EE441- Programming Assignment 1

Due Date: 30.10.2022, 23:55

For your questions: Utkucan Doğan – utkucan@metu.edu.tr

This assignment consists of two independent parts. You are going to create a separate CodeBlocks project for both parts. You are also going to prepare a pdf file with your answers and observations. Don't forget to write comments to your code as they are also graded.

Part 1 – Classes & Arrays [50 points]

In this part, you are going to implement an NxN matrix class. This class will hold a C-like array for matrix data and some useful functions. To generalize your implementation, you are going to use template feature of C++.

Templates are a method to generalize a given implementation with different types or constants. For this assignment, you are only required to have templated size N.

```
template <int N>
class Matrix
{
private:
    double data[N][N];

public:
    int const SIZE = N;
};
```

Figure 1: Templated class implementation

1) Implement this class with the following member functions:
    a. A constructor that initializes the matrix as an identity matrix
    b. A getter and a setter function to get and set an element for a given pair of (row,column). Important: Don't forget to check if the indices that entered are valid.
2) Implement addition, subtraction and multiplication functions that take two matrices and return a new matrix with the result.
   Important: State which argument passing method you use with your reasoning.
3) Implement a determinant function that takes one matrix and returns its determinant. Hint: You can use cofactor method and implement it recursively.

Part 2 – Recursion & Algorithmic Complexity [50 points]

In this part, you are going to write some functions, find their algorithmic complexities and check your result by benchmarking. This part should contain a main file that runs each benchmark.

1) Tower of Hanoi is a mathematical game that consists of three rods where you are required to transfer a stack of discs with varying sizes to the last rod. Discs can only be stacked by increasing diameter.

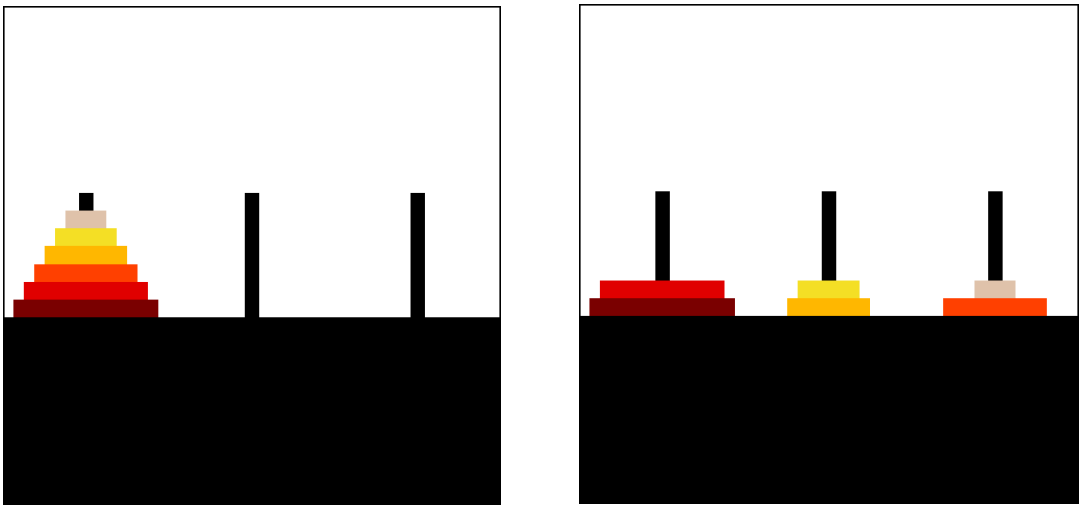

Figure 2: Tower of Hanoi start and mid-game

A general solution for this game is to transfer all tower but the bottom disc to the middle rod, transfer the bottom disc and retransfer the rest of the tower to the top of the bottom disc. This method can be modelled as a recursive algorithm.

a) Implement a class to model discs with varying diameters. Diameter 0 should be reserved for no disc.

b) Implement a class for game where you use three arrays with size 20 to model three rods.

```
class Hanoi
```

c) Write a constructor to initialize the game. Your constructor should take an integer to determine the number of discs to use.

d) Write a member function called "move" with following declaration:

```
void move(int from, int to);
```

This function should move a disc from one rod to another (Rods are identified by index 0, 1 and 2). Your function should check:

- If rod indices are valid
- If the move is legal

If move is successful, your function should print the move to the console like:

```
Disc 1 is moved from Rod 0 to Rod 2
```

e) Implement a function that takes a Hanoi object and solves it using the recursive method described above. State the complexity of your algorithm in big-o notation.

```
void solve_hanoi(Hanoi& game);
```

2) Implement a recursive function that takes a null-terminated string and prints it backwards.

```
void print_backwards(char const * string);
```

3) Implement a function that returns the nth prime number. State the complexity of your algorithm in big-O notation.
Hint: You can use a helper function for its implementation.

```
int nth_prime(int n);
```

4) To check if the complexities that you have found is true, use the benchmark code provided to you. You should add appropriate header file to the code for it to compile correctly. Document your time measurements for each function.