# Documentation of Project implementation
# 2. task for IPP 2018/2019

Name and surname: Viktória Halečková
Login: xhalec00

## 1 Interpreter of XML representation of IPPcode19 (interpret.py)

This script loads xml representation of program, checks lexical and syntactic correctness of source and executes the program, or returns error code.

### 1.1 Input arguments

There is imported `OptionParser` from `optparse`. Function `add_option` is used for usage information of program, it is called when using `--help` argument. There are no implemented extensions, so there were only arguments `--source` and `--input` needed for implementation. Files of source and input are set in program by using dictionary.

### 1.2 XML representation check

Imported `ElementTree` from `xml.etree` helps with lexical and syntactic correctness check of xml source. Well-formed xml needs to have correct program element, language attribute and valid number of attributes used. In case of not well-formed xml source, program returns code 31. Other errors are represented in return code as 32.

### 1.3 Instructions

When lexical and syntactic correctness of xml source is checked, there is created instruction list. List contains information about instructions and their arguments. There is implemented while loop checking if instruction opcode used in program is valid and if instruction list contains this instruction opcode. Opcodes are saved in dictionary and for every instruction is set, what arguments it needs.

### 1.4 Error handling

There is implemented function `exit_program(return_value, message)` for printing error message on standard output and returning right code. In case of invalid xml source, program returns codes 31 or 32. Return codes from 52 to 58 are for interpreter errors and program returns code 11 if it can not open input file, or code 12 if it can not open output file.

## 2 Testing script (test.php)

This script is for automated testing of scripts parse.php and interpret.py, checks functionality of scripts, and prints HTML representation on standard output.

### 2.1 Input arguments

For parsing argument is used function `getopt`, because testing script has to deal with 7 different input arguments. Using getopt makes parsing easier. When input argument `-help` is used, program prints information about script on standard output.

### 2.2 Files creating

Implemented `foreach` loop contains if conditions, checks `.rc`, `.in` and `.out` files, in case they do not exist, testing script creates them. When `.rc` file does not exist, script creates this file containing return code 0. In case of `.in` and `.out` files, script creates files containing empty string.

## 2.3 Tests running

Function `exec` is used for running commands in command line. Output and return value are always stored as expected variables for checking with actual output and return value.

## 2.4 HTML representation

For HTML representation on standard output there is always scale, depending on used arguments, and loop `foreach` source file. After executing parse.php, creating xml representation as source file for interpret.py, return codes of scripts are stored and checked with expected return codes and output file, if needed. For comparing expected output files and output from interpreter is used function `diff`. Finally, there is always table containing successful, failed and total count of tests.

## 2.5 Error handling

For error handlings there is implemented function `exit_program(return_value, message)`, which prints error message for user and returns error code. Return codes used in `test.php` are: 10 for invalid arguments used, 11 for errors with opening input files, 12 for errors with opening output files.