Requirements Specification

Washington State University – Tri Cities

CPT_S 322: Software Engineering Principles I

Project:

Interactive Turing Machine Console Application

Submitted by:

Raul Y. Martinez

Course Instructor:

Dr. Niel B. Corrigan

Date:

February 24th, 2025

Table of Contents:

Title Page ---------- 1

Table of Contents ----------2

List of Figures ----------4

Revision History ----------5

1.0 Introduction ----------6

2.0 Background ----------7

2.1 Formal Specification ----------8

2.2 Example Computation ----------9

3.0 Overview ----------13

4.0 Environment ----------14

4.1 Input and Output Devices ----------14

4.2 Turing Machine Definition File ----------15

4.3 Input String File ----------17

5.0 Operation ----------18

5.1 Invocation ----------18

5.1.1 Command Line ----------18

5.1.2 Configuration Settings ----------18

5.1.3 Opening Turing Machine ----------19

5.2 Commands ----------19

5.2.1 Help User (H) ----------19

5.2.2 Show Status (W) ----------20

5.2.3 View Turing Machine (V) ----------20

5.2.4 List Input Strings (L) ----------20

5.2.5 Insert Input String (I) ----------20

5.2.6 Delete Input String (D) ----------21

5.2.7 Set Transitions (E) ----------21

5.2.8 Truncate Instantaneous Descriptions (T)----------21

5.2.9 Run Turing Machine (R) ----------21

5.2.10 Quit Turing Machine (Q) ----------22

5.2.11 Exit Application (X) ----------22

5.3 Termination ----------22

5.3.1 Closing Turing Machine ----------22

References ----------23

List of Figures

Figure 2.1: Image of Alan Turing----------7

Figure 2.2 Image of Physical Model of a Turing Machine----------8

Figure 2.3: Table with Example Transition Table----------9

Figure 2.4: Image of Turing Machine Formal Specification Sheet ----------11

Figure 4.1: Image of Prototype Menu----------14

Revision History

1st Edition----------2/24/2025

2nd Edition----------4/21/2025

Changes: Minor formatting changes and minor content alignment changes for accuracy.

1.0 Introduction:

This semester of CPT_S 322 Software Engineering Principles I each individual student will be fully implementing a command-line based Turing machine application. This requirements specification is meant to be an outline for this project and to specify the core functions of the Turing machine in order to provide a clearer understanding to both the student Software Engineer and the Project Sponsor and auditor (Dr. Niel B. Corrigan) of the steps necessary to complete this application with the fixed due date of the last class day of the 2025 Spring Semester per the official Washington State University semester scheduler.

This requirements specification is outlined in a clear and concise manner to provide a comprehensible outline of the requirements of the Turing machine application. This document contains an introduction providing a high-level overview of the project and application due at the end of the 2025 Spring Semester, background on the Turing machine including some history and its inner workings as a mathematical model, a software and hardware based technical overview, an environment section where input and output devices used in the application are be described in detail, an operation section with detailed information on the actions of the Turing machine application as well as how interacting with it works, and a references section for proper documentation of all resources used in any section of this requirements specification.

2.0 Background

A **Turing machine (TM)** is an abstract mathematical model used to define computation and algorithmic processes. It was introduced by **Alan Turing in 1936** as a theoretical construct to formalize the concept of computability. The Turing Machine TM is considered to be the foundation of modern computing as it defined the key characteristics of any computer. A device becomes a computer when it is able to accept input, process data based on a set of instructions (program), store information, and produce output. The Turing machine is a very early attempt at achieving a device with these qualities. A Turing machine consists of:

- An **infinite tape** divided into discrete cells, each capable of holding a symbol.
- A **tape head** that moves **left (L) or right (R)** and can **read, write, or overwrite** symbols.
- A **finite set of states** that control its behavior based on a **transition function**.
- A **transition function (δ)** that dictates the machine's movement and operations.

The Turing machine begins in an **initial state** and processes an input string according to the transition rules. It may enter an **accepting state** if the input is valid or a **rejecting state** if the input does not conform to the machine's logic. This simple yet powerful model can simulate any computable function, making it fundamental to **computability theory, complexity theory, and formal language theory**.

Figure 2.1: Image of Alan Turing.

Figure 2.2: Image of a physical model of a Turing machine.

2.1 Formal Specification

For this application, the **formal definition of the Turing Machine (TM)** is provided by **Dr. Niel B. Corrigan**, serving as the **customer** (class instructor) for this project. The Turing machine is formally defined as:

$$M=(Q,\Sigma,\Gamma,\delta,q0,B,F)$$

where:

- **Q**: A finite set of states.
- **Σ**: A finite input alphabet ({a,b}).
- **Γ**: A finite tape alphabet ({a,b,X,Y,−})
- **δ**: The transition function $\delta:Q\times\Gamma\rightarrow Q\times\Gamma\times\{L,R\}$ which determines state transitions, symbol writing, and tape movement.
- **q0**: The initial state (s0).
- **B**: The blank symbol (-).
- **F**: The set of final (accepting) states (s4).

A string x ∈ Σ^* is **accepted** by M if, after a series of valid transitions, the machine reaches a state where the **tape head encounters a blank space (-) and enters an accepting state (s4)**.

The transition function for a **Turing machine that recognizes the language** $L=\{a^n b^n | n\geq 1\}$ is defined as follows:

Current State	Read Symbol	New State	Write Symbol	Move
s0	a	s1	X	R
s0	Y	s3	Y	R
s1	a	s1	a	R
s1	b	s2	Y	L
s1	Y	s1	Y	R
s2	a	s2	a	L
s2	X	s0	X	R
s2	Y	s2	Y	L
s3	Y	s3	Y	R
s3	-	s4	-	R

Figure 2.3: Example Transition Table for TM

The machine **marks** each 'a' with 'X' and the corresponding 'b' with 'Y' while ensuring they are paired correctly. If the input matches the pattern $a^n b^n$, the machine **halts in state s4**; otherwise, it rejects the input.

2.2 Example Computation

To illustrate a full execution of a **Turing machine**, consider this simple case where the machine processes the string of characters **"aabb"** which in this case are in the input alphabet $a^n b^n$, here is how the turing machine should behave based on this example input alphabet and string:

1. The machine starts at **state s0** and marks the first 'a' as 'X'.
2. It moves right, skipping remaining 'a's, and marks the first 'b' as 'Y'.
3. It moves left to find the next 'a', marking it as 'X'.
4. The next 'b' is marked as 'Y', and the machine ensures there are no extra characters.
5. If all 'a's and 'b's are matched, the machine transitions to **state s4** and halts, indicating acceptance.

Here is an example definition file for the turing machine which ought to be included with the naming format anbn.def if that is the intended input alphabet.

anbn.def:

STATES: s0 s1 s2 s3 s4

INPUT_ALPHABET: a b

TAPE_ALPHABET: a b X Y -

TRANSITION_FUNCTION:

s0 a s1 X R

s0 Y s3 Y R

s1 a s1 a R

s1 b s2 Y L

s1 Y s1 Y R

s2 a s2 a L

s2 X s0 X R

s2 Y s2 Y L

s3 Y s3 Y R

s3 - s4 - R

INITIAL_STATE: s0

BLANK_CHARACTER: -

FINAL_STATES: s4

And here is the official Turing machine specification document guidelines provided by customer Dr. Niel B Corrigan:

TURING MACHINE SPECIFICATION

Computer Science 322
Spring Semester, 2025

A Turing machine is specified by $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where

Q is a finite set of states
$\Sigma \subseteq (\Gamma - \{B\})$ is a finite input alphabet
Γ is a finite tape alphabet
δ is a transition function from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
$q_0 \in Q$ is the initial state
$B \in \Gamma$ is the blank symbol
$F \subseteq Q$ is a finite set of final states

A string $x \in \Sigma^*$ is accepted by M if $q_0 x \vdash^* \alpha_1 p \alpha_2$ for some $p \in F$ and $\alpha_1\alpha_2 \in \Gamma^*$. Otherwise, the string is rejected.

A Turing machine which accepts the language $L = \{a^n b^n \mid n \geq 1\}$ is shown below. Any transition not specified in this machine is undefined.

$Q = \{s0, s1, s2, s3, s4\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, X, Y, -\}$

$\delta(s0, a) = (s1, X, R)$
$\delta(s0, Y) = (s3, Y, R)$
$\delta(s1, a) = (s1, a, R)$
$\delta(s1, b) = (s2, Y, L)$
$\delta(s1, Y) = (s1, Y, R)$
$\delta(s2, a) = (s2, a, L)$
$\delta(s2, X) = (s0, X, R)$
$\delta(s2, Y) = (s2, Y, L)$
$\delta(s3, Y) = (s3, Y, R)$
$\delta(s3, -) = (s4, -, R)$

$q_0 = s0$

$B = -$

$F = \{s4\}$

Figure 2.4: Turing Machine Formal Specification Sheet Provided by Dr. Niel B Corrigan.

This application aims to **simulate** Turing machines through a **command-line interface**, allowing users to:

- Define their own Turing machines by specifying states, transitions, and input alphabets.
- Load input strings and test them against machine rules.
- Step through execution to visualize tape modifications in real time.
- Analyze computations by tracking state transitions.

By providing an **interactive simulation**, the application serves as a practical tool for **students, educators, and researchers** studying formal computation models as well for student software engineers to gain insight into a full development cycle of an application of substantial difficulty.

3.0 Overview:

The application will be **developed, compiled (with GNU GCC), and executed in a Unix-based Ubuntu Linux command-line environment via SSH to WSU-TC's elec servers**, ensuring compatibility with a range of operating systems, including **Linux, macOS, and Windows (via WSL or a VM)**. It will be implemented in **C++**, utilizing the **standard input/output streams and file handling** for loading Turing machine definitions and input strings. Users will interact with the program through a text-based interface on the command-line, issuing commands to manipulate and execute Turing machines.

Users of this application may include:

- **Computer Science students** studying automata theory and formal computation.
- **Educators** who want to demonstrate Turing machine execution in courses.
- **Researchers and enthusiasts** exploring computational models and decidability problems.

To operate the application, users will:

1. **Launch the program** via the command line.
2. **Load a Turing machine definition file**, specifying states, transitions, and symbols.
3. **Provide an input string file** for processing by the machine.
4. **Execute the machine step-by-step** or in full, observing state changes and tape modifications.
5. **Interact with built-in commands** to view the machine's configuration, modify transitions, or list available inputs.
6. **Terminate execution** and exit the application when finished.

Overall, this application is designed to help users explore Turing machines, a fundamental concept in computer science. It provides a way to experiment with how computations work in theory by simulating a step-by-step execution of a Turing machine. By using this tool, users and the developers (students) will better understand key ideas in computability, algorithm design, and the limits of computation, and software engineering in a UNIX-like environment.

For students developing this application, its purpose will be to show what a full development cycle is like in its entirety from design to implementation to submission.

4.0 Environment:

The Turing machine application operates in a **command-line environment**, relying on **text-based input and output** for interaction. Users will provide machine definitions and input strings through **text files**, while the application will display results and execution steps in the terminal. This section describes the devices and files used by the application, including expected formats and possible errors.

4.1 Input and Output Devices

The application is designed to run in a **terminal or command prompt** on operating systems such as **Linux, macOS, and Windows (via WSL or a VM)**. It accepts input in the following ways:

- **Keyboard Input** – Users can enter commands to load Turing machine definitions, input strings, and control execution.
- **Text Files** – The application reads predefined Turing machine definitions and input strings from structured text files.
- **Terminal Output** – Results, state transitions, and tape contents are displayed as plain text in the command-line interface.

There are **no graphical elements** in this application per the requirements outlined by Dr. Niel B Corrigan; all interactions are handled through text-based commands which will be the following:

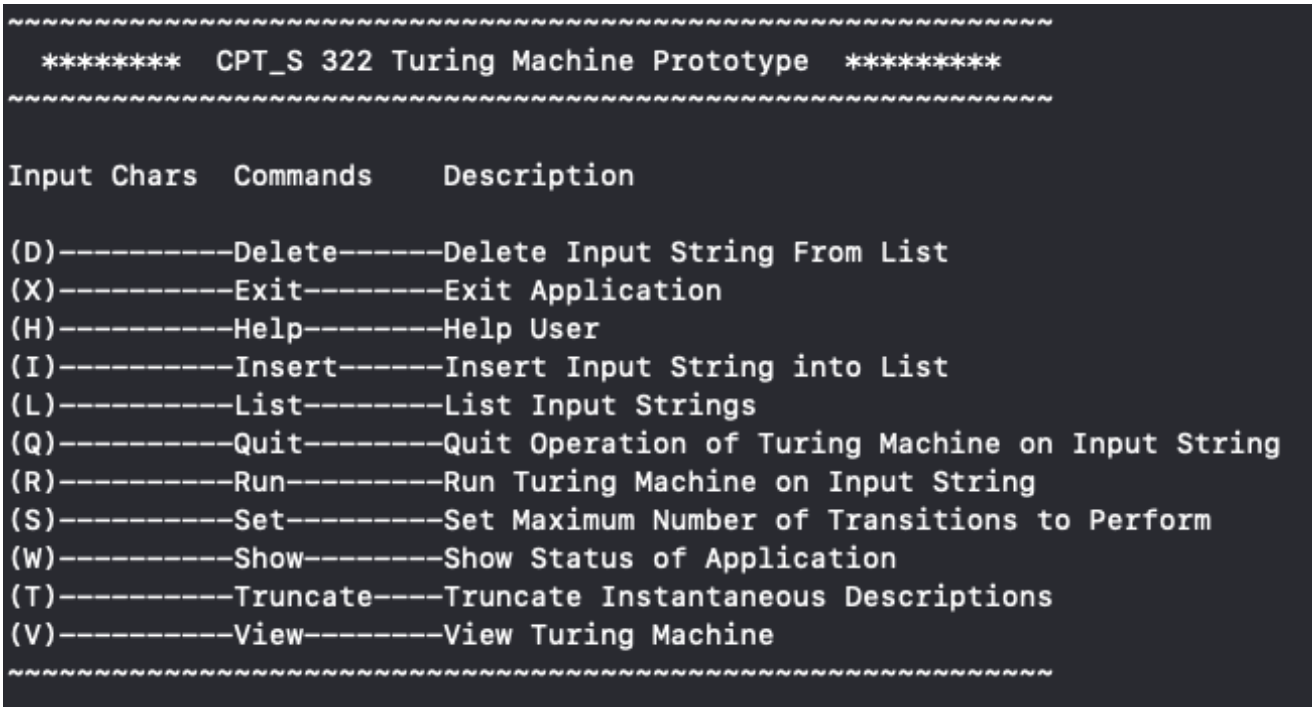

Figure 4.1: Picture of Command from Prototype created by Raul Y. Martinez (author).

4.2 Turing Machine Definition File

The Turing machine application requires a **definition file** (ex: anbn.def) to specify the states, input alphabet, tape alphabet, transitions, and other essential components of the Turing machine. The file follows a structured format where each section is explicitly labeled.

Below is an example definition file, **anbn.def**, which describes a Turing machine that accepts strings of the form $\mathbf{a^n b^n}$, meaning one or more 'a's followed by the same number of 'b's.

andn.def:

This Turing machine accepts the language of one or more a's followed by the same number of b's.

STATES: s0 s1 s2 s3 s4

INPUT_ALPHABET: a b

TAPE_ALPHABET: a b X Y -

TRANSITION_FUNCTION:

s0 a s1 X R

s0 Y s3 Y R

s1 a s1 a R

s1 b s2 Y L

s1 Y s1 Y R

s2 a s2 a L

s2 X s0 X R

s2 Y s2 Y L

s3 Y s3 Y R

s3 - s4 – R

INITIAL_STATE: s0

BLANK_CHARACTER: -

FINAL_STATES: s4

Explanation of the Definition File anbn.def:

- **States**: The machine has five states (s0 to s4), including an initial state and an accepting state.
- **Input Alphabet**: The input consists of {a, b}.
- **Tape Alphabet**: In addition to a and b, the tape includes X (to mark processed 'a's), Y (to mark processed 'b's), and - (the blank symbol).
- **Transition Function**: Defines how the machine moves, marks symbols, and progresses through states.
- **Initial State**: The machine starts in s0.
- **Blank Character**: - represents the blank tape space.
- **Final State**: The machine halts in s4 if the input is valid.

Errors in Definition Files:

If the definition file contains errors or is incorrectly formatted, the application will return an **error message**. Common errors include:

- **Missing states or transitions** (e.g., undefined start state)
- **Invalid symbols** (e.g., symbols not in the tape alphabet)
- **Incorrect formatting** (e.g., missing arrows in transitions)
- **Missing Keywords** (e.g., Missing “FINAL_STATES:” in .def file, this will cause an err.
- **Incorrect Whitespace** (e.g., If the Turing Machine reads in whitespace incorrectly instead of an input symbol an error will be thrown).

The application will display an **error message**, reject the file, and the user will need to correct the errors before the Turing machine can be loaded.

4.3 Input String File

The Turing machine application requires an **input string file** that contains test strings for processing by the Turing machine. Each line in the file represents a separate input string to be evaluated by the Turing machine.

Below is an example input string file, **anbn.str**, which provides test cases for the $\mathbf{a^n b^n}$ Turing machine.

Example Input String File:

anbn.str:

```
a
ab
\
aaabb
aaaaaaaaaabbbbbbbbbb
aabb
aaaaaabbbbbb
ba
aba
bb
```

Explanation of Input Strings

Each input string is tested to determine whether the input string is in the input alphabet (a^n b^n):

- **Valid Inputs (Accepted by the Machine)**:
  - ab → One 'a' followed by one 'b' (Accepted)
  - aabb → Two 'a's followed by two 'b's (Accepted)
  - aaaaaaaaaabbbbbbbbbb → Ten 'a's followed by ten 'b's (Accepted)
  - aaaaaabbbbbb → Six 'a's followed by six 'b's (Accepted)
- **Invalid Inputs (Rejected by the Machine)**:
  - a → Missing a 'b' (Rejected)
  - \ → Likely an empty input (Rejected)
  - aaabb → Mismatched 'a's and 'b's (Rejected)
  - ba → 'b' appears before 'a' (Rejected)
  - aba → Improper order of symbols (Rejected)
  - bb → Missing 'a's (Rejected)
  - → Whitespace (Rejected)

Errors in Input Files

If an input string contains **invalid symbols** (i.e., characters do not part of the defined input alphabet {a, b}), the application will return an error message and stop the application. Example errors include:

- Using numbers or special characters (e.g., a2b).
- Empty lines or formatting issues.
- Spaces within an input string (e.g., a b instead of ab).

The application ensures that only **well-formed inputs** are processed while notifying users of any **incorrect or malformed** entries. This as specified by Dr. Niel B. Corrigan is done to ensure the Turing machine application only runs on proper input files.

5.0 Operation

The Turing machine application operates through a **command-line interface**, allowing users to load, configure, and execute Turing machines. This section describes the actions performed at the beginning and end of execution, as well as the available user commands.

5.1 Invocation

The application is invoked through the **command line** and requires a **Turing machine definition file** and an **input string file** to function properly.

5.1.1 Command Line

The application is launched using the following format:

Loading…
Turing Machine Loaded Successfully!
./tm anbn
Command: L
1. aabb

./tm anbn loads the Turing machine defined in anbn.def and the input strings from anbn.str.

5.1.2 Configuration Settings

Before running the turing machine (after compilation), the user may **configure** other maximum number of transitions using the **Set** command (see Section 5.2.7). This prevents infinite loops and ensures controlled execution. However, there are defaults pre-configured.

5.1.3 Opening Turing Machine

Once the application starts, it attempts to **load the Turing machine definition file (anbn.def) and the input string file (anbn.str)**. If both files are valid the machine is successfully initialized and loaded, and the user can interact with it. If the file is **malformed or missing**, a polite **error message** is displayed, and the program terminates.

5.2 Commands

After successfully loading a Turing machine, users can **execute various commands** to interact with it. Commands are **case-insensitive** and can be entered using their **corresponding input character**.

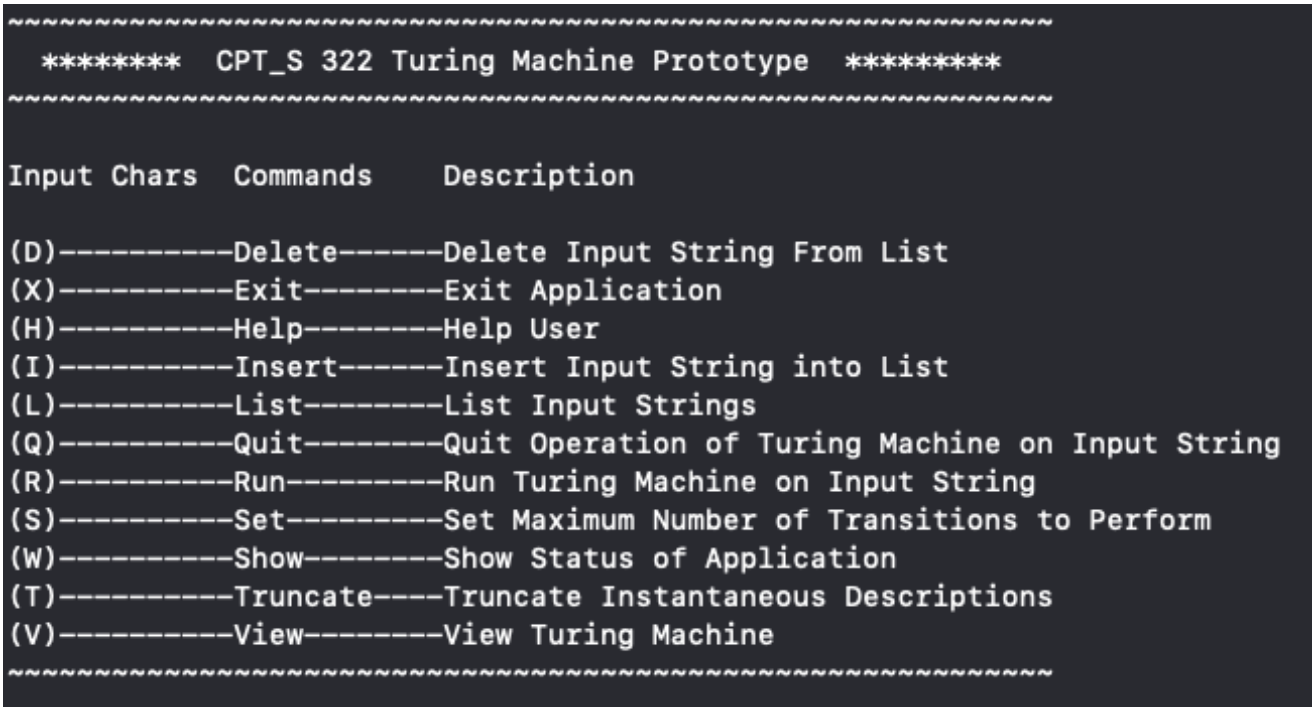

Figure 4.1: Picture of Command from Prototype created by Raul Y. Martinez (author).

5.2.1 Help User (H)

Displays a list of available commands along with their descriptions. This helps users understand the operations they can perform. As well as indicates symbol to execute.

5.2.2 Show Status (W)

Displays the **current status of the application**, including:

- Whether a Turing machine is loaded.
- The number of input strings available.
- The maximum number of transitions allowed.
- Information about the Developer of the Application

5.2.3 View Turing Machine (V)

Displays the **current Turing machine definition file** and all of its information such as:

- The set of states.
- The input and tape alphabets.
- The transition function.
- The initial state.
- Blank Character.
- Final States.

5.2.4 List Input Strings (L)

Lists all **available input strings** that can be processed by the Turing machine.

Example Usage:

Command: L

1. aabb

2. aaabbb

3. ab

5.2.5 Insert Input String (I)

Allows the user to **add a new input string** to the list.
Example usage:

Command: I
New String: aabb

This inserts aabb into the input string list.

5.2.6 Delete Input String (D)

Removes an existing input string from the list.
Example usage:

Command: D
String to Delete: 1

This deletes ab from the available input strings (if ab is string 1 of course).

5.2.7 Set Transitions (E)

Sets the **maximum number of transitions** the machine can perform before stopping. This prevents infinite loops.
Example usage:

Command: S
Current Maximum Number of Transitions: 1
Maximum Number of Transitions: 1000

This limits execution to **1,000 transitions**.

5.2.8 Truncate Instantaneous Descriptions (T)

You are prompted with cells to truncate with the original value. Default value is 32.

5.2.9 Run Turing Machine (R)

Executes the Turing machine on a selected input string. The user after every time they click ‘r’ the Turing Machine will execute the selected string according to the set number of transitions (Command E), every time that ‘r’ is entered the Turing Machine will move the X and say what state it is in (s1,s2, etc.) Once the string is completed the Turing Machine will say if the string was **accepted or rejected.**

Example Usage:

COMMAND: r

Running Turing machine...

Select string index to run:

Select input string: 1

0. [s0]aab

1. X[s1]ab

COMMAND: r

Running Turing machine...

2. Xa[s1]b

COMMAND: r

Running Turing machine...

3. X[s2]aY

Runs the machine on the input aabb and displays the step-by-step execution until a rejecting state or accepting state is reached.

5.2.10 Quit Turing Machine (Q)

Stops the execution of the **currently running Turing machine** and returns to the main command prompt. If this command is thrown while running a string the following message will be displayed:

COMMAND: q

Terminating current machine

Quitting… String not accepted or rejected in 4 transitions.

5.2.11 Exit Application (X)

Terminates the Turing machine application when user enters the (X) command.

5.3 Termination

The application can be **terminated in two ways**:

5.3.1 Closing Turing Machine

- If the user **exits normally** using the **Exit (X)** command, the application shuts down cleanly. (Input string information is saved)
- If the application detects an **error** (e.g., invalid input file), it displays an error message and exits automatically while loading.

References:

Britannica, The Editors of Encyclopaedia. Turing machine. *Encyclopædia Britannica*. 2024. Available from: https://www.britannica.com/technology/Turing-machine

Alan Turing: Computer and LGBTIQ+ Trailblazer. 2024. *Australian Workplace Law*. Available from: https://awl.com/news/alan-turning-computer-lgbtiq-trailblazer/

Davey M. An interview with Mike Davey about his homemade Turing machine. *TechCrunch*. 2010 Apr 5. Available from: https://techcrunch.com/2010/04/05/an-interview-with-mike-davey-about-his-homemade-turing-machine/

Corrigan NB. Turing Machine Formal Specification Sheet. 2025. Provided directly by Dr. Niel B. Corrigan.

Martinez R. Prototype Menu for Turing Machine Application. 2025. Created by the author.