M1 Info Project: Turing Machines Interpreter 2019
Due date: Sunday, Decembre 8^{th} 2019

I Reminder on Turing Machines

Definition A Turing machine is a tuple $\langle Q, \Gamma, B, \Sigma, q_0, \delta, F \rangle$ with :

- $Q = \{q_0, q_1, \ldots, q_m\}$, a finite set of *states*
- Γ, the finite set of symbols used by the machine (*vocabulary*)
- $B \in \Gamma$, a particular symbol called *blank*
- $\Sigma \subseteq \Gamma$, the *input vocabulary*
- $q_0 \in Q$, the *initial state* of the machine
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, the *transition function*
- $F \subseteq Q$, the set of *final states*

The Turing machine is deterministic when the transition function δ maps each configuration (current state, symbol on the tape) to *at most one* output (next state, symbol to write, move).

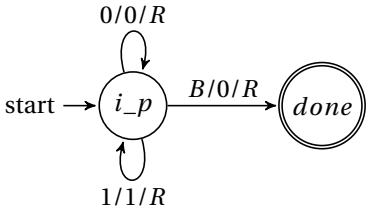

FIGURE 1 – The machine that reads an integer in binary notation, and multiplies it by 2.

II Turing Machines Interpreter

The goal of the project is to implement a program that reads the description file of a deterministic Turing machine, and that execute this Turing machine on a given input word.

Turing Machine Description File The Turing machine description file is a text file that follows these rules :

- each line of the file describes one element of the machine ;
- the initial state of the machine is described by the line `initial(XXX)`, where `XXX` is the name of the initial state ;
- the other states are described by the line `state(XXX,Y)` where `XXX` is the name of the state, and `Y` is either `T` or `F` to indicate that `XXX` is a final state (`T` : true) or not (`F` : false) ;
- the line `blank(B)` gives the name of the blank symbol ;
- the other symbols are described by `symbol(X,Y)` where `X` is the name of the symbol, and `Y` is either `T` or `F` to indicate that `X` is a symbol from the input vocabulary Σ (`T` : true) or not (`F` : false) ;
- finally, transitions are described by `transition(XXX,X,YYY,Y,M)` where `XXX` is the current state, `X` is the current symbol, `YYY` is the next state, `Y` is the symbol to be written on the tape, and `M` is the move (either `R` for right, or `L` for left).

Example 1 : The file below, `machineExample.tm`, describes the Turing machine from Figure 1.

```
initial(In Progress)
state(Done,T)
blank(B)
symbol(0,T)
symbol(1,T)
transition(In Progress,0,In Progress,0,R)
transition(In Progress,1,In Progress,1,R)
transition(In Progress,B,Done,0,R)
```

The command line The program will be executed through a command line interface, following these rules :

- The command line parameters must start with at least one of these :
  - `-steps` : the program will print all the steps of the machine execution ;
  - `-state` : the program will print the final state of the machine ;
  - `-tape` : the program will print the content of the tape at the end of the execution.

One, two or three of these parameters must be given.

- After the printing options, the path to the Turing machine description file must be given.
- The last parameter is the input word of the machine.

Example 2 : We execute the Turing machine described above on the word `101010` :

```
java -jar Turing.java -steps -state -tape machineExample.tm 101010
```

The printed output format The output of the program must follow these rules :

- if the `-steps` option is set, all the transitions must be printed : `(XXX, X) -> (YYY, Y, M)` with `XXX`, `X`, `YYY`, `Y` and `M` as described previously. ⚠ Be careful to the white spaces.
- if the `-state` option is set, then the final state is printed : `Final State: XXX` with `XXX` the name of the final state.
- if the `-tape` option is set, then the content of the tape at the end of the computation is printed : `Final Tape: XXXXXXXX` with `XXXXXXXX` the content of the tape.

The output format is exemplified below :

```
(In Progress, 1) -> (In Progress, 1, RIGHT)
(In Progress, 0) -> (In Progress, 0, RIGHT)
(In Progress, 1) -> (In Progress, 1, RIGHT)
(In Progress, 0) -> (In Progress, 0, RIGHT)
(In Progress, 1) -> (In Progress, 1, RIGHT)
(In Progress, 0) -> (In Progress, 0, RIGHT)
(In Progress, B) -> (Done, 0, RIGHT)
Final State: Done
Final Tape: 1010100
```

FIGURE 2 – Execution of the command line given at Example 2.

III Instructions

- You must implement a deterministic Turing machine interpreter, *i.e.* a program that reads a Turing machine description file, and executes the machine on a given input word. The command line parameters and the printed output must follow the rules given in the previous section.
- The solver must be implemented in Java, C or C++.
- You must deliver before Sunday, Decembre 8^{th}, 23 :59 (Paris time) the whole source code, and a script `build.sh` that allows to build an executable file (or an executable jar archive). The solver must work on Ubuntu (18.04 Bionic Beaver) or MacOS (10.15 Catalina).
- A short report of the project is expected. The report must describe every programming choice (including the class diagram, and any algorithm that you consider to be non-trivial). The report must be delivered as a pdf file.
- The interpreter must be jointly implemented by two or three students.
- The source code, the script `build.sh` and the pdf file must be contained in a directory corresponding to the students' names (for instance `JohnDoe_JaneDoe`, see below). This directory must be uploaded on Moodle as an archive (zip or tar.gz).

```
JohnDoe_JaneDoe/
├── src/
├── build.sh
└── report.pdf
```

- There will be penalties for any violation of these instructions.