Solution to Problem 11

Problem. In class, we designed a Turing machine for computing π_2^2. Use this design as a sample to design a Turing machine for computing π_2^3. Trace, step-by-step, on an example, how your Turing machine works. For example, you can take as input the triple $(1, 3, 2)$. Also, check also that your code works when one of the numbers is 0, especially when the third number is 0.

The Turing machine for computing π_2^2 is based on the following idea:

- first, we place 1 in the very first cell, to make sure that we will know when to stop when we get back,
- then, one by one, we eliminate all the ones from the 1st number,
- then, we go to the second number and continue going until we reach the first blank space after its end,
- we need to move the second number closer to the starting cell,
- moving a unary number one step to the left means that we erase the last 1, and add a 1 before this number; this will keep the same number of ones, but we get one step closer to the starting cell of the Turing machine,
- so, once we reach the blank space after the second number, we go back one step, erase the 1 symbol, and start going left,
- we go left until we reach the end of the number, i.e., the first blank space, which we replace by 1,
- if directly to the left of the replaced black space is a symbol 1, this means that we are at the starting cell of the Turing machine, thus we have moved the number already; now all we need to do is replace this symbol 1 with blank space and stop,
- on the other hand, if directly to the left of the replaced blank space is an empty cell, this means that we need to again go right and repeat the same move-one-step-to-the-left procedure.

A special care needs to be taken for a special case when the second component of the original pair is number 0. In this case, once we erase the 1st number, there is nothing left to erase, so we simply go back (and replace 1 back to blank when we reach the starting cell).

This Turing machine has the following main rules:

- start, – → 1, R, erase1st
- erase1st, 1 → R, –
- erase1st, – → R, right
- right, 1 → R
- right, – → L, erase
- erase, 1 → –, L, left
- left, 1 → L
- left, – → 1, L, checking
- checking, – → R, right
- checking, 1 → –, halt.

The following three additional rules take care of the case when the second number is 0:

- erase, – → L, finish
- finish, – → L
- finish, 1 → –, halt.

Solution. The main idea is that if we have the 3rd number, then, before we start moving the 2nd number to the left, we need to erase the third number:

- start, – → 1, R, erase1st
- erase1st, 1 → R, –
- erase1st, – → R, in2nd
- in2nd, 1 → R
- in2nd, – → R, in3rd
- in3rd, 1 → R
- in3rd, – → L, erase3rd
- erase3rd, 1 → –, L
- erase3rd, – → L, erase
- right, 1 → R
- right, – → L, erase
- erase, 1 → –, L, left

- left, $1 \to$ L
- left, $- \to 1$, L, checking
- checking, $- \to$ R, right
- checking, $1 \to -$, halt.
- erase, $- \to$ L, finish
- finish, $- \to$ L
- finish, $1 \to -$, halt.

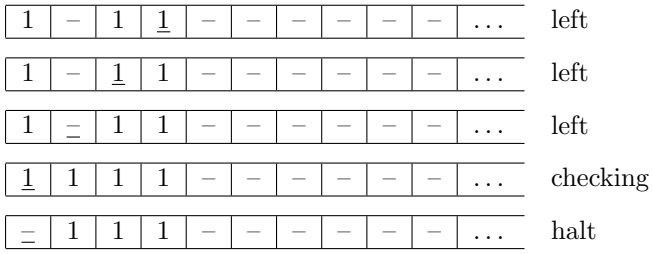
Let us trace this Turing machine for the triple $(1, 3, 2)$:

$\underline{-}$	1	–	1	1	1	–	1	1	–	...	start
1	$\underline{1}$	–	1	1	1	–	1	1	–	...	erase1st
1	–	$\underline{-}$	1	1	1	–	1	1	–	...	erase1st
1	–	–	$\underline{1}$	1	1	–	1	1	–	...	in2nd
1	–	–	1	$\underline{1}$	1	–	1	1	–	...	in2nd
1	–	–	1	1	$\underline{1}$	–	1	1	–	...	in2nd
1	–	–	1	1	1	$\underline{-}$	1	1	–	...	in2nd
1	–	–	1	1	1	–	$\underline{1}$	1	–	...	in3rd
1	–	–	1	1	1	–	1	$\underline{1}$	–	...	in3rd
1	–	–	1	1	1	–	1	1	$\underline{-}$	...	in3rd
1	–	–	1	1	1	–	1	$\underline{1}$	–	...	erase3rd
1	–	–	1	1	1	–	$\underline{1}$	–	–	...	erase3rd
1	–	–	1	1	1	$\underline{-}$	–	–	–	...	erase3rd
1	–	–	1	1	$\underline{1}$	–	–	–	–	...	erase
1	–	–	1	$\underline{1}$	–	–	–	–	–	...	left
1	–	–	$\underline{1}$	1	–	–	–	–	–	...	left
1	–	$\underline{-}$	1	1	–	–	–	–	–	...	left
1	$\underline{-}$	1	1	1	–	–	–	–	–	...	checking
1	–	$\underline{1}$	1	1	–	–	–	–	–	...	right
1	–	1	$\underline{1}$	1	–	–	–	–	–	...	right

1	–	1	1	$\underline{1}$	–	–	–	–	–	...	right
1	–	1	1	1	$\underline{–}$	–	–	–	–	...	right
1	–	1	1	$\underline{1}$	–	–	–	–	–	...	erase
1	–	1	$\underline{1}$	–	–	–	–	–	–	...	left
1	–	$\underline{1}$	1	–	–	–	–	–	–	...	left
1	$\underline{–}$	1	1	–	–	–	–	–	–	...	left
$\underline{1}$	1	1	1	–	–	–	–	–	–	...	checking
$\underline{–}$	1	1	1	–	–	–	–	–	–	...	halt

For the triple $(1, 3, 0)$, we have the following:

$\underline{–}$	1	–	1	1	1	–	–	–	–	...	start
1	$\underline{1}$	–	1	1	1	–	–	–	–	...	erase1st
1	–	$\underline{–}$	1	1	1	–	–	–	–	...	erase1st
1	–	–	$\underline{1}$	1	1	–	–	–	–	...	in2nd
1	–	–	1	$\underline{1}$	1	–	–	–	–	...	in2nd
1	–	–	1	1	$\underline{1}$	–	–	–	–	...	in2nd
1	–	–	1	1	1	$\underline{–}$	–	–	–	...	in2nd
1	–	–	1	1	1	–	$\underline{–}$	–	–	...	in3rd
1	–	–	1	1	1	$\underline{–}$	–	–	–	...	erase3rd
1	–	–	1	1	$\underline{1}$	–	–	–	–	...	erase
1	–	–	1	$\underline{1}$	–	–	–	–	–	...	left
1	–	–	$\underline{1}$	1	–	–	–	–	–	...	left
1	–	$\underline{–}$	1	1	–	–	–	–	–	...	left
1	$\underline{–}$	1	1	1	–	–	–	–	–	...	checking
1	–	$\underline{1}$	1	1	–	–	–	–	–	...	right
1	–	1	$\underline{1}$	1	–	–	–	–	–	...	right
1	–	1	1	$\underline{1}$	–	–	–	–	–	...	right
1	–	1	1	1	$\underline{–}$	–	–	–	–	...	right
1	–	1	1	$\underline{1}$	–	–	–	–	–	...	erase


1 – 1 1 – – – – – – ... left
1 – 1 1 – – – – – – ... left
1 – 1 1 – – – – – – ... left
1 1 1 1 – – – – – – ... checking
– 1 1 1 – – – – – – ... halt