Alper Kutay ÖZBEK

Part 1

1)

A) I show the results of two different matrixes which are 3 and 7 dimensions.

```
N=3 :
1 0 0
0 1 0
0 0 1
N=7 :
1 0 0 0 0 0 0
0 1 0 0 0 0 0
0 0 1 0 0 0 0
0 0 0 1 0 0 0
0 0 0 0 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
```

Figure 1 : Part_1_a

B) I set the two indexes in both matrices ; however, if indexes of matrixes is written wrong, function prints that is below.

```
1 0 8
0 1 0
7 0 1
Please enter valid values
1 0 0 0 0 0 0
0 1 5 0 0 0 0
0 5 1 0 0 0 0
0 0 0 1 0 0 0
0 0 0 0 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
```

Figure 2 : Part_1_b

2)I implemented addition, subtraction and multiplication functions that take two matrices and return a new matrix with the result which is below.

```
MATRIX_0=>
1 0 8
0 1 0
7 0 1
MATRIX_1=>
1 0 7
0 1 5
0 5 1
MATRIX_0+MATRIX_1=>
2 0 15
0 2 5
7 5 2
MATRIX_2*MATRIX_1=>
2 75 29
0 27 15
7 15 76
MATRIX_3-MATRIX_1=>
1 75 22
0 26 10
7 10 75
```

Figure 3 : Part_2_a

3) I implemented a determinant function that takes one matrix and returns its determinant.

```
MATRIX_0=>
1 0 8
0 1 0
7 0 1
MATRIX_1=>
1 0 7
0 1 5
0 5 1
MATRIX_0+MATRIX_1=MATRIX_2
2 0 15
0 2 5
7 5 2
MATRIX_2*MATRIX_1=MATRIX_3
2 75 29
0 27 15
7 15 76
MATRIX_3-MATRIX_1=MATRIX_4
1 75 22
0 26 10
7 10 75

Determinant of MATRIX_0 is  -55
Determinant of MATRIX_1 is  -24
Determinant of MATRIX_2 is  -252
Determinant of MATRIX_3 is  6048
Determinant of MATRIX_4 is  3096
```

Figure 4: Part_3

Part 2)

Q1) Zeros are reserved for no discs and numbers represents diameter of the discs. The figure below shows that initilized Hanoi for N=5. Discs begin in the rod_0 and they are in order.

```
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
1 0 0
2 0 0
3 0 0
4 0 0
5 0 0
```

Figure 5: Part_1 Q2

Move_to function has several if-else instructions. Each instruction represents different condition. Before the switch operation destion index of the road increase and after that source value is replaced with 0 and index decreases.

```
Disc 1 is moved from Rod 0 to Rod 2
Disc 2 is moved from Rod 0 to Rod 1
Disc 1 is moved from Rod 2 to Rod 1
Disc 3 is moved from Rod 0 to Rod 2
Disc 1 is moved from Rod 1 to Rod 0
Disc 2 is moved from Rod 1 to Rod 2
Disc 1 is moved from Rod 0 to Rod 2
Disc 4 is moved from Rod 0 to Rod 1
Disc 1 is moved from Rod 2 to Rod 1
Disc 2 is moved from Rod 2 to Rod 0
Disc 1 is moved from Rod 1 to Rod 0
Disc 3 is moved from Rod 2 to Rod 1
Disc 1 is moved from Rod 0 to Rod 2
Disc 2 is moved from Rod 0 to Rod 1
Disc 1 is moved from Rod 2 to Rod 1
Disc 5 is moved from Rod 0 to Rod 2
Disc 1 is moved from Rod 1 to Rod 0
Disc 2 is moved from Rod 1 to Rod 2
Disc 1 is moved from Rod 0 to Rod 2
Disc 3 is moved from Rod 1 to Rod 0
Disc 1 is moved from Rod 2 to Rod 1
Disc 2 is moved from Rod 2 to Rod 0
Disc 1 is moved from Rod 1 to Rod 0
Disc 4 is moved from Rod 1 to Rod 2
Disc 1 is moved from Rod 0 to Rod 2
Disc 2 is moved from Rod 0 to Rod 1
Disc 1 is moved from Rod 2 to Rod 1
Disc 3 is moved from Rod 0 to Rod 2
Disc 1 is moved from Rod 1 to Rod 0
Disc 2 is moved from Rod 1 to Rod 2
Disc 1 is moved from Rod 0 to Rod 2
```

Figure 6 : Moves of Solving Hanoi with 5 discs

When we look at the code of solve_hanoi, we can see that there are two recursive function. When each function goes to another recursive function so that it keeps rising with 2^n. If we show that with moves, figure 7 can help. Each number doubles with prior one. Therefore we can assume we are correct.

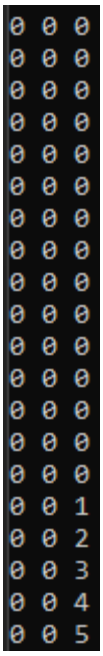

Figure 7 : Solving Hanoi

In this function str pointer increases until null-termination shows up after recursion ends , function prints backwards.

Figure 7: Print Backwards Function output

Q3) If we look at the code , there is no recursion so that we can calculate the complexity by examining the loops. There are two different “while” loops an done of them consists of the other one which means complexity is O(n). The reason is time complexity of program related with outer “while” loop of the function and it is increasing with value of n

I aim to find Nth prime number by finding each one of them until N comes

n	time (ns)
1	0.011514
2	0.0131598
3	0.0175676
4	0.028373
5	0.0344219
6	0.0590176
7	0.108072
8	0.203647
9	0.310208
10	0.473755
11	0.948082
12	1.83893
13	3.7028
14	7.43973
15	14.9383
16	29.5392
17	58.9784
18	120.586
19	236.287
20	476.045

n	time (ns)
1	3.628e-05
2	7.303e-05
3	0.00010629
4	0.00014149
5	0.0003363
6	0.00021147
7	0.00024672
8	0.00028389
9	0.00031764
10	0.00052717
11	0.00044432
12	0.00042676
13	0.00052313
14	0.00054516
15	0.00053291
16	0.00072524
17	0.0006024
18	0.00069434
19	0.0007088
20	0.00071193

n	time (ns)
1	7.178e-05
2	7.173e-05
3	7.196e-05
4	7.182e-05
5	0.00010676
6	0.0001067
7	0.00010772
8	0.00010722
9	0.00010731
10	0.00010811
11	0.00010774
12	0.00010853
13	0.00024162
14	0.00010908
15	0.00010928
16	0.00011055
17	0.00011191
18	0.00011128
19	0.00011249
20	0.00011323

Figrue 9 : Benchmark Results

n	time (ns)
1	1.6e-07
2	1.6e-07
3	2.1e-07
4	3.1e-07
5	5e-07
6	8.6e-07
7	1.5e-06
8	2.72e-06
9	5.22e-06
10	1.017e-05
11	2.038e-05
12	4.007e-05
13	7.981e-05
14	0.00016011
15	0.00032055
16	0.00082003
17	0.00129329
18	0.00254799
19	0.00506261
20	0.0101935

Figure 10 : Benchmark result of Hanoi without printing move function

If we look at the relation between time intervals we can see that solve_hanoi function keeps rising with double. So that we can say that we are true and complexity is O(2^n). In addition when we look at the second and thirds result in the figure 9, It can be seen that time sequance changes with linearly;therefore , complexity is O(n) which means we are correct.