CSE 30151 Spring 2014
Project 3
Due on Thursday, April 17 at 11:59pm

Project Description

This project consists of developing the third and final simulator for this course, Turing machines (TM), and running experiments with it. This is a team project in teams consisting of two students (students can also submit projects individually, but teams are encouraged). The overall specifications and guidelines are similar to the previous projects, and it should be possible to reuse their code.

For project 3, each team is to create a program that simulates a deterministic TM. The program should read a description of a TM and one or more inputs for it. It will process one input and output the execution path and other information about it. After processing one input, the program should reset the automaton and proceed with the next input.

The project will also include writing Turing machines for different types of functionality and experimenting with them on various inputs.

Machine description

A machine description will be provided in a file. The file will consists of text lines defining all parts of a TM. The format is as follows:

- A line starting with "`Q:`" will list the set of states, where the states are separated with a comma (with no spaces) and the list is terminated by the end-of-line character. For example, a list of states can look like `Q:q1,q2,q3`.
- A line starting with "`A:`" will list the input alphabet Σ, where the symbols are separated with a comma (with no spaces) and the list is terminated by the end-of-line character. For example, the alphabet for binary strings will be given as `A:0,1`. The blank character " " is not allowed to be in the input alphabet.
- A line starting with "`Z:`" will list the tape alphabet Γ, where the symbols are separated with a comma (with no spaces) and the list is terminated by the end-of-line character. The input alphabet and the blank character " " should always be a part of the tape alphabet.
- A line starting with "`T:`" provides a single transition rule. The format of the line is the starting state, the current tape symbol, the resulting state, the symbol to be written on the tape, and the direction for the tape head (all are separated by commas with no spaces). The only valid choices for the head direction are `L` and `R`. For example, we might have a line `T:q1, ,q2,a,L`. There will be multiple lines of this form per machine description. The program must check that both states given in the transition function are valid states from the Q set, both characters are valid tape characters, and the direction is valid.
- A line starting with "`S:`" will specify the start state. For example, we might have `S:q1`. The program must check that the state given is a valid state.
- A line starting with "`F:`" will specify (exactly) two states: the accept and reject states (in that order). They must be valid different states.

Since the project deals with deterministic TM, upon reading all transition rules from the machine's description, the program must verify that computation can proceed deterministically. That is, no two transition rules can have the same starting state and the same current tape symbol. Some transition rules can be omitted and should be assumed to be given implicitly: if a simulator encounters a configuration for which there is no transition rule, it should transition to the reject state. For consistency, assume that in such cases the TM does not change the contents of the tape and transitions right. The reject state will always be specified explicitly. Transitions from the accept and reject states are not allowed.

The tape head is assumed to be positioned on the leftmost symbol of the tape at the beginning of each simulation. If more than one input is specified, the simulator will reset the state and move the tape head to the leftmost position after each simulation.

There will be a description of a single machine in a file. The file name will be given as an argument to the program. As a sample description, consider the machine presented in class for the language $L = \{x\#y \mid x, y \in \{0,1\}^* \text{ and } |x| < |y|\}$:

```
Q:q1,q2,q3,q4,q5,q6,q7,q8
A:0,1,#
Z:0,1,#, ,x
T:q1,0,q2,x,R
T:q1,1,q2,x,R
T:q2,0,q2,0,R
T:q2,1,q2,1,R
T:q2,#,q3,#,R
T:q3,x,q3,x,R
T:q3,0,q4,x,L
T:q3,1,q4,x,L
T:q4,x,q4,x,L
T:q4,#,q5,#,L
T:q5,0,q5,0,L
T:q5,1,q5,1,L
T:q5,x,q1,x,R
T:q1,#,q6,#,R
T:q6,x,q6,x,R
T:q6,0,q7,0,L
T:q6,1,q7,1,L
S:q1
F:q7,q8
```

Tape description

Once the program is initialized with a TM description, it will take tape input from the standard input. It should silently wait for input without any prompts. The format is as follows:

- The first line will contain a single integer by itself indicating how many tape inputs follow.
- If the first line contained a number n, then the next n lines will contain tape inputs, one sequence per line. Each line will consist of a list of alphabet symbols possibly with blank characters (but no trailing blank characters), each separated by a comma. For example, if the alphabet is $\Sigma = \{0,1\}$, then one input line could be `0,0,1,1,0` or `1,0,1, ,1`. This

input should be interpreted as a beginning of an infinite tape, where all characters after the last specified symbol are blank, and the head should be able to move beyond the last (i.e., rightmost) character of the input string.

TM simulation and program output

Since TMs may not halt on some input strings, we will impose an upper bound on the number of steps during TM simulation. For that reason, the simulator is to run until the TM halts, or to perform 1000 transitions and stop, whichever comes first.

The program should print all information to the standard output. For each input tape, the program's output will consist of a trace of program execution. Unlike previous simulations, this time the program will output the current configuration of the TM after each transition. The initial configuration should be printed regardless of whether the machine later performs any transitions or not. Then after each transition the current configuration is to be printed as well. Each configuration will consist of the following:

1. the sequence of the tape symbols from the leftmost position until the position to which the head points (not including that cell). The symbols should be enclosed between ( ) and separated with commas (no spaces). If the head is positioned on the leftmost square of the tape, the list will be empty, but the brackets still should be output.

2. the current state;

3. the sequence of tape symbols starting from the position under the tape head to the rightmost non-blank character on the tape. The symbols are also to be enclosed between ( ) and separated by commas. This list will be empty if the head points to a blank character and all characters to the right of it are also blank.

If the machine enters the accept state, the program should output `ACCEPT` on a line by itself; if it enters the reject state, the program should output `REJECT`. If the limit on the number of transitions is reached and the machine has not halted, the simulation stops and the program should output `DID NOT HALT`. Then machine will reset and proceed to processing the next input tape if more inputs are remaining. For example, given the TM above, if the input is:

```
2
0,#,0,1,1
1,1
```

the program's output will be:

```
()q1(0,#,0,1,1)
(x)q2(#,0,1,1)
(x,#)q3(0,1,1)
(x)q4(#,x,1,1)
()q5(x,#,x,1,1)
(x)q1(#,x,1,1)
(x,#)q6(x,1,1)
(x,#,x)q6(1,1)
(x,#)q7(x,1,1)
ACCEPT
```

```
()q1(1,1)
(x)q2(1)
(x,1)q2()
(x,1, )q8()
REJECT
```

Turing machine programs

Another part of the project involves creating and running Turing machines using the simulator. The submission should include three non-trivial Turing machines (which differ from the languages or functions for which TM *state diagrams* were given in the lectures, the textbook, or supplied with this project) from the following categories:

- one TM performing arithmetic operation(s);
- one TM performing operation(s) on strings or sets;
- one TM of any other kind of your choice.

For each of the TMs above, you are to submit the machine description file (as specified in this document), provide a description of the language/functionality and machine design in the report, and provide at least three inputs corresponding to different classes of inputs for each machine. The goal of having substantially different inputs for each machine is to provide adequate testing of the implemented functionality. For instance, if a TM is designed to insert a character into a string, three inputs might correspond to cases when insertion is to be made at the beginning, middle, and at the end of the string. Similarly, for addition, you might consider adding zero and non-zero numbers. The choice of such inputs and the results of TM execution of the inputs are to be documented in the report file.

Program submission and execution

The program should be submitted using the dropbox for this course of one of the students in the team. The name of the submission should be `proj3`; that is, create a directory named `proj3`, place all of the files to be submitted into that directory and submit it. The program can be written in a programming language of your choice, but **it must run on CSE student machines** student00–03.cse.nd.edu. If we are unable to compile or run it, the **project risks getting zero credit**.

Your submission should have a Makefile with rules to compile the program, unless the programming language you are using does not require compilation. Typing `make` in the submission directory should produce an executable named `tm`. The program should take one argument, which is the name of the file containing a TM description. The tape inputs will be provided from the standard input and all output should go to the standard output.

To test the programs, we will use input and output redirection. It is recommended that you test the program using the same mechanism on the sample files (provided) as well as your own inputs. The program will be tested as follows:

```
> make
```

If the program requires building, this should produce an executable called `tm`.

```
> ./tm tm1.txt < tm1-input.txt > tm1-output.txt
```

This command line corresponds to program execution using files `tm1.txt` and `tm-input1.txt`, which are supplied as a part of this project. After the execution, your program's output will be

written to file `tm-output1.txt`. The contents of that file should be identical to the contents of the file with the same name supplied as a part of the project.

If the program cannot be run as above and requires special handing (e.g., usage of Python or Java interpreter), create a small shell script that would execute the appropriate command and pass the argument to the program, so that the program can be run using the syntax specified above. It is recommended that in this case you submit a README file describing how the program is being invoked.

Project report

You are encouraged to experiment with the project running it on different TMs and inputs. Besides the program's source code, your submission should include three TMs and their inputs, as well as a project report (a PDF document) describing the following:

- The language(s) and platform used for testing.
- A high level description of the program including the algorithms and data structures used (for storing and verifying the program description, for the simulation itself such as handling of the tape, etc.).
- Description of the functionality and implementation of the designed TMs; description of the inputs and what functionality of the TM each of them tests, and the results of TM simulation on the inputs.
- Any major difficulties encountered or any notable implementation decisions.

To summarize, the submission will include:

- the source file(s);
- Makefile if necessary;
- README file if necessary;
- a PDF report file;
- TMs of different types and their inputs;
- supplemental files (if any).