Labyrinth & Formats

CS 4500 Assignment B

Due Tuesday, September 24, midnight

Submission You must deliver your artifacts in a directory called `B` in your repository (master branch, see note below):

1. For task 1, `traversal.md` – the specification document.
2. For task 2, an executable called `B`, together with a file called `format` *only* containing one of these 3 strings `json`, `xml`, or `sexp`.

All auxiliary files must be put into a sub-directory called Other.

Note: When this assignment is released you will not have the details for your repository yet. These will be provided shortly after releasing this assignment.

Task 1

Your company has hired a group of stellar programmers. Your manager has put you in charge of writing a specification for the interface for a module, dubbed `Labyrinth`, which these stellar programmers will implement in your favorite language and ship back real soon.

As the name suggests, the high-level purpose of the desired module is to provide the services of a basic labyrinth. For our purposes a labyrinth is a connectivity specification for nodes in a simple graph, and a simple graph contains at most one edge between any pair of nodes. The desired software must support these operations:

1. the creation of a plain labyrinth with named nodes;
2. the addition of a colored token to a node;
3. the query whether some colored token can reach some named graph node.

Formulate your specification in a mix of English and technical terms appropriate for your chosen language. For example, in Java you would use the term package and you might use types while a Python programmer would speak of modules and use informal data definitions.

Write the specification using the **Markdown** format. When printed, it must fit on a single page. Specify the precise language (name, version) in which you expect the product to be written in.

The goal of Task 1 is to get a glimpse of the tasks that team leaders perform in software companies. This specification is quite small so that a single page should easily suffice to give precise instructions to developers.

Task 2

In this task you'll explore data exchange formats in your chosen language. We gave you a choice of supporting JSON, XML, or s-expressions. We recommend using JSON, but we'll give you the flexibility to choose a different format.

Develop a program that reads a series of well-formed XML/JSON/s-exp values from standard input (STDIN). Values are restricted to the following format:

1. strings
2. arrays, where the first element is a string
3. objects, where the object has a key "this" and whose value is a string

No other format of input should be accepted.

In addition to reading from STDIN, your program should accept two command line options: `-up` or `-down`. Based on the argument, the program will sort the values in ascending (`-up`) or descending (`-down`) order, according to the string designated string parts (i.e., the string itself, the first element of an array, or the "this" component of an object.

Example Input

Here are example inputs:

1. JSON

```
["b", 1, 2, 3]

  "a"

{"this" : "c",

    "other" : 0}
```

2. XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<array>
  <element>b</element>
  <element>1</element>
```

```
  <element>2</element>
  <element>3</element>
</array>

<?xml version="1.0" encoding="UTF-8" ?>
<element>a</element>

<?xml version="1.0" encoding="UTF-8" ?>
<object>
  <this>c</this>
  <other>0</other>
</object>
```

3. S-expressions

```
("b" 1 2 3)

("a")

((this "c")
 (other 0))
```

Example Output

If invoked with `-up` the program should output something like this for input in JSON.

```
"a"

["b", 1, 2, 3]

{"this" : "c", "other" : 0}
```

Similarly for XML and s-expressions. The output should be in the same format as the input. Specify which format your program is using (as `json`, `xml`, or `sexp`) in a text file called `format` supplied with your executable.

The goal of Task 2 is to figure out:

1. how your language processes command line arguments,
2. whether it has a good library for reading and writing structured data formats, and
3. how to deploy programs that run on Linux in your chosen language. See sources like this one for details.