TMD-2 Quick Start Guide

Welcome to TMD-2 the state of the art in Turing teaching technology.

What is a Turing Machine?

The Turing machine was invented in 1936 by Alan Turing. By providing a mathematical description of a simple device capable of arbitrary computations, he was able to prove the properties of computation in general.

A Turing machine mathematically models a mechanical machine that operates on a tape. More explicitly, a Turing machine consists of:

- A tape divided into adjacent cells. Each cell contains a symbol from some finite alphabet. The alphabet contains a special blank symbol and one or more other symbols. Cells that have not been written before are assumed to be filled with the blank symbol.
- A head that can read and write symbols on the tape that can be moved left and right one (and only one) cell at a time.

- A state register that stores the current state of the Turing machine. There can be many states. Among these is the special start state with which the state register is initialized.
- A finite table of instructions that, given the state the machine is currently in and the symbol it is reading on the tape (symbol currently under the head), tells the machine to do the following transition steps in sequence:
  1. Write a symbol from the finite alphabet replacing the one that was there. Note that the symbol written might be the same as before or different
  2. Move the tape either one cell to the left or one cell to the right.
  3. Assume the same or a new state as prescribed by the go to state.

TMD-2

The Turing Machine Demonstrator (TMD-2) will have the following characteristics:

- One tape with 100,000 cells and a single head.
- The alphabet used will have seven symbols: {0, 1, 2, 3, 4, 5, b}. **0** will be the blank symbol and **b** is an endmarker symbol that can be read from the tape but not written.
- There will be six states: {A, B, C, D, E, F}. State **A** will be the start state plus there is a special HALT state **H**.

Working With TMD-2

Turing Machine Demonstrator Mark 2 (TMD-2) is an application that brings Turing's conceptual model to life. To start let's look at the various parts of the screen and find out what they do.

The Tape
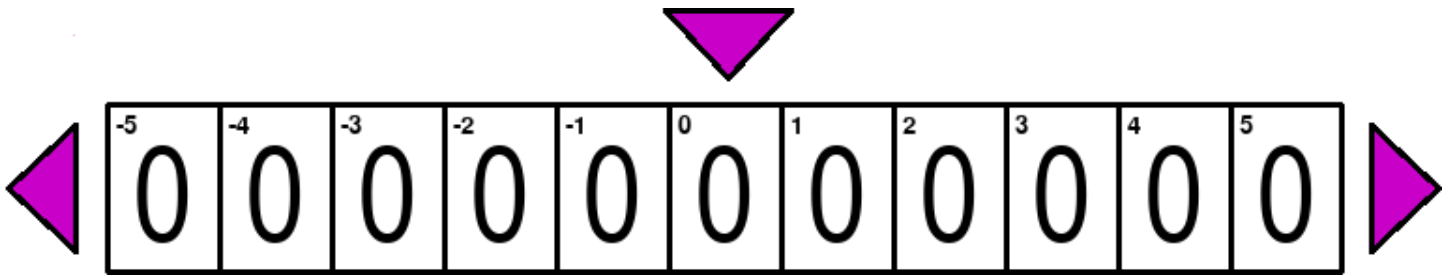
This part of the screen shows an eleven cell "window" into the Turing machine's tape. Think of the tape as being "underneath" this window. While the definition of a Turing machine usually refers to the tape as being infinite, for practical purposes the tape in this application is limited to 100,000 cells.

You can use the **tape controls** to manipulate the tape on screen.

Left: When clicked will move the tape under the window one cell to the left. The tape head will now be pointing at the cell that was to the immediate right prior to the move. You can also use the *Left Arrow* key to accomplish the same thing.	**Tape Head:** This symbol points to the only cell on the tape that can be read from and written to by the Turing machine. If the machine is not "running" you can click on this control to advance the cell value on the tape to the next allowable tape symbol.	**Right:** When clicked will move the tape under the window one cell to the right. The tape head will now be pointing at the cell that was to the immediate left prior to the move. You can also use the *Right Arrow* key to accomplish the same thing.

If the TMD-2 is not running (the Play button will not be green), in addition to the above controls, you can manipulate the visible tape cell symbols with your mouse. By clicking on the upper half of a tape cell, you will advance the cell value to the previous allowable tape symbol. Similarly clicking on the bottom half, you will advance the cell value to the next allowable tape symbol (just like clicking on the Tape Head symbol). You can also hover the mouse cursor over the tape cell that you want to change and use the scroll wheel.

By default, when you first launch the application, the Tape Head is centered on the underlying tape. So, there will be about 50,000 cells to each side of it. As you can see in the picture above the cells are numbered in increasing order from 0 to the right and decreasing order to the left of this central point.

The State Transition Table

Overview

	A						B						C					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

	D						E						F					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

This is the part of the screen where you "program" the Turing machine. As the name implies this table defines the transitions between states. The main grid is divided into six sections, one for each state, with state names **A** through **F** centered in top row of each.

	A					
READ	0	1	2	3	4	5
WRITE						
MOVE						
GOTO						

Within a state section there are columns for each of the symbols that can be read from the tape, **0** through **5**. Each column represents a single transition with the tape symbol it represents appearing in the READ row.

	A					
READ	0	1	2	3	4	5
WRITE						
MOVE						
GOTO						

A transition for a given tape symbol will proceed through the READ, WRITE, MOVE, and GOTO steps as indicated by the labels on the left.

Operation

You program TMD-2 by "filling in" the State Transition Table. Much like the Tape cells, if the TMD-2 is not running (the Play button will not be green), you can manipulate the visible State Transition Table cell symbols with your mouse. By clicking on the upper half of a state table cell, you will advance the cell value to the previous allowable cell symbol. Similarly clicking on the bottom half, you will advance the cell value to the next allowable symbol. You can also hover the mouse cursor over the state table cell that you want to alter and use the scroll wheel to change the value.

The allowable symbols for a cell vary by row:

- **READ**   – 5 b
- **WRITE** – 0 1 2 3 4 5
- **MOVE**   – L R
- **GOTO**   – A B C D E F H

In addition, all rows allow the space character so that a cell can be cleared if not required.

Endmarker Symbols

Having an endmarker symbol allows you to optionally implement a restricted form of Turing machine called a "linear bounded automaton" (LBA). Being able to determine the beginning and end of an input area can be especially useful for some applications.

In addition to the normal Turing machine behaviors, an LBA has the following three conditions:

- The input alphabet includes a special symbol **b** which can serve as a left and right endmarker.

- Transitions may not print other symbols over the endmarkers. In addition, TMD-2 will not allow an endmarker to be written to the other part of the tape (**b** is not part of the tape alphabet for WRITE).

- Transitions may neither move to the left of the left endmarker nor to the right of the right endmarker. For TMD-2 such a move will HALT the machine.
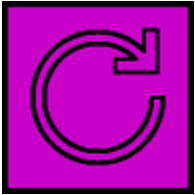
In the READ rows of the State Transition Table, the symbols **0** through **4** are fixed and cannot be changed. The **5** symbol can however be replaced with a **b**. If it is set to **b** the corresponding WRITE symbol is also set to **b** and cannot be changed since an endmarker cannot be written over.

We will see an example of setting up a program that does this a bit later.

Controls

When you mouse over any control, if they are available for use, they will be highlighted by changing to a lighter color. If they do not change color, they cannot be used in the current context.

In addition to the tape controls already mentioned, these controls are used to operate the state machine.

Reset: When clicked, the reset button will immediately clear the tape to zeros and set the tape head to point to the middle of the tape (cell 0). In addition, a popup dialog will ask you to decide if the state transition table should be cleared as well. Click YES to clear the table and NO leave the table with the current program.

Halt: This control is both a button and an indicator. When the state machine is running (play button green) you can click on halt to suspend execution. The halt control will turn red to indicate this. Similarly, if the executing state machine reaches a halt condition, either because a transition with the special H state was processed, or an error in execution was detected, execution is suspended and the halt control turns red.
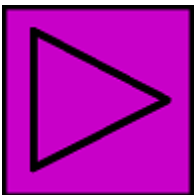
Play: Click this control to start the execution of the state machine, or to resume execution from a halt condition. When pressed the button will turn green indicating that the state machine is running.

Mode: These radio buttons control the state machine's running mode. If set to STEP, you must click the play button once for each step of each transition (READ, WRITE, MOVE, and GOTO). DEMO mode will execute these steps automatically at a rate of about one step per second. In both of these modes, the state transition table elements will be highlighted to indicate which state, transition, and step is currently being executed, and the tape will be updated with each transition. RUN mode will disable this highlighting and tape updates to improve performance. Only the halt button can be clicked when in RUN mode.

Example

In this screen shot we can see the following:

- The state machine is running since the play button is green.
- We are in STEP mode, so the application is waiting for play to be pressed again to execute the next step.
- The next step to be executed is a MOVE.
- The transition being executed is for the 1 symbol in the A state.
- We can see that a 1 has just been written to the tape.
- The tape will be shifted to the left one cell when the step is executed.

Saving Your Work

You can click on **SAVE** to write the current "workspace" to a file. You will be prompted to enter a file name to save under. The workspace saved will include the current contents of the tape, the state transition table values, and any other information required to resume work at the point that you saved. You could for instance halt a long running program in the middle of executing, then save. When you **LOAD** the workspace again you can resume running the program from where you halted by pressing the play button.

In addition to the workspace file, **SAVE** will also generate a dump. This is a plain text file that shows the tape, some statistics, and the state transition table at the point the workspace was saved. The result of a 3-state / 2-symbol busy beaver run looks like this:

```
Showing tape from cell -2 to cell 3.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| 1 | 1 | 1 | 1 | 1 | 1 |

Counts
~~~~~~
0: 0
1: 6
2: 0
3: 0
4: 0
5: 0
b: 0

State Transition Table
~~~~~~~~~~~~~~~~~~~~~~~~~
                A
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | - | - | - | - |
| R | L | - | - | - | - |
| B | C | - | - | - | - |

                B
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | - | - | - | - |
| L | R | - | - | - | - |
| A | B | - | - | - | - |

                C
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | - | - | - | - |
| L | L | - | - | - | - |
| B | H | - | - | - | - |

                D
| 0 | 1 | 2 | 3 | 4 | 5 |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| - | - | - | - | - | - |

                E
| 0 | 1 | 2 | 3 | 4 | 5 |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| - | - | - | - | - | - |

                F
| 0 | 1 | 2 | 3 | 4 | 5 |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
```
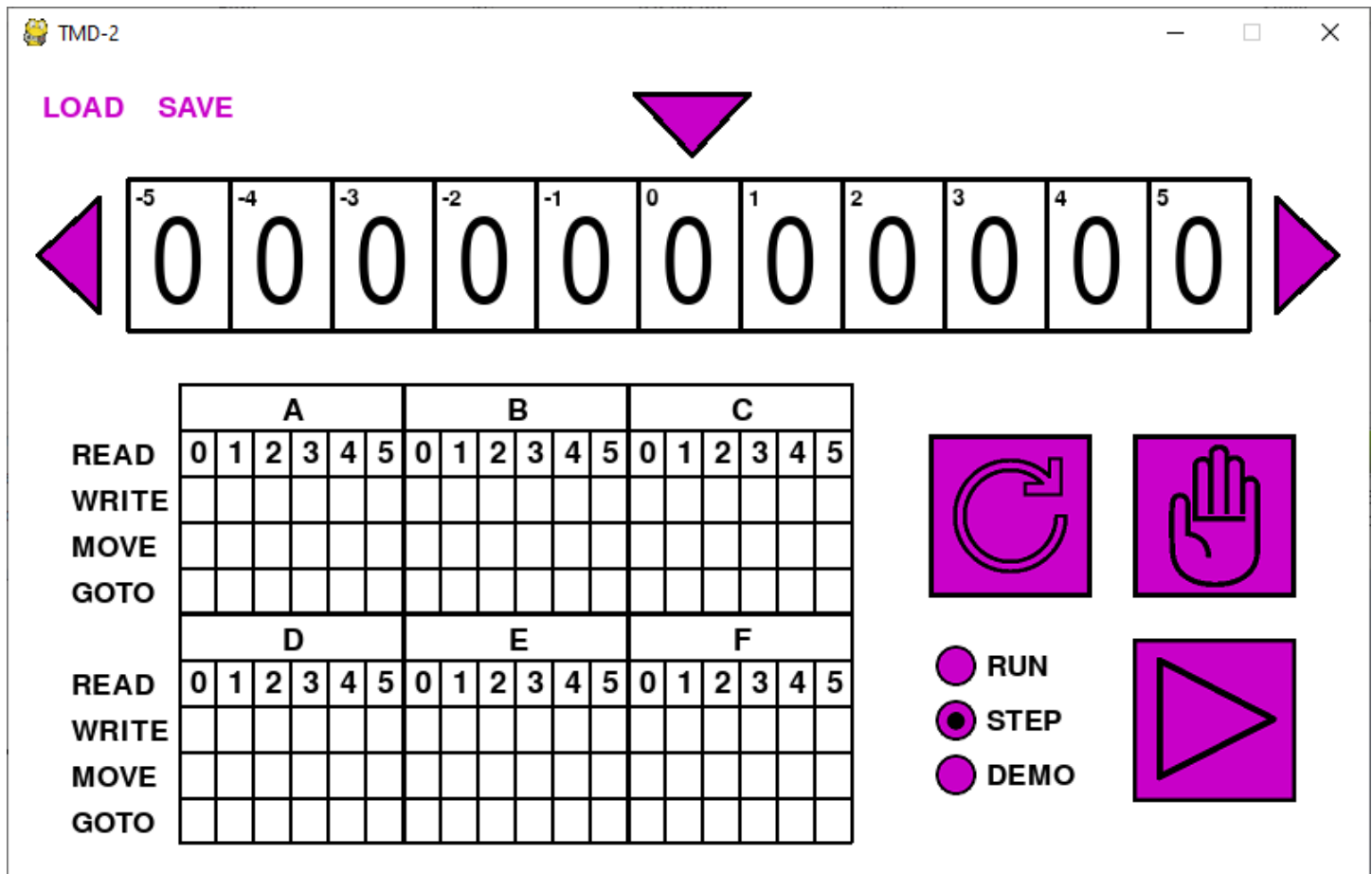
On System Start or Reset

When the application is first run or has just been Reset you should see:

This means that TMD-2 has been initialized with the following starting settings:

- All cells on the tape have been cleared to 0s.
- The tape has been centered and the tape head is pointing to cell 0.
- All the buttons and controls are set to the default color.

In other words, TMD-2 is ready to go.

Your First TMD-2 Program

With the "introductions" out of the way we are going to jump into the deep end and create our first Turing Machine Demonstrator program. The problem is simple:

> *For this problem we will limit ourselves to only two symbols 0 and 1. Write a program to invert all the symbols in a bounded input area. So, all 1s become 0s and vice-versa.*

Let's get started.

Setup

Before we can do start programming the problem, we must create a "bounded input area" as specified in the problem definition. This is accomplished by setting leftmost and rightmost **b** endmarkers on the tape. Use the mouse to change the symbols in the -5 and 5 cells of the tape to **b** as described in the **Tape** section above. Your screen should now look like this:

The Program

Programming TMD-2 is as simple and "filling in" the State Transition Table. We know that machine will be in the **A** state by default. Since we are limiting ourselves to the symbols **0** and **1**, and we know that the **b** symbols have been added to the tape, when a READ is executed, the symbol read could be any one of **1**, **0**, or **b**. Since we are in the **A** state, we will have to fill in all three transitions associated with those symbols and **A**.

But wait, there is no **b** symbol in the READ row of A state. Remember that the **5** symbol can be switched to **b**. Use the mouse to change the **5** symbol in the READ row of the **A** state to a **b** as described in the **State Transition Table - Operation** section above. Notice that the corresponding WRITE cell also changes to a **b**.

Our first task will be to move the tape head to one end of the input area without changing any of the tape values. We will know that the end is reached when we READ a **b** symbol on the tape. When the end of the input area is reached, we want to reverse direction and start inverting symbols. So, the first part of the program looks like this:

- When we read a **0:** write a **0**, move **Left**, stay in the **A** state.
- When we read a **1:** write a **1,** move **Left**, stay in the **A** state.
- When we read a **b:** write a **b,** move **Right**, switch to the **B** state to do inversions.

Now the next part of the problem is very similar. Move the tape head to the other end of the input area, inverting all of the tape values on the way. We will know that the other end is reached when we READ a **b** symbol on the tape. At that point, the problem is done so we can halt. The completed program looks like this:

- When we read a **0**: write a **1**, move **R**ight, stay in the **B** state.
- When we read a **1**: write a **0**, move **R**ight, stay in the **B** state.
- When we read a **b**: write a **b**, move **L**eft, halt the state machine.

So that's it. Our first program is finished, using only a two states to accomplish the task. But is the program correct? We should probably STEP through the first Transition to make sure everything is working as expected.

Slow and Steady

Using the STEP feature is a great way to debug a TMD-2 program. Set the mode radio button to STEP. Change some of the tape cells to **1** so that we can make sure that inversion is working as expected. Press the play button once:
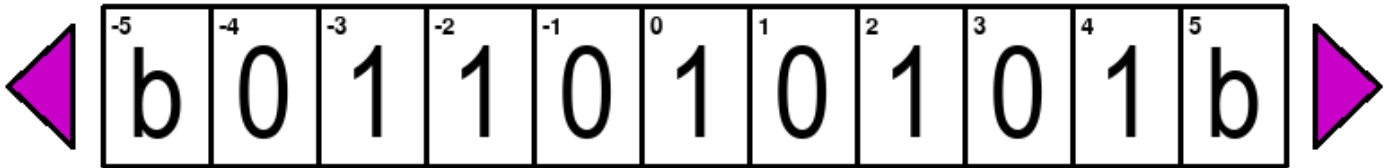

We know that the state machine is running because the play button is green. **A** is the active state, and we are about to perform the READ step. Press play again.

LOAD   SAVE

-5	-4	-3	-2	-1	0	1	2	3	4	5
b	0	1	1	0	1	0	1	0	1	b

	A						B						C					
READ	0	1	2	3	4	b	0	1	2	3	4	b	0	1	2	3	4	5
WRITE	0	1				b	1	0				b						
MOVE	L	L				R	R	R				L						
GOTO	A	A				B	B	B				H						

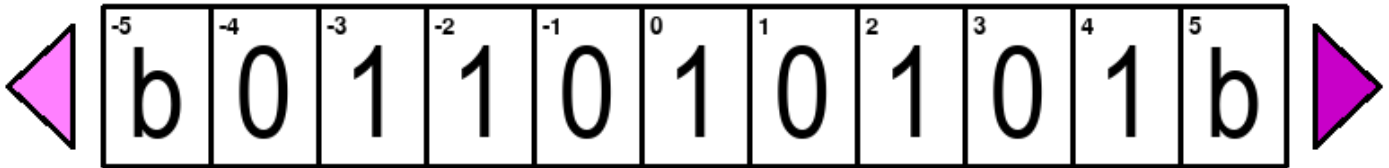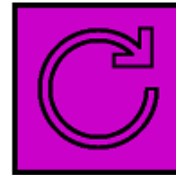|	D						E						F					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

○ RUN
◉ STEP
○ DEMO

We can see that the READ operation has selected the **A1** Transition to execute which is correct since we are in state 'A' and the Head is on a 1 cell. The Next Transition Step has been set to WRITE. Check! Press the play button again.

LOAD   SAVE

-5	-4	-3	-2	-1	0	1	2	3	4	5
b	0	1	1	0	1	0	1	0	1	b

	A						B						C					
READ	0	1	2	3	4	b	0	1	2	3	4	b	0	1	2	3	4	5
WRITE	0	1				b	1	0				b						
MOVE	L	L				R	R	R				L						
GOTO	A	A				B	B	B				H						

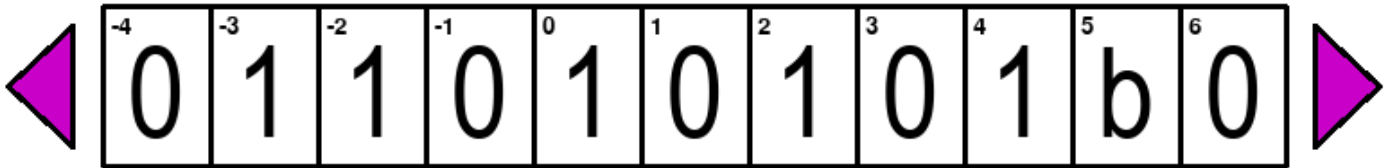|	D						E						F					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

○ RUN
◉ STEP
○ DEMO

WRITE was executed and the tape head stays a **1**. Next Transition Step set to MOVE. Check! Press the PLAY button again.

TMD-2

LOAD   SAVE

-4	-3	-2	-1	0	1	2	3	4	5	6
0	1	1	0	1	0	1	0	1	b	0

	A						B						C					
READ	0	1	2	3	4	b	0	1	2	3	4	b	0	1	2	3	4	5
WRITE	0	1				b	1	0				b						
MOVE	L	L				R	R	R				L						
GOTO	A	A				B	B	B				H						

	D						E						F					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

RUN
● STEP
DEMO

Next Transition Step set to GOTO. Tape has been correctly moved one cell to the left. Check! Press the PLAY button one more time.

	A						B						C					
READ	0	1	2	3	4	b	0	1	2	3	4	b	0	1	2	3	4	5
WRITE	0	1				b	1	0				b						
MOVE	L	L				R	R	R				L						
GOTO	A	A				B	B	B				H						

	D						E						F					
READ	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
WRITE																		
MOVE																		
GOTO																		

RUN
STEP
DEMO

GOTO has Transitioned back to the A state and is ready to READ the next cell. Check! All is looking good. Time to try the other modes.

Demoing

Click the DEMO radio button. Your program should continue to run at a rate of about one step per second. At this speed you should be still be able to follow what the state machine is doing. When you have seen enough click the halt button.

Running

At this point your program should be suspended. Click the RUN radio button and then click the play button to resume execution. The program will complete without any screen update. When done the halt button should be red and the tape values inverted. I shifted the tape so that the 0 cell was centered under the head and this is my result.

Your Next Program

Hopefully, I've convinced you how easy it is to write a TMD-2 program. A good next step might be to extend the inversion program we just wrote.

> *If you consider the input area with its 0s and 1s to be a binary number, then what the inversion program does is to create what is called the 1s compliment of that number. The 2s compliment of a binary number is just the 1s compliment of the number to which a 1 is added (mathematically).*

You have 4 whole unused states to add the 2s compliment functionality. Are you up to the challenge?

More Challenges

Here are a few more "starter" problems that you can try. Like the 1s an 2s compliment examples just presented they are all designed to be run in a **bounded input area** using only the symbols **0** and **1**.

- *Counting (in binary). Working from the rightmost endmarker, starting with 1, continually replace the value with the next highest binary number (1, 10, 11, 100, etc.)*
- *Sorting. Move all the 1's in the input area to the right or left.*
- *Shift the input area one cell to the right or left (multiply / divide by 2)*

Busy Beavers – The Ultimate Challenge

Another fun thing to do with Turing machines is to create busy beaver programs. The busy beaver "game" consists of designing a halting, binary-alphabet Turing machine which writes the most 1s on the tape, using only a given set of states. By definition, a busy beaver program running on TMD-2 is prohibited from using the endmarker symbol.

You will find busy beaver programs for 3, 4, and 5 state machines as part of this program distribution. They can be loaded using the file names beaver3, beaver4, and beaver5. Running these programs will yield the following results:

	3-State	4-State	5-State
Number of 1s	6	13	4,098
Number of Steps	21	107	47,176,870

The number of steps for the 5-State machine is not a mistake. As the number of states increases, the complexity (of which the number of steps is an indicator), increases exponentially. **And here's the thing.** While the number of 1s and steps for the 3-State and 4-State machines are known to be optimal, no one has proven that the 5-State machine presented here is the best that can be achieved. So, the ultimate challenge to you is to create a better 5-State busy beaver. Good luck!