# EE441 - Programming Assignment 1

# Due Date: 7.11.2023, 23:55

## Part 1 – Classes & Arrays

1. This class holds number 4 ,and an operator ‘+’, and a member function to apply the given operation to a number. The input of the function is 5. Therefore, as a output we expect to see 9.

A computer code with black text

Description automatically generated with medium confidence

Figure : Main of the part 1 question 1

A screenshot of a computer

Description automatically generated

Figure : Output of the part 1 question 1

1. In this question, I impelement a class for operation queue. I put operation into the queue by using push\_back() function, then using pop\_front(), I applied the operation with input 2. I expected the output to be 6, 7 , 4 since it is a queue and first in operation will be applied firstly.

A screen shot of a computer code

Description automatically generated

Figure Main of the part 1 question 2

A screenshot of a computer

Description automatically generated

Figure

Output of the part 1 question 2



A screen shot of a computer code

Description automatically generated

Figure Code of the part 1 question 3

1. In order to calculate 3 + 9𝑥 + 5 this function, I manuplated the function as 3\*( and put the operation objects by starting the inside of manuplated function. In op3 object double casting performed due to avoid loss the information.

A computer code with text

Description automatically generated

Figure Main of the part 1 question 4

A screenshot of a computer

Description automatically generated

Figure Output of the part 1 question 4

## Part 2 – Recursion & Algorithmic Complexity

### 

In this question to model three rod I implemented a class. In this class the top information of nonzero values are holded and they are change when the push and pop functions are called. My class for rods is below.

A screenshot of a computer code

Description automatically generated

Figure 9 Class to model rods

In hanoi constructer, the rods are filled with zeros then the discs are put in the first rod. The figure below shows the hanoi game which is constructed with 5 discs.

For the move function, the indices are checked whether they are bigger than and equal to or smaller than 3. To check if the move is legal or illegal, I just check the condition that diameter of top disc of source rod smaller than the top disc in the destination rod. In the algoritm move disc from rod to same rod is legal since it does not change anything.

**A number on a black background

Description automatically generated**

Figure 10 Hanoi game constructed with 5 discs

A black screen with white numbers

Description automatically generated

Figure 11 After calling move(0,2) function

To solve hanoi with n discs recursively, the algorithm is constructed as follows:

1. Solve hanoi for first n-1 disc and move them from source rod to middle rod
2. Move nth disc from source to destination rod
3. Solve hanoi for first n-1 disc and move them from middle rod to destination rod

First the solve\_hanoi function takes hanoi object and gets the number of disc in the first rod dynamically. The function calls solve\_hanoi\_recursively, which is explained above.

The figure 12 shows the functions for solve hanoi and complexity calculations for recursive hanoi.

A screenshot of a computer program

Description automatically generated

Figure 13 Complexity calculation of recursive solve hanoi function

A screenshot of a computer

Description automatically generated

Figure 13 Output of solve hanoi function

2.

The algorith for print\_backwards as follows:

1. Run the function until see the null-terminated string. Our termination condition of recursive algorithm is (character [n] == ‘­\0’).
2. If char is not equal to null terminated string, call the function for string has last n-1 character.
3. After calling function, print the current character.

The figure 14 complexity of this algorithm.

A screenshot of a computer program

Description automatically generated

Figure 14 Complexity calculation

As a input “EE441 Data Structures” is given the function.



Figure 5 Output of print\_backwards

3. The complexity of GCD algorithm is O(log(a · b)) = O(log a + b) = O(log n)

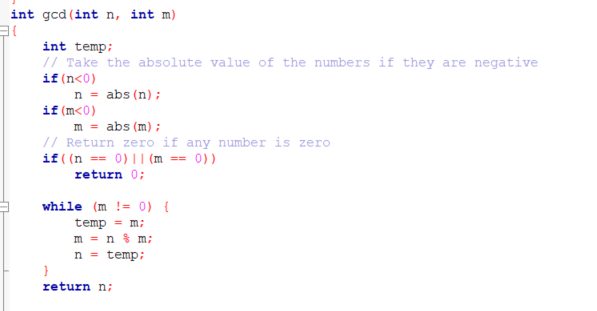


Figure 15 GCD Algorithm

4. The complexity of LCM algorithm is O(log n) since I use the gcd algorithm and complexities of other lines is constant.

A white background with text

Description automatically generated

Figure 15 GCD Algorithm

A computer code with text

Description automatically generated

A black background with white text

Description automatically generatedFigure 16 Main function implementation of gdc and lmc functions

Figure 17 Output of functions

A screenshot of a computer screen

Description automatically generated

Figure 18 Benchmark result of hanoi game

A graph with a line

Description automatically generated

Figure 19 Plot of comlexity of hanoi game

As we can see from Figure 19, complexity of hanoi is O(2^n).

A screenshot of a computer screen

Description automatically generated

Figure 20 Benchmark result of print\_backwards

A graph with a line going up

Description automatically generated

Figure 21 Plot of complexity of print\_backwards.

As we can see in figure 21, the complexity of print\_backwards is O(n).

A black and white screen with numbers

Description automatically generatedA screenshot of a computer screen

Description automatically generated

Figure 22 Benchmark results of gcd and lcm respectively

The complexities of gcd lcm algorithms are very small which is compatible with expectations.