet's length, and then translating the resulting integers back into plaintext letters. The Hill Cipher is resistant to brute-force attacks due to the large number of possible matrix keys, but can be vulnerable if the matrix size is too small or if the same key is used repeatedly.

Reference to Hill Cipher decryption `https://www.math.stonybrook.edu/~scott/papers/MSTP/crypto/8Hill_Cipher.html#:~:text=The Hill cipher works by,a paper published in 1931.`

3. For each helper method you implemented for the assignment, briefly explain how the method works. Include in your explanation any inputs or outputs the method uses.

Explanation:

- **printArray(T* data, int size)** - this method take 2 parameters array(of any datatype) and size of an array. Method for print an array to the console.
- **print2DArray(int array[2][2], int numRows, int numCols)** - this method take 3 parameters 2D int array, row size and column size). Method for print 2D array to the console.
- **decodeBlock(int block[2], int inverseMatrix[2][2])** - take 2 parameters array of integers, and 2D array of integers. This method decode block of two integers by an inverse matrix, mod and return decoded block ints.
- **isInverse(int a[2][2], int b[2][2])** - takes 2 parameters two 2D arrays and return boolean whether the compare matrix is inverse or not.
- checkValue(int n) - takes one parameter and return a integer which is in range 0-25 if its smaller than 0 we keep adding 26 and if its bigger we do modulo 26.
- **calculateInvDeterminant(int m[2][2])** - takes one parameter which is 2D array and return integer the determinant of a 2x2 matrix
- **calculateInverse(int matrix[2][2], int invDet, int inverse[2][2])** - takes 3 parameters two 2D arrays and determinant it calculate the modulo 26 inverse of a 2x2 matrix.
- **decode(int* ciphertextNumbers, int ciphertextSize)** - this method take 2 parameters int array and size of an array. It decrypt the numbers to letters return char array.

4. In part a), a matrix is taken from the user. Explain how this is read into your program and how the user provides the matrix.

Explanation:

The number is taken one integer at the time after each integer (number) entered the user have to press `enter` key. I think the message for the user is clear :

"Please enter the 2x2 matrix values one at a time, hitting enter after each integer: "

Note it is not catered for inputing wrong characters. Assumption - user always input numbers (- and + is acceptable)

5. What is the mod 26 inverse of matrix A? Explain how this is calculated programatically in your program.

Explanation:

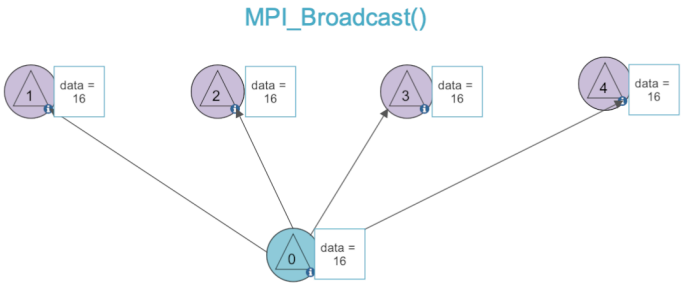
```
The mod 26 inverse determinant of the matrix is: 23
And the inverse matrix is :
11 18
12 17
```

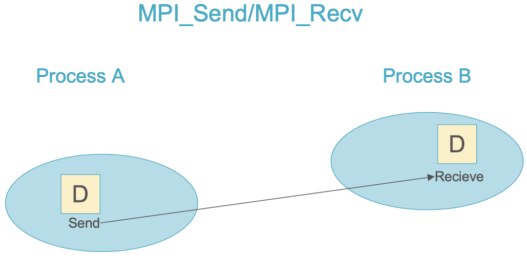
We calculate the inverse of a 2x2 matrix using modular arithmetic. The method takes the original matrix and the inverse determinant as input, than we calculate the elements of the inverse matrix using a formula for a 2x2 matrix inverse. To ensure all elements in the inverse matrix are positive, it adds 26 to any negative elements. The resulting inverse matrix is stored in the inverse 2D array.

6. Explain the difference between MPI_Send/MPI_Recv and MPI_Broadcast. Include in your explanation a diagram for each.

Explanation + Diagrams:

It is important to know that MPI_Bcast() is not a simple wrapper around MPI_Send() and MPI_Recv() function. If we look at the diagrams we can see that MPI_Send and MPI_Recv are used for point-to-point communication between two specific processes, when one process have one variable and only one other processor need to access it. While MPI_Broadcast is used for one-to-many communication between one process and all other processes in the program.





7. What is the decoded message? Define this term and discuss two potential benefits and two challenges of implementing it in a Distributed System.

Explanation:

Encoded word is:
TRANSPARENCY

Transparency - `Transparency is the quality of being easily seen through `- also similar in a Distributed System it hide the differences or its underlying complexity from users and applications. Advantages of transparency would be -greater flexibility and easier use. The implementation of transparency could face two difficulties: a potentially complex system, and performance issues.

8. Explain how the ciphertext is read into your program.

Explanation:

Ciphertext is read in by using `std::cin >> ciphertext` . The `std::cin` function is part of the standard C++ library and is used to read input from the console or standard input stream. The user can then enter the ciphertext on the console, followed by a press enter key on keyboard. Than the input will be read and stored in the ciphertext variable.

Note it is not catered for inputing wrong characters. Assumption - user always correct text for encryption.

9. Will the output produced in your program always be the same, given the same ciphertext and matrix? Explain why or why not.

Explanation:

I think it is possibility that the output will stay same every time you run the program given the same `ciphertext` and `matrix`. However, this might be a coincidence and not something that we can guarantee. The reason for this is that we are is using MPI, which is designed for parallel computing. MPI can split the computation among multiple processors, and the order in which these processors execute the computations can be different depending on some factors, such as the load on the processors, the availability of resources… It is similar if you have 1 people doing the job vs 4 people doing the job, one might fail or take a break and therefore he might finish later than the others. Therefore we can’t expect the same output all the time.

10. If any part of your submission doesn't work, please detail this below including relevant screenshots.

Explanation + Screenshots:
Everything working as expected. I tried also with the ciphertext from the lab10 given and also seem to work as expected, running on 6 nodes. Improvement could be done for checks of user input if the decrypted text is correct or key is correct. Maybe store all encrypted text in array and randomly give user a word to decrypt.