tint

A Turing machine Interpreter- Carter Codell

-Tasneem Naseem

Download

Download the appropriate executable file based on your computer system or OS.

- `tint-linux` is compiled for the Linux OS with amd64 architecture.
- `tint-windows` is compiled for the Windows OS with amd64 architecture.
- `tint-mac` is compiled for the Mac OS with amd64 architecture.

Note: If downloaded “tint” with Safari on a Mac, Safari may add a spurious extension “.dms”. Students should remove that extension.

Simulator/ Tint folder

- Create a new folder/directory on your Desktop
- Move the downloaded tint executable to this new folder
- Rename the executable to **tint**
- Run the executable here:
    - Or by running ./tint from the command line(for Linux/Unix systems)
    - Or by running start tint from the command line(for Windows systems)
- Note: If you are using command line make sure you are in the folder you created before running the executable. Also check for the folder/directory permissions
- Use “chmod 744 tint” to change access permissions if needed

Machine file

- YAML-specified machine with listed states and transitions
- Use the template from the example files till you are familiar with writing your own
- The file can have a .txt extension or no extension. We use the .yaml extension just to remember it has to be in YAML format

Common mistakes to avoid

- Leaving out indentation for the transitions.
- Using tabs instead of spaces for transitions.
- Leaving a special character unquoted.
- Not putting a space after ":", "-", or ",".
- Not putting a "#" to begin a comment.
- Forgetting to put commas (",") in between states or values in a transition.
- Misspelling.
- Copy and paste errors.

Test file

- The test file is used to simulate the machine
- On each line, there is a single, unquoted test
- For example,
  - `a a b b a a`
  - `a b a`
  - `one two three`
- There has to be a space between each character recognized by the language on the same line
- Each new test case/sample on a new line
- The test file should end with a new empty line otherwise the last string won’t be simulated.

Using/running tint

Command:

```
./tint -m MACHINE_TYPE MACHINE_FILE TEST_FILE(Mac/Linux)
```

```
tint -m MACHINE_TYPE MACHINE_FILE TEST_FILE(Windows)
```

To use `tint` you must use the -m flag to specify a machine type-

- "dfa"
- "tm"

You can also give a single test string in quotes instead of TEST_FILE for the purpose of testing

Using/running tint

There are two flags more that can be used with the tint command. These are the -v and -t flags

- The -v flag prints each simulation verbosely: step by step.
- The -t flag interprets the test file as a single, quoted test.Here is an example of using this flag:

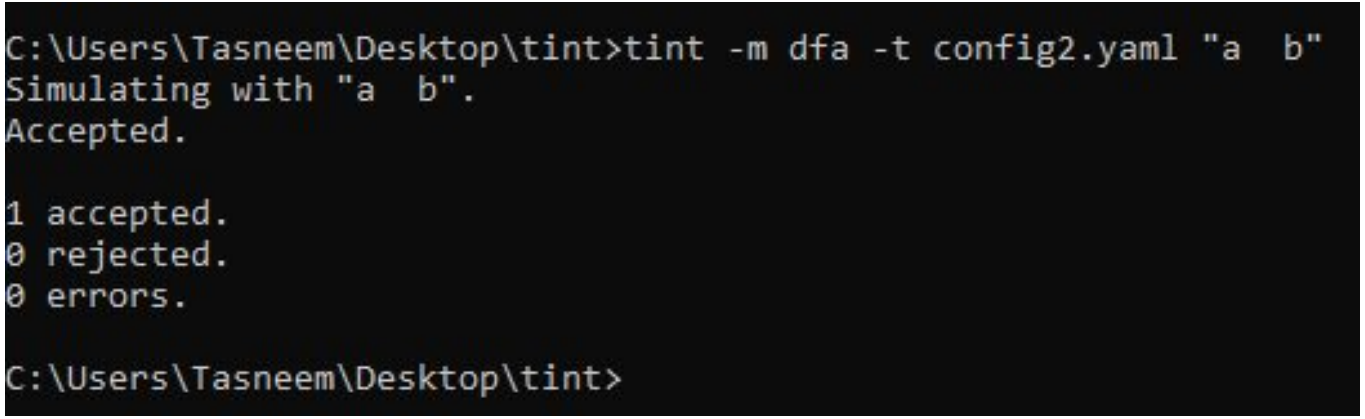
Output

Note: Here example.yaml file is a machine over language (a, b, c)that accepts any string with ‘a b c ’ as substring

Using command (with -t flag):

./tint -m dfa -t config2.yaml "a   b" (For Mac)

tint -m dfa -t config2.yaml “a b b ” (For Windows)



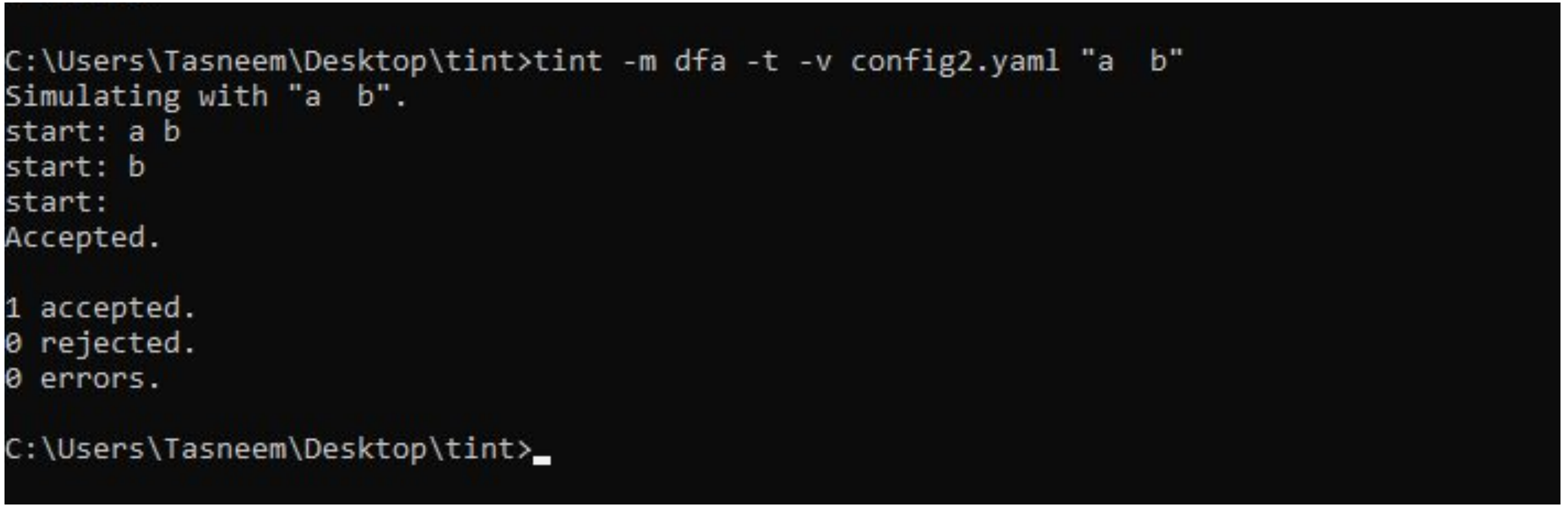
```
C:\Users\Tasneem\Desktop\tint>tint -m dfa -t config2.yaml "a  b"
Simulating with "a  b".
Accepted.

1 accepted.
0 rejected.
0 errors.

C:\Users\Tasneem\Desktop\tint>
```

Output

Using command (with -t -v flag):

tint -m dfa -t -v config2.yaml "a   b"

Output:



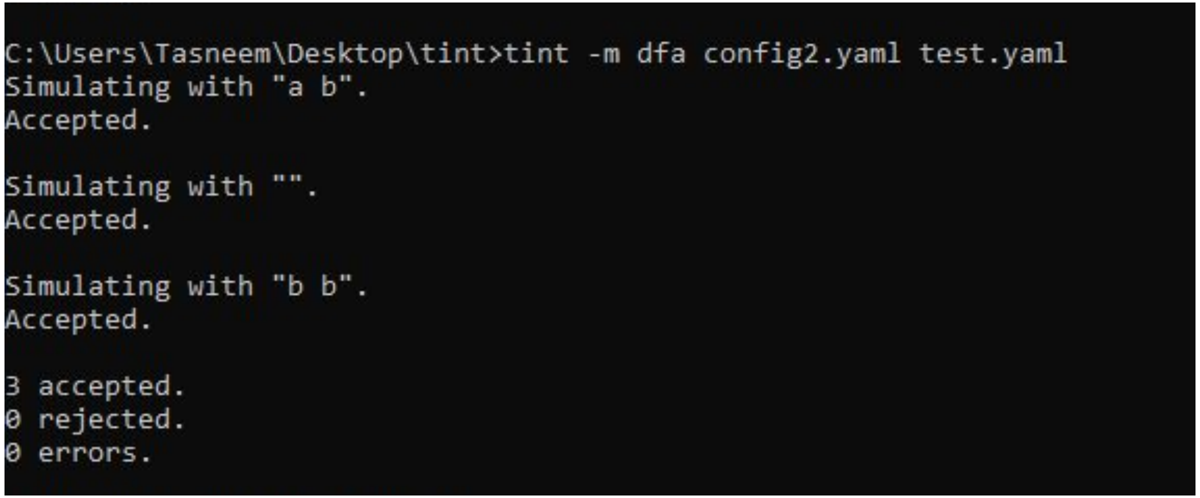
```
C:\Users\Tasneem\Desktop\tint>tint -m dfa -t -v config2.yaml "a  b"
Simulating with "a  b".
start: a b
start: b
start:
Accepted.

1 accepted.
0 rejected.
0 errors.

C:\Users\Tasneem\Desktop\tint>_
```

Output

Using command (when using a text file for testing **do not** use the -t flag):

tint -m dfa config2.yaml test.yaml

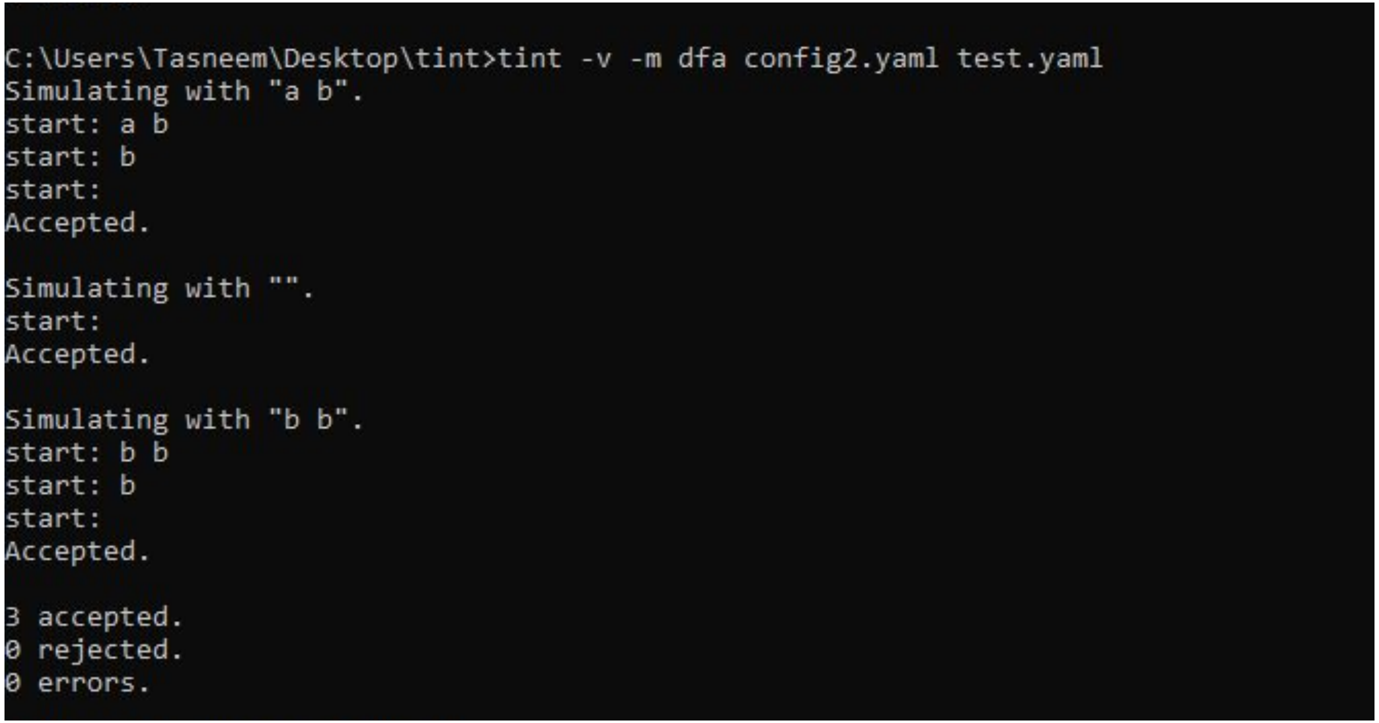
Output:



```
C:\Users\Tasneem\Desktop\tint>tint -m dfa config2.yaml test.yaml
Simulating with "a b".
Accepted.

Simulating with "".
Accepted.

Simulating with "b b".
Accepted.

3 accepted.
0 rejected.
0 errors.
```

Output

Using command (you **can use** -v flag while using a text file for testing):



```
C:\Users\Tasneem\Desktop\tint>tint -v -m dfa config2.yaml test.yaml
Simulating with "a b".
start: a b
start: b
start:
Accepted.

Simulating with "".
start:
Accepted.

Simulating with "b b".
start: b b
start: b
start:
Accepted.

3 accepted.
0 rejected.
0 errors.
```