Packet problem

Introduction

The purpose of this task is to create a command line program for managing the serialization and deserialization of specific packets. The serialization part will split the input of the user into chunks and it will package them in a specific way and write them to a file. The deserialization part will read packets from a file, it will verify them and if they are valid it will print the contents to the console.

> For the purpose of this task you will have to use MD5 hashing function and Base64 encoding. You are advised to use the existing implementation of those in your framework/language of choice. You are NOT required to implement these two algorithms from scratch. You do not have to completely understand their inner workings.

Serialization

When the program is started with an input in the form of `-o output.txt` it will capture user input, split it to chunks that have length of 5 or less characters, package them and write them to a file.

Input reading

The user input should be read until a new line `\n` is detected.

Chunk

After the user input is read the program needs to split the string into chunks of 5 characters or less.
For example:

> helloworld

Would be split into two chunks

> hello

> world

Individual packet serialization

Let's pick the first chunk `hello`.

Signature

First we would have to generate the MD5 signature for that string. The MD5 signature will return an array of bytes. We will encode these bytes using Base64 encoding. The Base64 encoded MD5 signature of `hello` is `XUFAKrxLKna5cZ2REBfFkg==`

Hint for testing and exploring md5 and base64:

> You can use the following command to generate bas64 encoded hashes in *nix `echo -n "hello" | openssl dgst -md5 -binary | openssl enc -base64`
> You can also use an online tool: https://cryptii.com/pipes/md5-hash - You will have to add a block for MD5 encoding, another one for Bas64 encoding and another one to view as text

The purpose of this task is not solely about using MD5 and Base64 so if you get stuck on this step please contact us and we will be happy to advise you

Packet generation

A packet consists of three parts:

1. Size (also called a header)
2. Data
3. Signature (also called a trailer)

The three parts are concatenated together separated by `;` - `size;data;signature`

For the example chunk `hello` the resulting packet would be `5;hello;XUFAKrxLKna5cZ2REBfFkg==`

Combining the packets

The packets that we have created for the individual chunks need to be grouped together. The grouping is done by putting `|` character between the individual packets.
The resulting output for `helloworld` would be
`5;hello;XUFAKrxLKna5cZ2REBfFkg==|5;world;fXkwN6B2AYZXSwKC8vQ15w==`

Writing the output to a file

The result from the previous step of combining the packets has to be written to a file specified when the program was ran, for example - `output.txt`

Deserialization

When the program is started with an input in the form of `-i input.txt` it will read `input.txt` parse the packets, verify their integrity, signature and length and print their data.

Reading the file

The file that was passed as a command line argument needs to be opened and read as a string

Splitting the individual packets

Following the example from above if the file `input.txt` contains `5;hello;XUFAKrxLKna5cZ2REBfFkg==|5;world;fXkwN6B2AYZXSwKC8vQ15w==` we would to split this string into two packets `5;hello;XUFAKrxLKna5cZ2REBfFkg==` and `5;world;fXkwN6B2AYZXSwKC8vQ15w==`

Verifying integrity

We should extract that data part of the packet, generate its MD5 base64 encoded signature and compare it to the one in the packet. In case they are not equal we should fail by saying that the packet integrity has been compromised. We should also extract the length of the data at the beginning of the packet and compare it to the actual length of the data found in the packet. In case they don't match we should fail

Combining the data and writing it to the console

If the verification has been successful we need to print the contents of the packets. In the example that we are following - we should print `helloworld` in the console.

Examples

Deserialization

input.txt:

```
5;hello;XUFAKrxLKna5cZ2REBfFkg==|5;world;fXkwN6B2AYZXSwKC8vQ15w==
```

`myprogram -i input.txt`

```
helloworld
```

Serialization

`myprogram -o output.txt`

```
helloworld
```

output.txt:

```
5;hello;XUFAKrxLKna5cZ2REBfFkg==|5;world;fXkwN6B2AYZXSwKC8vQ15w==
```

Submission

You are required to submit a `.zip` file or a link to a public repository.

> .rar files will not be accepted

Partially completed tasks will also be accepted. If you didn’t manage to finish the whole task you are still advised to submit it.

> If you get stuck and you don’t know how to proceed please contact us and we will be happy to assist you