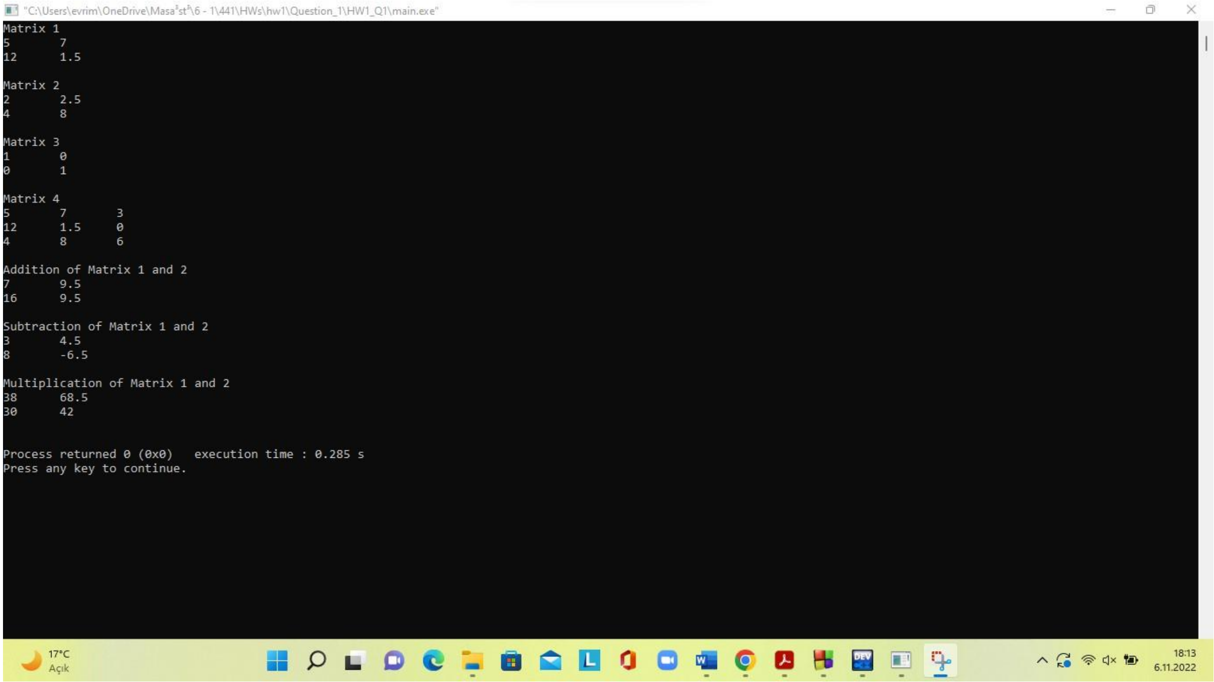
Ramazan Cem Çıtak 2231686 Submission Date: November 6, 2022

EE441 Programming Assignment 1

PART 1

1) In this part I constructed `Getter` and `Setter` functions as well as a constructor that initializes a matrix as an identity matrix.
2) I used pass by value in addition, subtraction and multiplication functions for matrices because it is more reliable than using a pointer especially compared to pass by address method.
3) I tried to use the cofactor method recursively in my `Determinant` function in my code, but I could not decrease the `N` value, or size, which is a `const int` value whenever I call the function for a recursive operation.

An example output for my program is given in Figure 1.


"C:\Users\evrim\OneDrive\Masa³st³\6 - 1\441\HWs\hw1\Question_1\HW1_Q1\main.exe"
17°C
Açık
18:13
6.11.2022

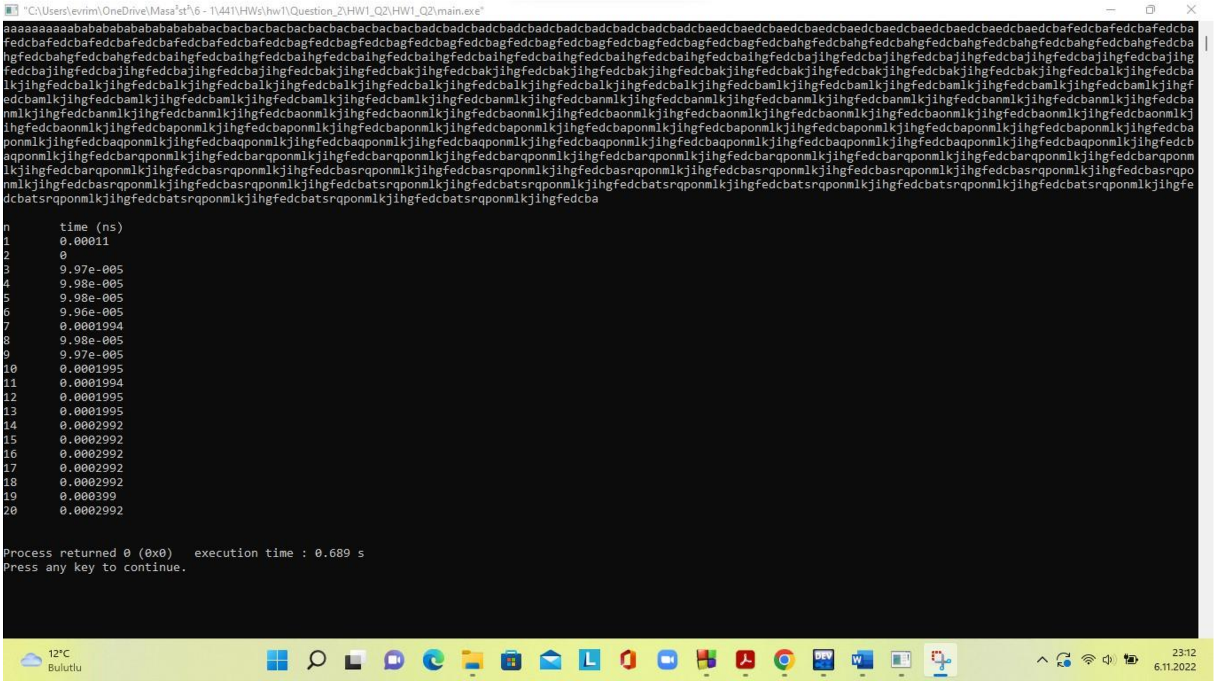
```
Matrix 1
5       7
12      1.5

Matrix 2
2       2.5
4       8

Matrix 3
1       0
0       1

Matrix 4
5       7       3
12      1.5     0
4       8       6

Addition of Matrix 1 and 2
7       9.5
16      9.5

Subtraction of Matrix 1 and 2
3       4.5
8       -6.5

Multiplication of Matrix 1 and 2
38      68.5
30      42


Process returned 0 (0x0)   execution time : 0.285 s
Press any key to continue.
```

Figure 1: An example output of my program in Part 1

PART 2

1)
   a) I implemented `Disc` class for varying diameters with a constructor that diameter 0 is reversed for no disc.
   b) I implemented class `Hanoi` with three integer arrays of size 20.
   c) I wrote a constructor to initialize the game with an input of the number of discs.
   d) I partially implemented this `move` function in my code.
   e) I could not write the recursive function `solve_hanoi`, but I found out that the worst case timing consideration, which is denoted by Big – O, for the Tower of Hanoi algorithm is equal to $O(2^n - 1) \cong O(2^n)$. This implies that every time we increase the number of discs by 1, the timing will be two times larger than the previous case in the algorithm.
2) I successfully implemented a recursive `print_backwards` function by using a helper function in order to separate the operation of string reversing and finding the size of the string.
3) I could not implement a `nth_prime` function, so I could not find the Big – O notation for this question.

4) The output of the benchmark program for the recursive `print_backwards` function is given in Figure 2. In the output we can see that the timing is not increasing exponentially, but it is linear with n, which is the number of the characters in the string. Therefore, the Big-O notation for this function is equal to $O(n)$.
Because my code for the question 1 and 3 did not work properly, I could not put outputs from the benchmark program from those questions.


"C:\Users\evrim\OneDrive\Masa³st³\6 - 1\441\HWs\hw1\Question_2\HW1_Q2\HW1_Q2\main.exe"
12°C
Bulutlu
23:12
6.11.2022

```
aaaaaaaaaababababababababababacbacbacbacbacbacbacbacbacbacbadcbadcbadcbadcbadcbadcbadcbadcbadcbadcbaedcbaedcbaedcbaedcbaedcbaedcbaedcbaedcbaedcbaedcbafedcbafedcbafedcba
fedcbafedcbafedcbafedcbafedcbafedcbafedcbagfedcbagfedcbagfedcbagfedcbagfedcbagfedcbagfedcbagfedcbagfedcbagfedcbahgfedcbahgfedcbahgfedcbahgfedcbahgfedcbahgfedcbahgfedcba
hgfedcbahgfedcbahgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbaihgfedcbajihgfedcbajihgfedcbajihgfedcbajihgfedcbajihgfedcbajihg
fedcbajihgfedcbajihgfedcbajihgfedcbajihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbakjihgfedcbalkjihgfedcba
lkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbalkjihgfedcbamlkjihgfedcbamlkjihgfedcbamlkjihgfedcbamlkjihgfedcbamlkjihgf
edcbamlkjihgfedcbamlkjihgfedcbamlkjihgfedcbamlkjihgfedcbamlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcba
nmlkjihgfedcbanmlkjihgfedcbanmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkjihgfedcbaonmlkj
ihgfedcbaonmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcbaponmlkjihgfedcba
ponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcbaqponmlkjihgfedcb
aqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponmlkjihgfedcbarqponm
lkjihgfedcbarqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbasrqpo
nmlkjihgfedcbasrqponmlkjihgfedcbasrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfe
dcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcbatsrqponmlkjihgfedcba

n       time (ns)
1       0.00011
2       0
3       9.97e-005
4       9.98e-005
5       9.98e-005
6       9.96e-005
7       0.0001994
8       9.98e-005
9       9.97e-005
10      0.0001995
11      0.0001994
12      0.0001995
13      0.0001995
14      0.0002992
15      0.0002992
16      0.0002992
17      0.0002992
18      0.0002992
19      0.000399
20      0.0002992


Process returned 0 (0x0)   execution time : 0.689 s
Press any key to continue.
```

Figure 2: The output of the benchmark program for the recursive `print_backwards` function