PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

pdst.ie

An Roinn Oideachais
Department of Education

# National Workshop 7

Session 5

# Session Schedule

| | |
|---|---|
| Section 1 | Introduction to Turing machines |
| Section 2 | Group activity on Turing machines |
| Section 3 | Artificial Intelligence |

# By the end of this session participants will have

- developed their understanding of **Turing machines** and understand their significance
- participated in an activity to develop a deeper understanding of how Turing machines operate
- participated in an activity on **artificial intelligence**

PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

# Section 1

Introduction to Turing machines

# A fundamental question of Computer Science…

*What is computable?*

*How do we define computability?*

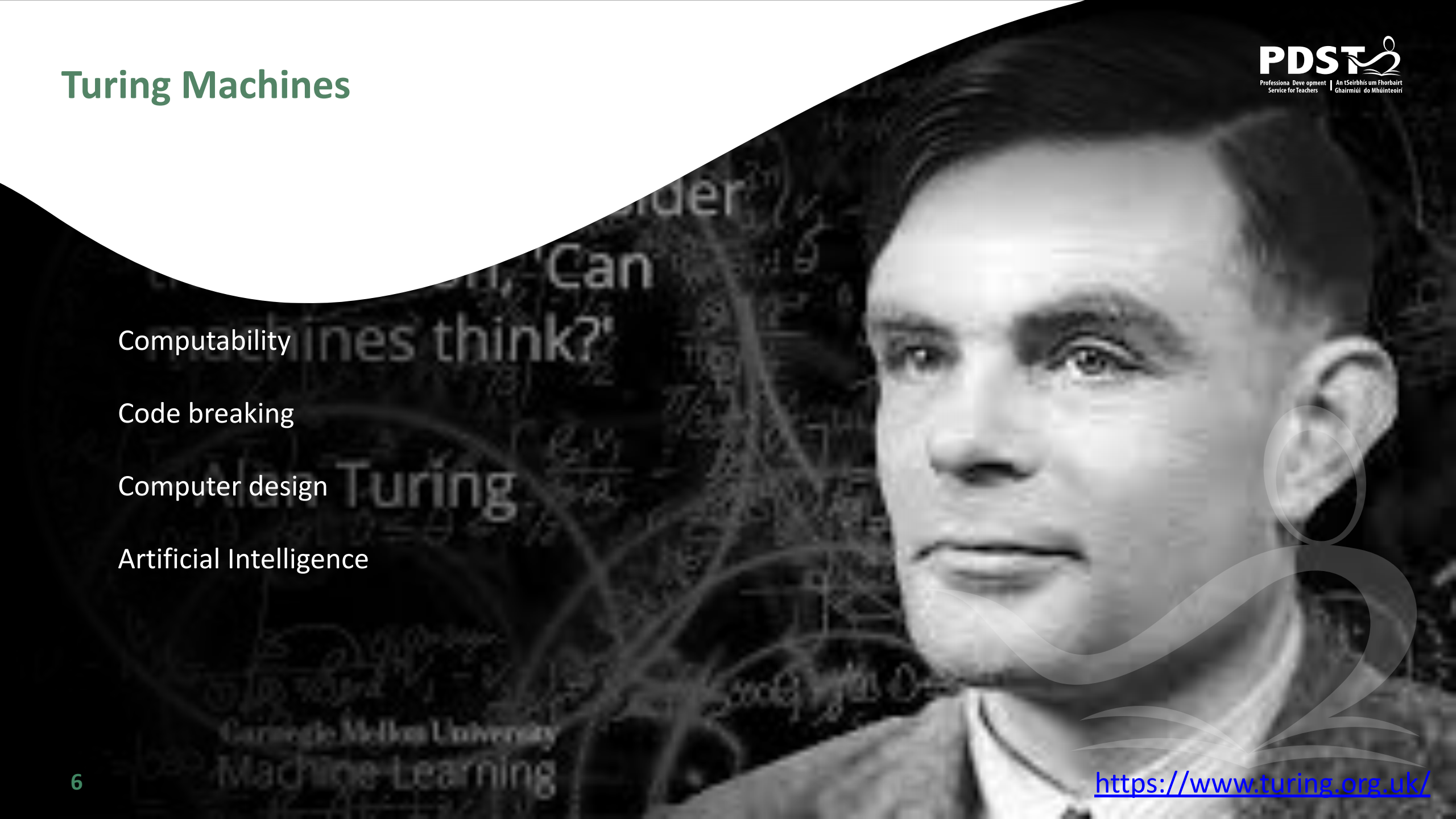
# Turing Machines

Computability

Code breaking

Computer design

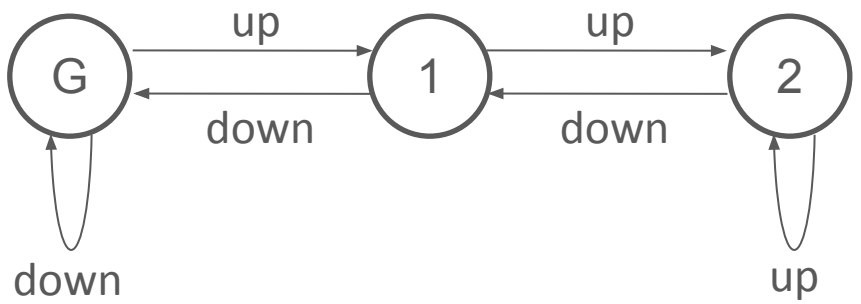
Artificial Intelligence




https://www.turing.org.uk/

# Introducing Turing machines

**The illustration below is of an elevator represented as a finite-state machine**




Circles represent states (in this case floors).

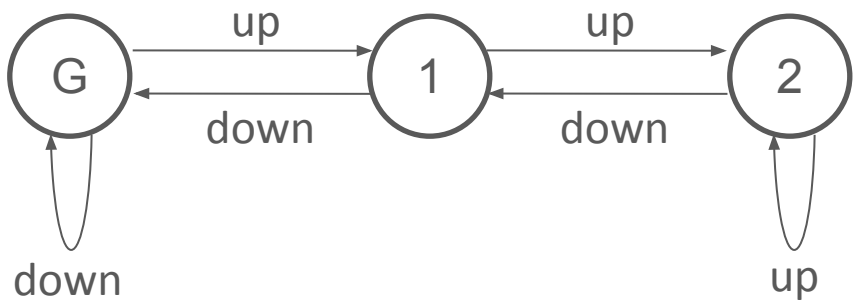
Arrows between circles represent transitions between states.

The labels on each transition represents the button press event.

What happens when we are on the ground floor and press the **UP** button?
What happens when we are on the ground floor and press the **DOWN** button?

# Introducing Turing machines

**The illustration below is of an elevator represented as a finite-state machine**



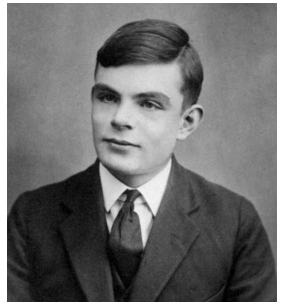

Assuming we start in state 1, what would the following input sequence yield?

**UU DDDD UU D**

The notion of a Turing machine is not too unlike this…
Given a Finite State Machine and an input, we can determine an output

# Introducing Turing machines

# Introducing Turing Machines

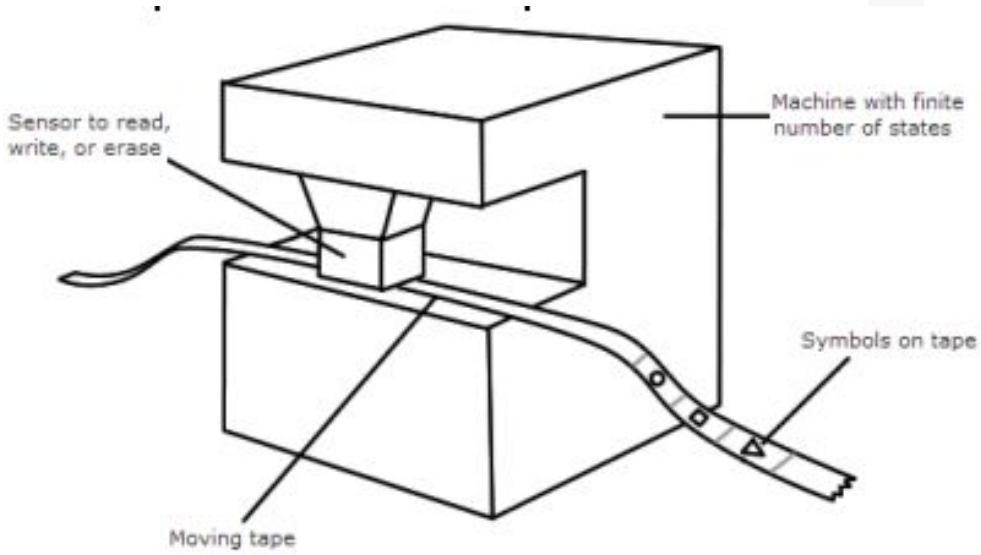


The Turing Machine (TM) was invented in 1936 by Alan Turing.

It is a basic abstract symbol-manipulating device that can be used to simulate the logic of any computer that could possibly be constructed.

Although it was not actually constructed by Turing, its theory yielded many insights.

*Anything that is possible to (mathematically) compute could be programmed on a Turing machine.*



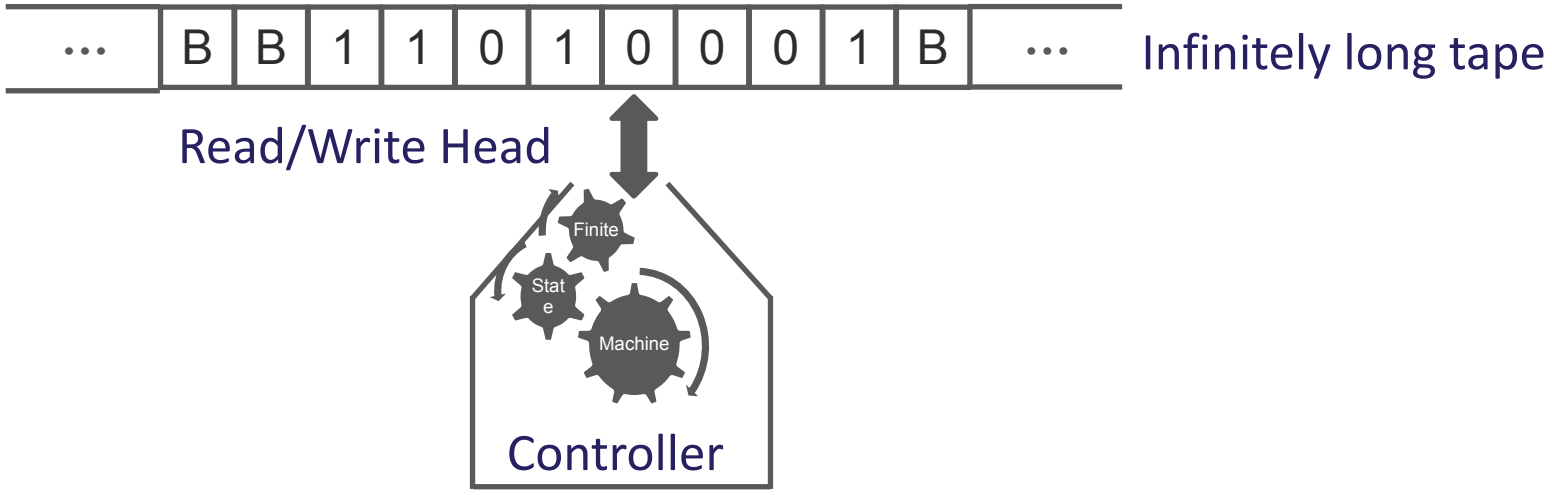
# Turing Machines - Introduction

A Turing Machine consist of three components as follows:

1. An infinitely long tape made up of individual cells. Each cell can contain a single character – typically 1, 0, or B (blank)

2. A read/write head pointed at an individual cell

3. A controller (aka finite-state machine) which instructs the read/write head what to do




**A schematic representation of a Turing Machine**

# Turing Machines - Operation

Initially the tape is inscribed with a sequence of characters – called the input

**For example:**

| ... | 1 | 0 | 1 | 0 | 0 | 1 | ... |
|---|---|---|---|---|---|---|---|

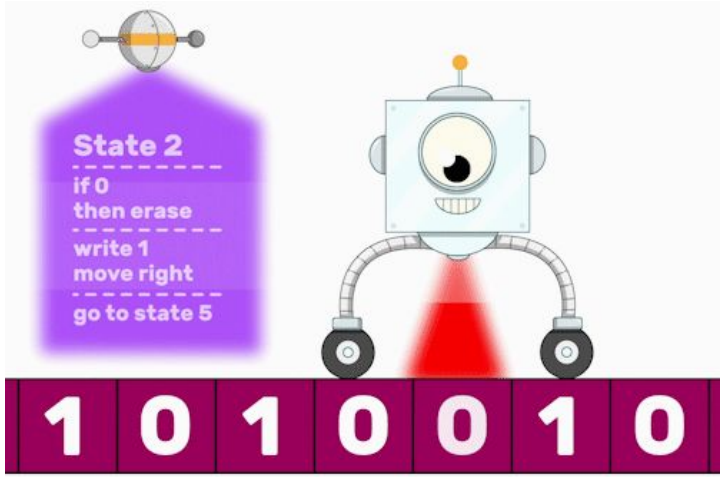
The operation of the Turing Machine is controlled by the finite-state machine (controller).

The operation takes place as a sequence of steps known as transitions.

The controller decides for a given (input character, state) pair, the (output character, state) pair - know as a transition.

Each transition involves:

- Reading
- Writing
- Moving
- Updating (state)

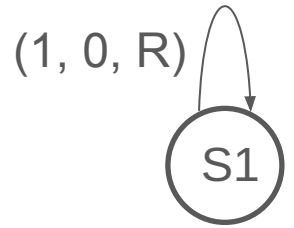


# Turing Machines - Operation

Transitions can be expressed using:

state transition tables OR state transition diagrams

| Current State | Read (input) | Write (output) | Direction to move r/w head | Next State |
|---|---|---|---|---|
| S1 | 1 | 0 | Right | S1 |



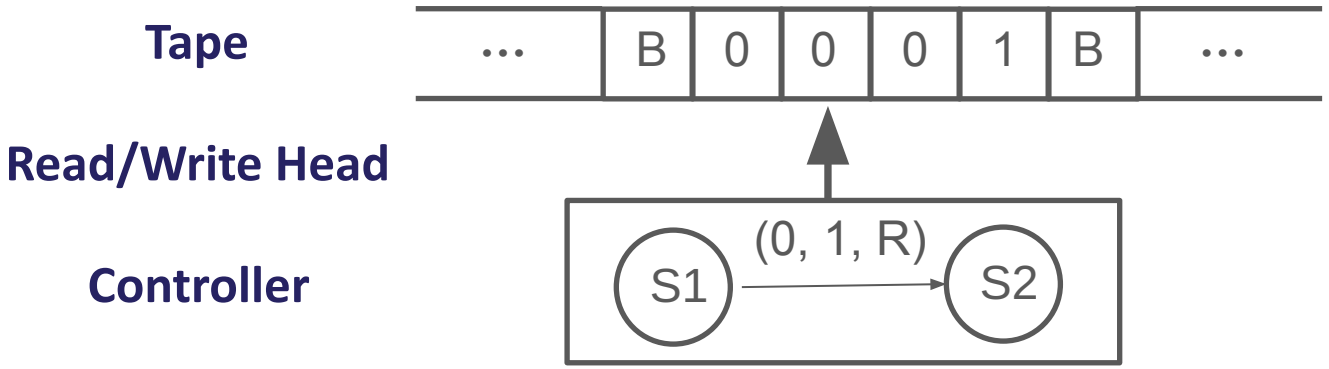

The above state transition table and diagram shows a single transition which says:

***"When in state S1 <u>and</u> the symbol being read is a one, write a zero, move right and remain in state S1".***

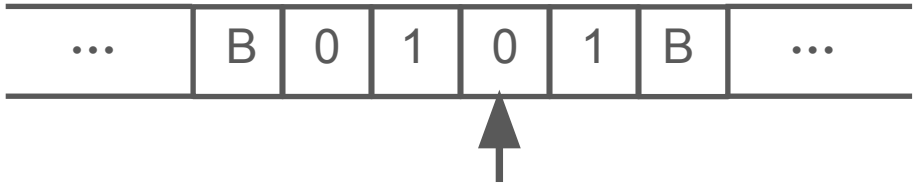
```
if (state == S1) and (character == 1):
   write 0
   move right
   set state to S1
```

# Turing Machines - Operation

The illustration below depicts a TM which defines a transition from state S1 to S2 when the current symbol being read in a zero.




After the transition has been completed, the symbol zero has been replaced with a 1, the read/write head has been moved right and the new state is set to S2.




The result of the computation (output) is the sequence of characters left on the tape if and when the Turing Machine halts.

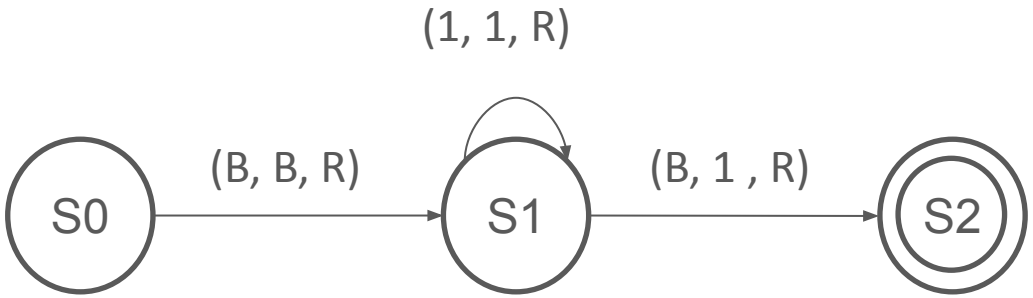
# Turing Machines – States

At any given time, a Turing Machine is said to be in a particular state. States are usually denoted by the letter S followed by a number e.g. S2 is taken to mean state two.

S0 is conventionally used to denote the initial state. This is the state the Turing Machine is in before it starts to operate.

A double circle is used to denote the final or *halting state*. This is the state the Turing Machine is in when it finishes.

For example:



# A fundamental question of Computer Science…

*What is computable?*

*How do we define computability?*

***Answer:*** *A task is computable if it can be carried out by a Turing Machine.*

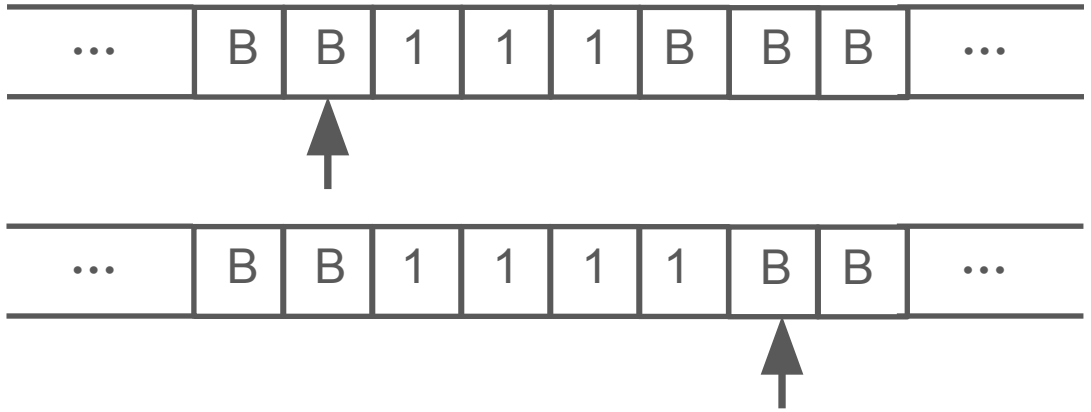
PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

# Section 2

Group activity: Turing machines

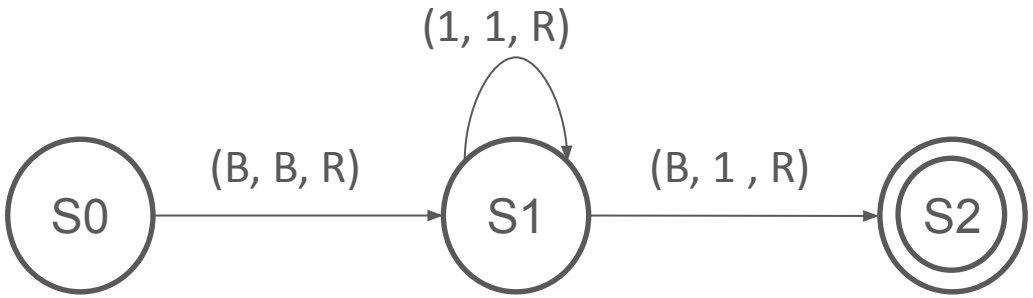
# Turing Machines – Example (unary increment)

**Test input: 111 (3)**



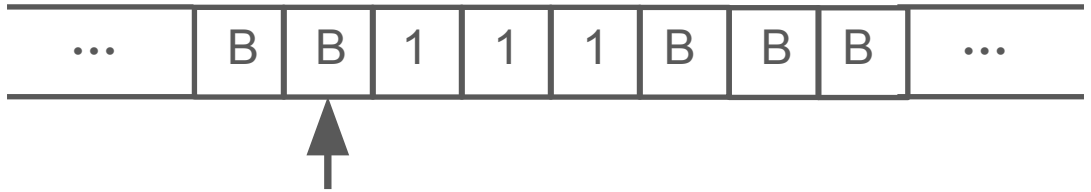

**Required output: 1111 (4)**



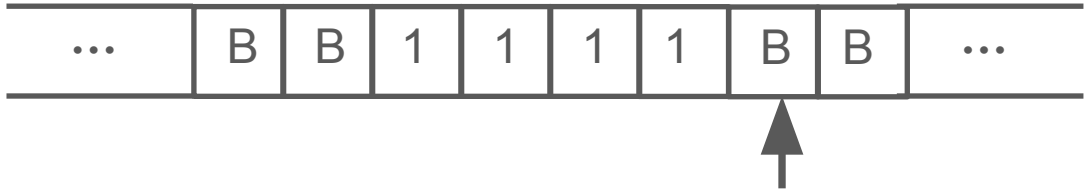



# Turing Machines – Example (unary increment)

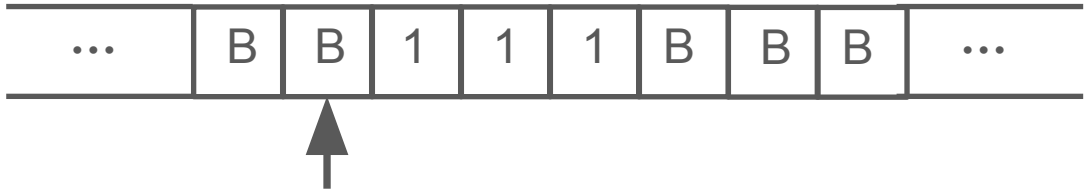
**Test input: 111 (3)**

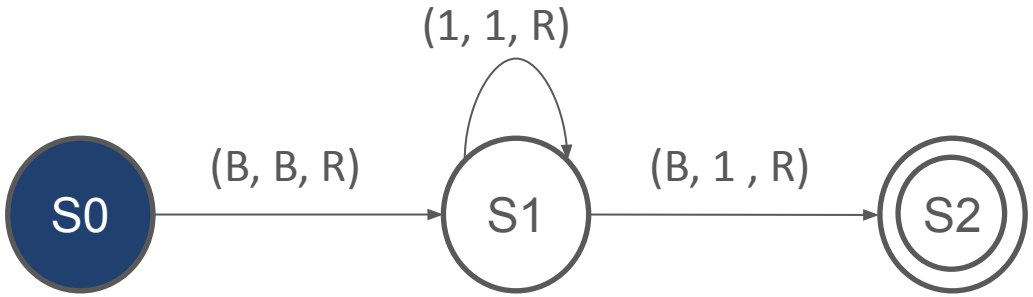



**Required output: 1111 (4)**



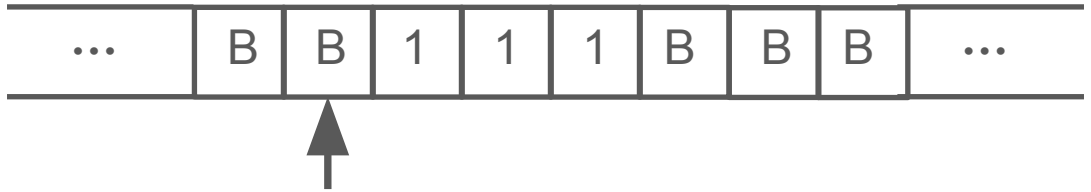

**State: S0**



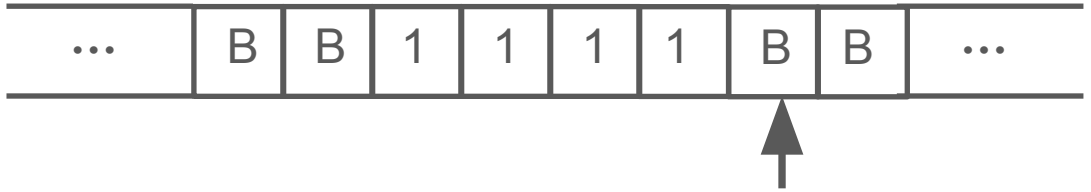



# Turing Machines – Example (unary increment)

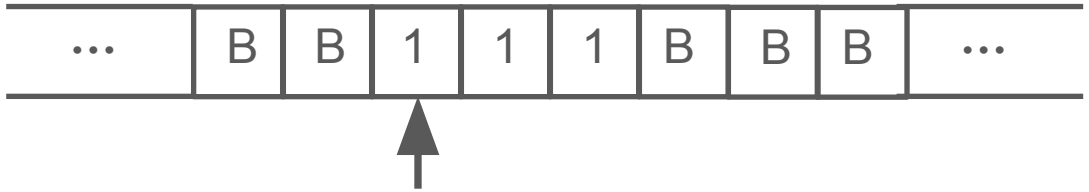
**Test input: 111 (3)**

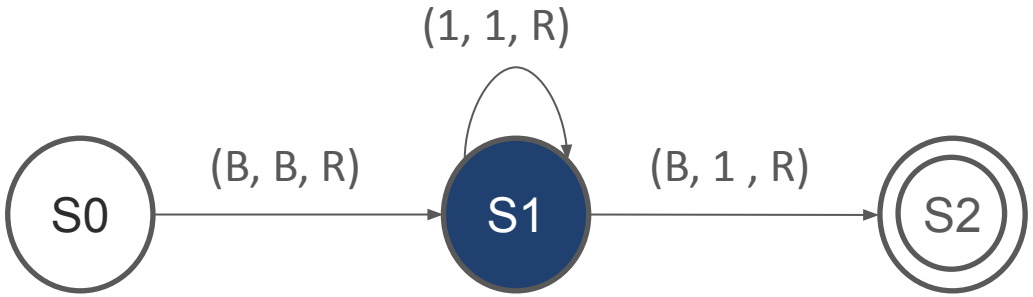



**Required output: 1111 (4)**



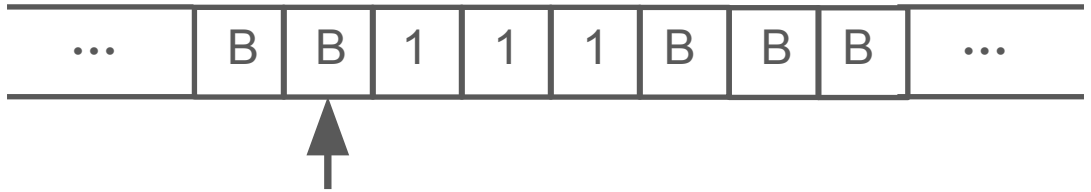

**State: S1**



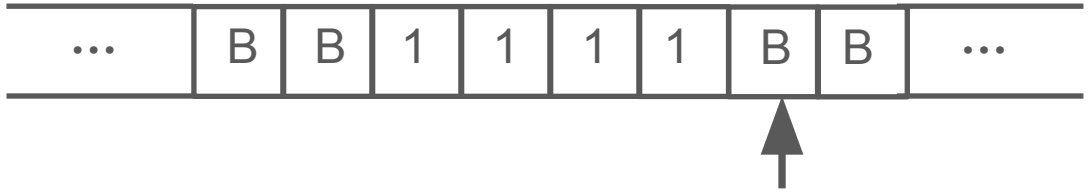



# Turing Machines – Example (unary increment)

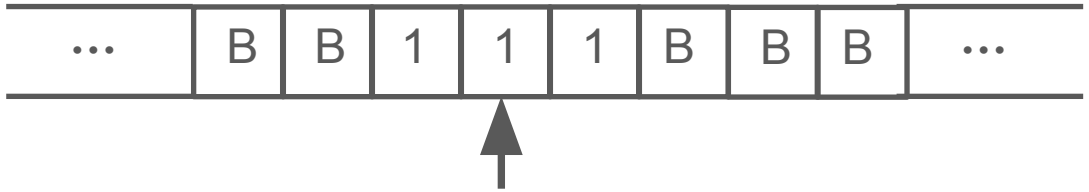
**Test input: 111 (3)**

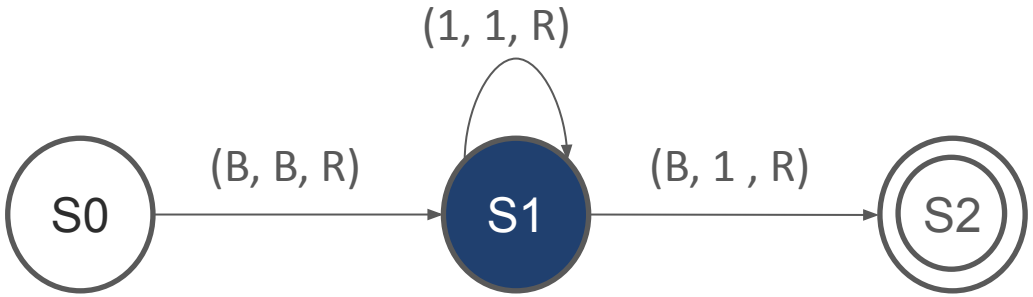



**Required output: 1111 (4)**



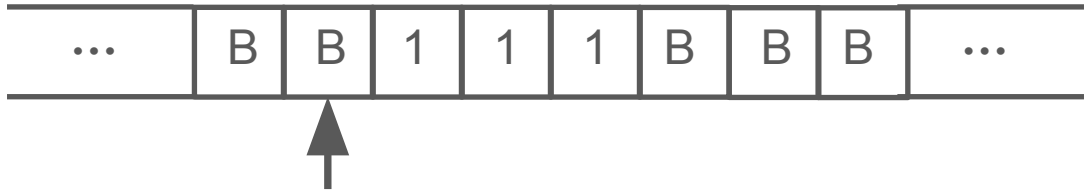

**State: S1**



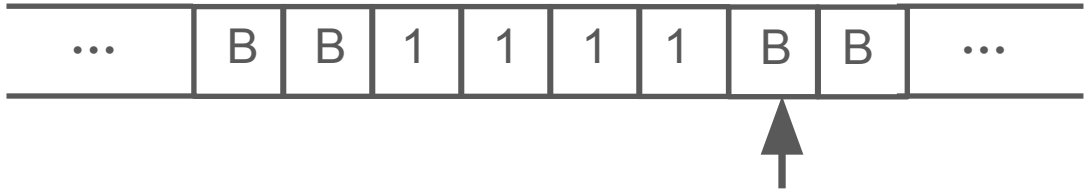



# Turing Machines – Example (unary increment)

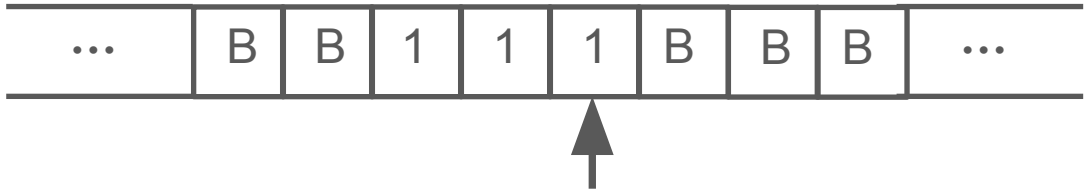
**Test input: 111 (3)**

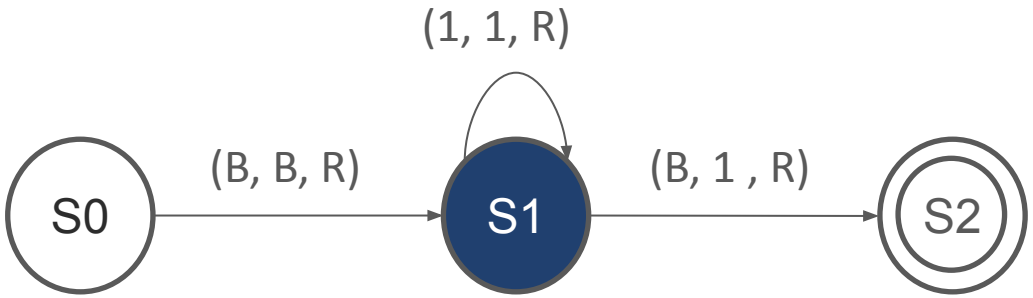



**Required output: 1111 (4)**



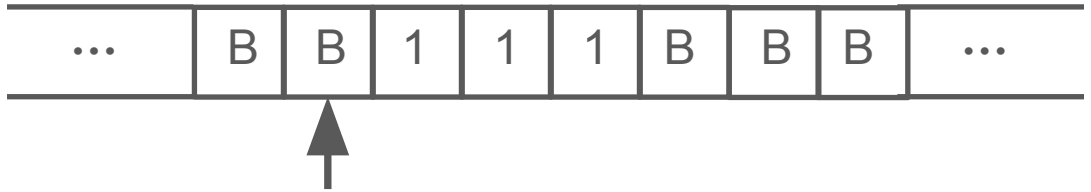

**State: S1**



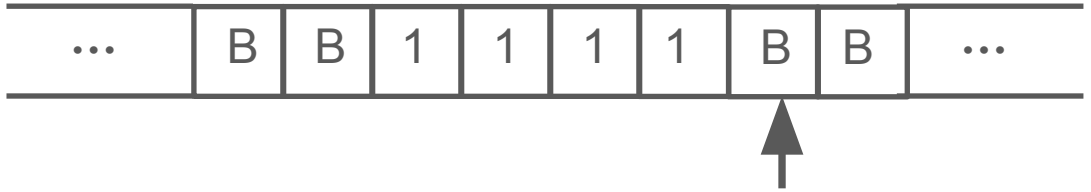



# Turing Machines – Example (unary increment)

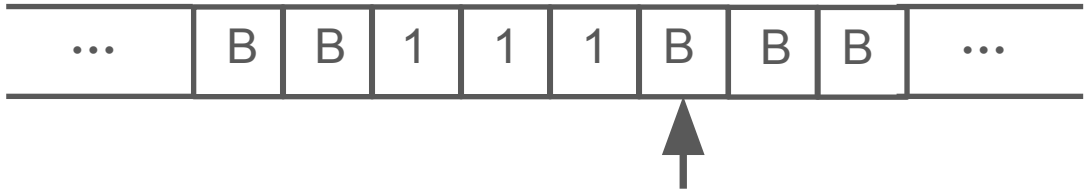
**Test input: 111 (3)**

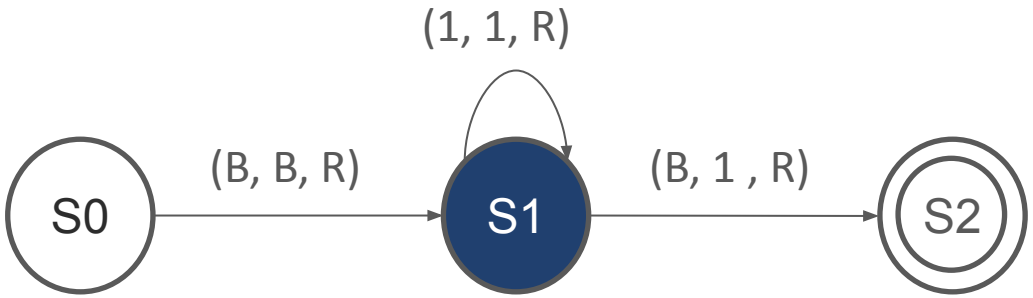



**Required output: 1111 (4)**



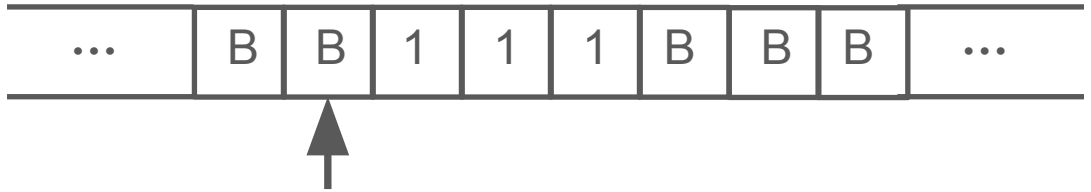

**State: S1**



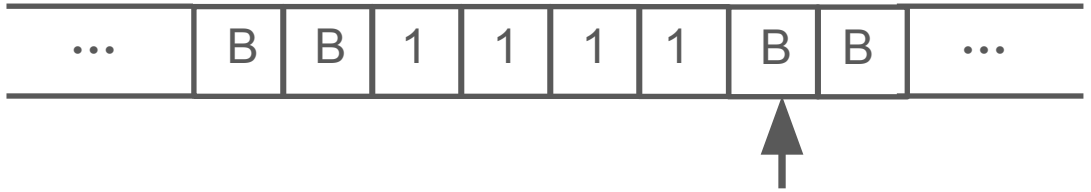



# Turing Machines – Example (unary increment)

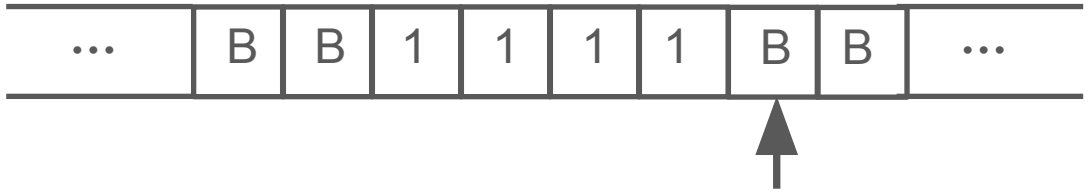
**Test input: 111 (3)**

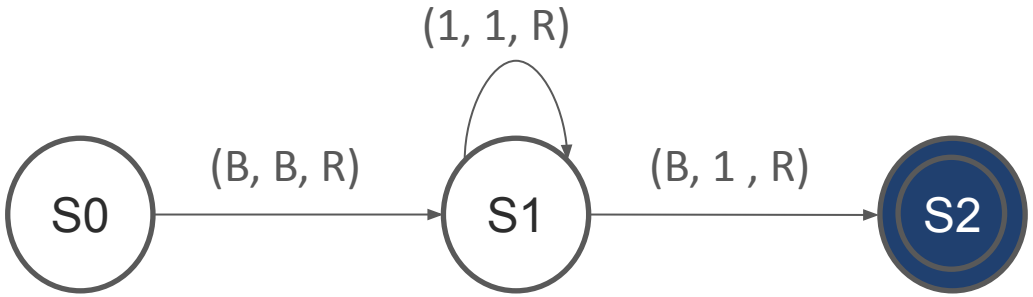



**Required output: 1111 (4)**




**State: S2**



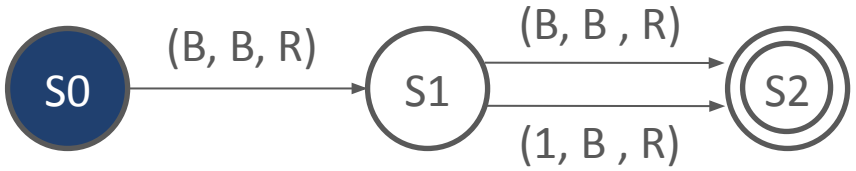



# Turing Machine Activity

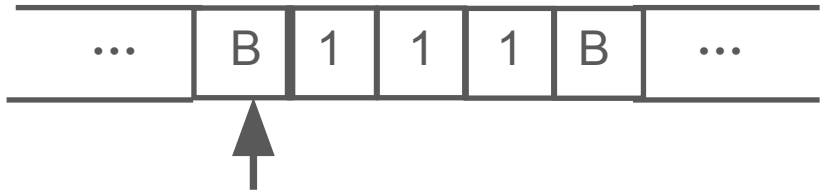
Each group will trace through the operation of the Turing Machines assigned.

# Turing Machines – Activity – Problem #1

Initial State: S0

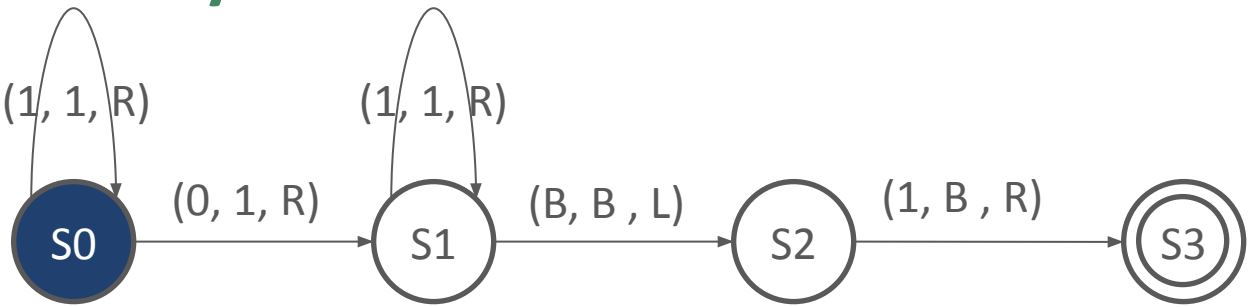



**Test input: B111B**

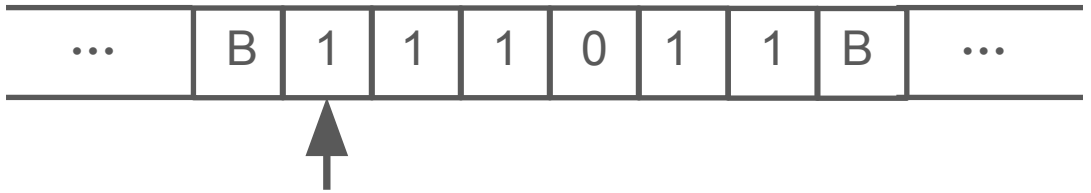


# Turing Machines – Activity – Problem #2

Initial State: S0

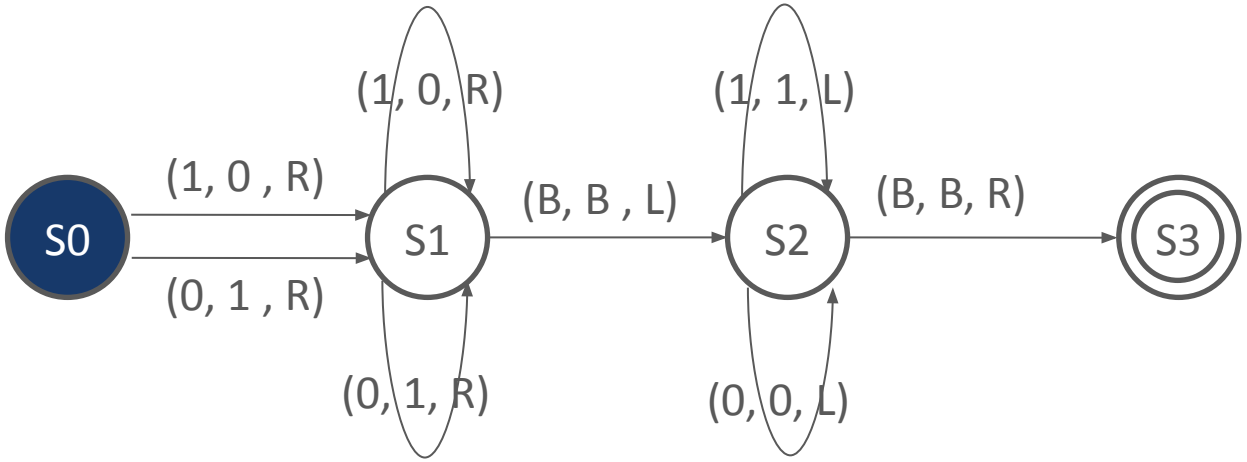



**Test input: 111011**

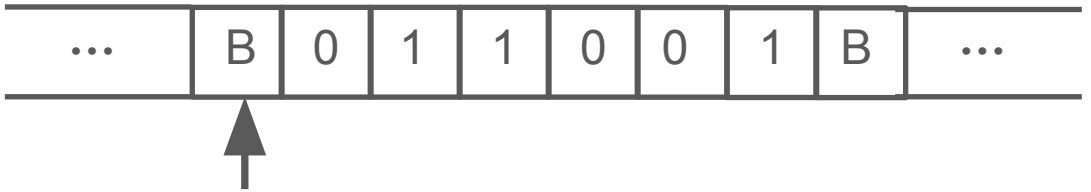


# Turing Machines – Activity – Problem #3

Initial State: S0

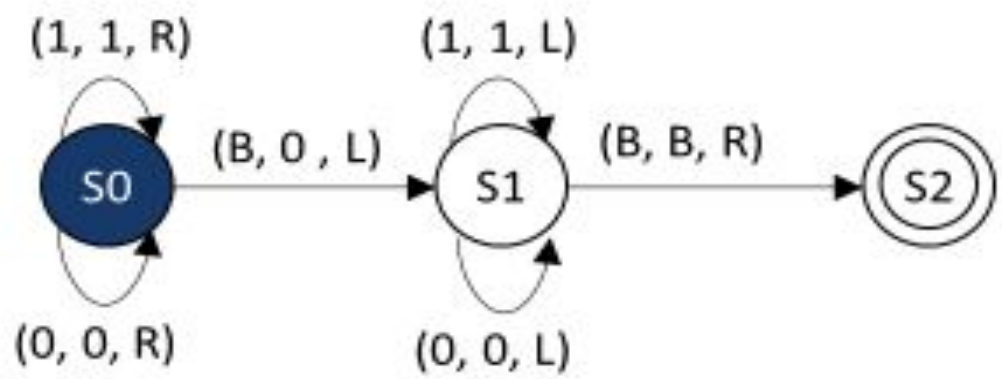



**Test input: B011001B**

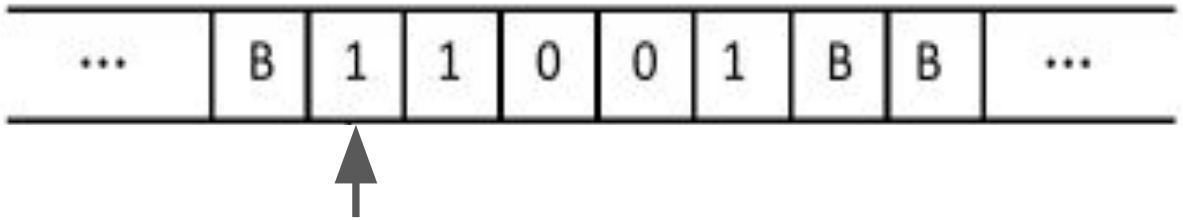


# Turing Machines – Activity – Problem #4

Initial State: S0



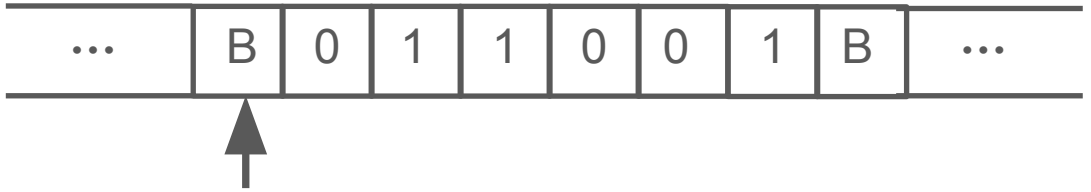

**Test input: B11001BB**



# Turing Machines – Activity Handout

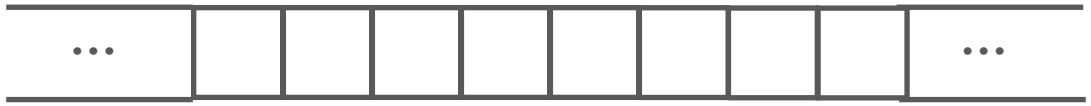
Initial Input

Initial State: S0



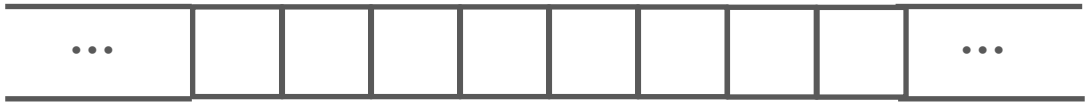

After Step 1:

Next State:




After Step 2:

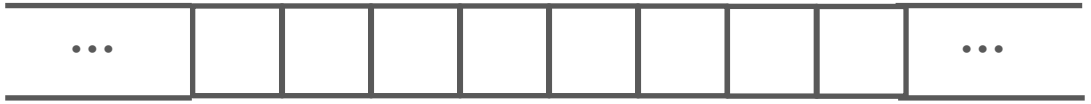
Next State:




...

...

Final Output



# Turing Machine Activity

Trace through the operation of the Turing Machines assigned.

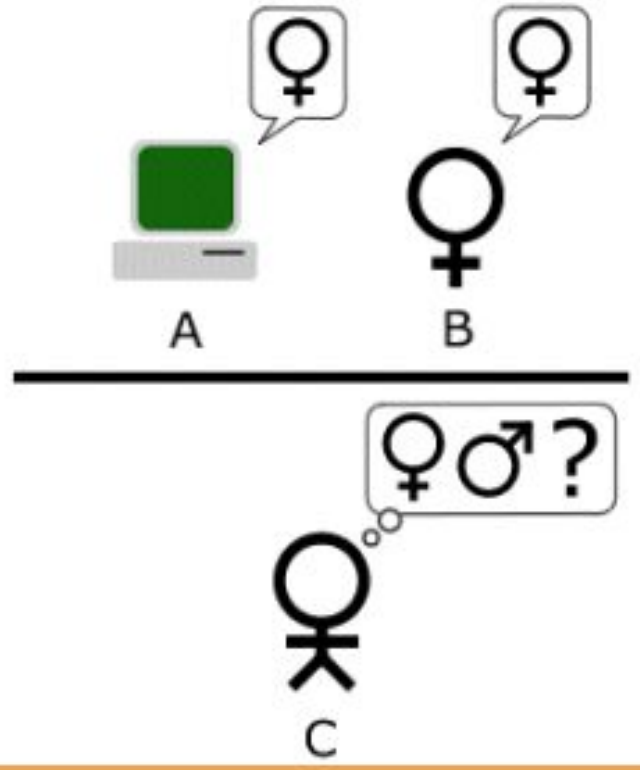
PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

# Section 3

Artificial Intelligence

# The Turing Test

The Turing test of artificial intelligence proposes a simple game where a hidden computer A and a person B converse with another person C. If C is unable to distinguish which he is conversing with, then the computer can be said to be able to "think".






Source: https://www.storyofmathematics.com/20th_turing.html

# A Test

# Lunch

An Roinn Oideachais
Department of Education