# Tartalomjegyzék



# Exercises

## Dynamic Programming

### Deep Sum

Write a recursive function whose input is a list containing lists up to any depth containing positive integers. The function must return the sum of all numbers inside the list.

Example:

```
[1, 2, 3, [4, 5], [[[6], 7]]] -> 28
```

### Palindrome

Write a recursive function which checks whether a given string is a palindrome or not. Then try to write a non recursive function for that and compare the time complexity.

Example:

```
aba -> True
```

```
abb -> False
```

### Factorial

Write a recursive function to compute the factorial of a given natural number. Then try to write a non recursive function for that and compare their time complexity.

### Sum Digit

Write a recursive function to compute the sum of the digits of a given positive integers.

Example:

```
234-->9
```

### Sum Series

Write a recursive function to compute sum of the nonnegative series n+(n-2)+... for a given input n.

### Power

Write a recursive function to compute a power of a number without using **. Example:

```
power(2,4)-->16
```

## Finite State Machine

### Target

Write a function whose input is a string of digits and an integer. Write down all possible combinations of digits and +, -, * operations that result in the given integer.

Example:

```
Target("http://wiki.math.bme.hu123"http://wiki.math.bme.hu, 6) -> '1+2+3', '1*2*3'
```

### Parenthesis

Given a string, replace the enclosed parts of the string with a **$** character. A formula is enclosed if it is surrounded by parenthesis. For example:

```
(xc)aa(c(b)) -> $$$$aa$$$$$$
```