High level language reverse engineering lab

Introduction

In this assignment, you will analyze bytecode of high level languages using techniques learnt during lectures and write code to reverse their effect. This assignment is worth 10 marks of your final score.

Question 1: High level reversing 1

The given executable for this question takes some arguments, performs some computations using those arguments and prints out some result. Analyze the executable to understand what computation it's performing and write a program that performs the reverse operation to recover the original data.

How we will test your solution

You will provide the following 2 files along with your solution. See the submission guidelines for more on how to submit the solution.

1. **compile.sh**: A shell script that will compile your solution to generate the executable. No arguments will be passed to this shell script when it is executed. This file is optional and required only if you implement your solution in a programming language that requires compilation such as C, C++, Java etc.
2. **run.sh**: A shell script that will execute your solution. Your solution will be passed similar arguments as the given executable file. One of the arguments will contain the output of the given executable. **This file is required: the grader will not grade your submission if this file is missing**.

Your solution can simply print the output(i.e. the recovered data) to stdout.
The sample session below illustrates how the shell scripts will be used.

```
$ sh ./compile.sh
$ sh ./run.sh <arguments>
<Output for arguments supplied>
```

Question 2: High level reversing 2

The given executable for this question provides some challenge and expects appropriate response. Analyze the executable, understand what it's doing and write a program to compute the expected answer.

How we will test your solution

You will provide the following 2 files along with your solution. See the submission guidelines for more on how to submit the solution.

1. **compile.sh**: A shell script that will compile your solution to generate the executable. No arguments will be passed to this shell script when it is executed. This file is optional and required only if you implement your solution in a programming language that requires compilation such as C, C++, Java etc.

2. **run.sh**: A shell script that will execute your solution. Your solution will be passed similar arguments as the challenge provided by the executable and it must display the corresponding response in the exact format as expected by the executable. **This file is required: the grader will not grade your submission if this file is missing**.

Unlike how the given executable provides the challenge and expects the response, we will pass the challenge to your solution as command line arguments and your solution should print out the result to standard output. The sample session below illustrates how the shell scripts will be used.

```
$ sh ./compile.sh
$ sh ./run.sh <arguments>
<Output for arguments supplied>
```