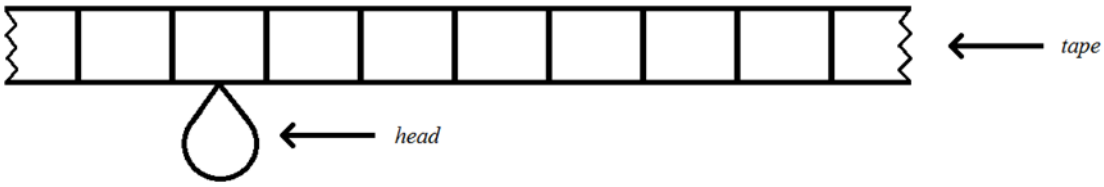
(K) The Dualization Game (1/5)

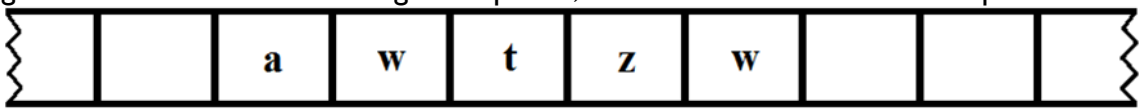
Turing Machines are a type of abstract computing machine first described by Alan Turing[1] in 1936. Although they have a very simple design, Turing Machines are very powerful – in fact, every computational task that a modern computer is capable of can also (theoretically) be done by a Turing Machine.



Turing Machines consist of a tape (a series of cells, infinite in both directions, each containing a blank or a symbol), and a head, which reads a particular cell on the tape and performs an operation according to the 'state' it is in: either writing something in that cell, moving left or right, both of these, or neither. A Turing Machine is defined by its instructions, which determine what operations it performs. Below are the instructions for a particular Turing Machine. Note that the symbol Ø indicates a blank on the tape; it is not the zero in state names (0):

Entry state	Read	Write	Move	Exit State
S0	w	Ø	R	S0
S0	Ø	[N/A]	[N/A]	HALT
S0	[otherwise]	[N/A]	R	S0

This Turing Machine deletes all w's on a given tape. So, if the machine were fed this tape:



…the tape would look like this when the machine was finished:

This transformation can be summarized as:

awtzw ⇒ atz (blanks within the letter sequence are not transcribed)

Some things to note:

- Turing Machines always start on the leftmost non-blank space on the tape
- Input tapes always contain a single string of symbols, unbroken by blanks
- A Turing Machine will only stop if it arrives at a HALT state or if there are insufficient instructions to proceed
- The initial state of a Turing Machine is always S0

Turing Machines can operate on strings of 0's and 1's, on arbitrary strings of letters (like above), or on words – in this last case, Turing Machines can be used to perform useful linguistic tasks.

[1]Alan Turing (1912-1954) was a British mathematician and logician who played a crucial role in the foundation of the field of computer science. He was the subject of the recent biopic 'The Imitation Game'.

(K) The Dualization Game (2/5)

Here is a simple Turing Machine designed for the English language, called Pluralizing Machine 1.0:

Entry state	Read	Write	Move	Exit State
S0	∅	s	[N/A]	HALT
S0	[otherwise]	[N/A]	R	S0

This machine makes the following successful (i.e., linguistically valid) transformations:

cat ⇒ cats apple ⇒ apples microscope ⇒ microscopes

However, this machine also makes the following unsuccessful (i.e., linguistically invalid) transformation:

* fox ⇒ foxs (* indicates an unsuccessful transformation)

The machines shown so far have used only a single state, S0 (not including the HALT state). Turing Machines that perform more complex tasks, however, will require multiple states, each with its own set of instructions. In multi-state machines, some lines of instructions will cause the machine to change state – in other words, exit the line in a different state than it entered.

Consider Pluralizing Machine 2.0:

Entry state	Read	Write	Move	Exit State
S0	∅	[N/A]	L	S1
S0	[otherwise]	[N/A]	R	S0
S1	x, s, z	[N/A]	R	S2
S1	[otherwise]	[N/A]	R	S3
S2	[otherwise]	e	R	S3
S3	[otherwise]	s	[N/A]	HALT

In addition to making the successful transformations made by Pluralizing Machine 1.0, this machine also makes successful transformations for many new words, including 'fox':

fox ⇒ foxes

K1. Give three more English words for which Pluralizing Machine 2.0 makes successful transformations, but that Pluralizing Machine 1.0 transforms unsuccessfully. Try to take advantage of all the added capabilities of the new machine.

K2. Pluralizing Machine 2.0 is not without its faults: what outputs does this machine give for the inputs 'quiz' and 'child'?

quiz: child:

Of course, Turing Machines can deal with any written language – not just English. The remaining Turing Machines in this problem perform tasks in Navajo – a language in the Na-Dené family, spoken primarily in Arizona, Utah, and New Mexico. With almost 170,000 speakers, Navajo is the most widely-spoken Indigenous language in the United States.

(K) The Dualization Game (3/5)

Consider the following verb forms from the Navajo language. Note: ł, ', and y are consonants in Navajo. An accent above a vowel, as in é, indicates high tone, pronounced with raised pitch. A hook beneath a vowel, as in ą, indicates that the vowel is nasal, pronounced through the mouth and nose. In the eyes of a Turing Machine, vowels that differ in tone or nasality are entirely different symbols.

Navajo	English
nidaahné	you (pl.) play
dajidlą	people (pl.) drink it
biyadahodiilyéés	we (pl.) frighten him
áchądahídéelni'	they (pl.) are greedy
bidajil'į	people (pl.) imitate him
nidaniiché	we (pl.) are on the run

Navajo	English
naahné	you (du.) play
jidlą	people (du.) drink it
biyahodiilyéés	we (du.) frighten him
áchąhídéelni'	they (du.) are greedy
bijil'į	people (du.) imitate him
naniiché	we (du.) are on the run

Sam designs a Turing Machine to transform the plural (pl.) form of a Navajo verb into its dual (du.) form. A dual verb has exactly two people/entities as its subject, and in Navajo this form contrasts with singular verbs (one person as subject) and plural verbs (three or more people as subject). Here is Sam's Dualizer Machine 1.0:

Entry state	Read	Write	Move	Exit State
S0	d	∅	R	S1
S0	[otherwise]	[N/A]	R	S0
S1	[otherwise]	∅	[N/A]	HALT

K3. Sam's Dualizer Machine 1.0 makes successful transformations for only four of the six plural verbs given above. Identify the other two, for which the machine makes unsuccessful transformations, and show the machine's output.

Verb: _____ Output: _____

Verb: _____ Output: _____

(K) The Dualization Game (4/5)

Here is the outline of Dualizer Machine 2.0:

Entry state	Read	Write	Move	Exit State
S0	d	∅	R	S1
S0	[otherwise]	[N/A]	R	S0
S1	[otherwise]			S2
S2		[N/A]		S3
S3	i	[N/A]		S4
S3			[N/A]	HALT
S4		[N/A]		S5
S4	[otherwise]	[N/A]	[N/A]	HALT
S5	[otherwise]		[N/A]	HALT

K4. Fill in the blanks of Dualizer Machine 2.0. The machine should make successful transformations for all six of the verbs given above.

Next, consider this set of Navajo verbs, with their English translation

Navajo	English
dádi'nishkaad	I sew it shut
ná'iishgááh	I bleach it
nistséés	I extinguish it
yishchxǫǫh	I destroy it
yishdééh	I scrape it off
hadishbin	I fill it up
ííníshta'	I read
ałk'íisgis	I entwine them
yiists'ił	I break it
yishhį́į́h	I melt it
iishchíih	I dye it red

Navajo	English
dádi'níłkaad	you (sg.) sew it shut
ná'iiłgááh	you (sg.) bleach it
níłtséés	you (sg.) extinguish it
niłchxǫǫh	you (sg.) destroy it
niłdééh	you (sg.) scrape it off
hadíłbin	you (sg.) fill it up
ííníłta'	you (sg.) read
ałk'íiłgis	you (sg.) entwine them
yiiłts'ił	you (sg.) break it
niłhį́į́h	you (sg.) melt it
iiłchíih	you (sg.) dye it red

Sam intends to design a Second-Personizer Machine that will transform the "I" form of each verb (the first-person singular form) to the "you (sg.)" (second-person singular) form.

(K) The Dualization Game (5/5)

Here is an outline of Sam's Second-Personizer Machine:

Entry state	Read	Write	Move	Exit State
S0				S1
S0	[otherwise]	[N/A]	R	S0
S1		ø		S2
S1	[otherwise]		L	S2
S2	[otherwise]			S3
S3		[N/A]	[N/A]	HALT
S3	[otherwise]		L	S4
S4		[N/A]	[N/A]	HALT
S4			[N/A]	HALT
S4	[otherwise]	[N/A]		S5
S5	[otherwise]		[N/A]	HALT

K5. Fill in the blanks of Sam's Second-Personizer Machine. This Machine should be able to successfully transform every verb above from its first-person form to its second-person form.

Here are two last Navajo verbs, with their English translations:

Navajo	English
íísínísts'ą́ą́'	I listen
bighánísh dééh	I sift it (as flour)

K6. Only one of the two verbs above is transformed successfully by the Second-Personizer Machine. Which verb do you think will be transformed unsuccessfully, and what will the machine output for this verb?

Verb: _____ Output: _____