TIC - Turing International Contest (I Edition)

Exercises

- If not otherwise specified, input sequences are always assumed to be *non empty.* That is, all input sequences contain at least one symbol.
- Integer numbers are assumed to be greater than or equal to zero, represented in decimal notation (using digits 0,1,...,9) with no leading zeros which are not significant. For example, 0 and 19 are valid decimal numbers, while 0032 must be written as 32.
- When producing a solution, recall to remove from the final tape all symbols that do not make up the answer!
- Each time you save a solution of an exercise on the Turing Machine simulator, the timestamp of the exercise gets updated with the current time.

Exercise 1 Heads or Tails [1 point].

A game of "heads or tails" is played by two players as follows. Initially, the first player decides between betting on *head* (symbol `H`), or *tail* (symbol `T`). Then, the second player is forced to bet on the option not taken by the first player. Finally, a coin is tossed, and based on the outcome of the toss, either the first player or the second scores a point. Write a Turing Machine program which, given an input tape containing the first player's choice (`H` or `T`), appends to it the second player's forced choice (`T` or `H`, respectively), and terminates.

Example

Input tape	Final tape
`H`	`HT`
`T`	`TH`

Exercise 2 Keeping Score [3 points].

Consider the game of "heads or tails" from the previous exercise. A full game is encoded by following the bets of the two players with the actual outcome of a single toss (again, either `T` or `H`). Write a program for a Turing Machine that, when given as input a sequence of full games (with no intervening separators), terminates by leaving on the tape the cumulative score of the first player, as a decimal number.

Example

Input tape	Final tape
`HTT`	`0`
`HTH`	`1`
`HTTHTHTHT`	`2`
`THTHTHTHHTHTHTHTHHTHHTHTHTHTHHHTHTHTHTTHTHTHT`	`10`
`THHHTT`	`0`

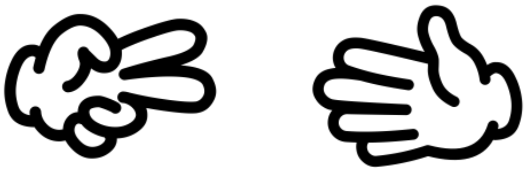
Exercise 3 Odds and Evens [4 points].

In a game of "odds and evens", two players simultaneously reveal their hands, each of which indicates a number between 0 and 5 depending on how many fingers are extended. Before revealing their hand, the players bet on whether the sum of the two numbers will be odd (symbol `O`) or even (symbol `E`) – of course, the two players will bet on different outcomes, as for "heads or tails". The player who bet on the correct answer wins the game.

We will encode a game of "odds and evens" by writing on the tape the bets of the two players, in order, (`OE` or `EO`), followed by their numbers (each between 0 and 5, again in the same order). Write a program for a Turing Machine that, when given as input the encoding of a game as specified above, terminates leaving on the tape the same input, followed by the symbol "`A`" if the first player won, or "`B`" if the second player won.
Notice: 0 is considered an even number.

Example

Input tape	Final tape
EO33	EO33A
EO34	EO34B
OE05	OE05A
OE55	OE55B



Exercise 4 The unbalanced coin [6 points].

We would expect a coin toss in "heads or tails" to result with the same likelihood in a `H` or `T` (i.e., 50% probability of each result). An unbalanced coin might end up producing significantly more `H` than `T`, or vice-versa. Of course, on a small number of tosses any difference in the frequency of the two results might be due to chance: but the chance that significant differences in frequency are observed in a given number of tosses, decreases as the number of tosses increases (i.e., in 1000 tosses we would expect `H` and `T` to appear *almost* exactly 500 times each).

We declare a coin *unbalanced* if in a series of tosses of length 10 or more, the number of `H`s and `T`s differ by at least 25% of the number of tosses. Write a program for a Turing Machine that, given as input a sequence of tosses (each of them encoded as a single `H` or `T`) of length > 9, terminates by leaving on the tape "`OK`" if the coin was balanced, "`KO`" otherwise.

Example

Input tape	Final tape
HTHTHTHHTTTHHTTHTHH	OK
THTTHTHTTTHTTHTTTHT	KO
HTHTHHTTHTHTHHHTHHHTTTHTHTHHHTHTTTHTHTHT	OK

Exercise 5   A chain of words [7 points].

A series of words form a *chain* if the *last segment* of each word, except for the final one, is a prefix of the next word. The last segment of a word is defined as the substring of the word starting with the last vowel (included) and ending at the end of the word. For this problem, we consider `A`, `E`, `I`, `O`, `U`, `Y`, `W` to be vowels. A typical word game based on this concept would have each player in a group, in turn, quickly come up with a word that "continues the chain" left by the previous player, until one utters a non-chained word, or takes too long to answer. As an example, the following is a valid chain: *round under error orifice cement entangled education onwards.*

Write a program for a Turing Machine that, given as input a series of words, terminates leaving on the tape the first non-chained word and all subsequent ones, or an empty tape if all words were chained (i.e., if the sequence was a valid chain).

Example

Input tape	Final tape
`CARD ARDENT ENTITY TYPE ERROR`	`TYPE ERROR`
`BOARD ARDENT ENTITY YOGURT`	
`BALL ALL ALLEGATION NATION`	`NATION`
`BALLROOM ROOM`	`ROOM`
`BALLROOM ONWARD ARDUOUS USING`	`ONWARD ARDUOUS USING`
`RED EDUCATION ONTARIO OLIVE EMINENCE EDGY`	

Exercise 6   Top 3 cards [8 points].

A deck of French playing cards is composed of four suits: Hearts, Tiles, Clovers and Pikes (we will use symbols `H`, `T`, `C`, `P` for suits). Each suit includes 13 cards of different values, traditionally marked - in increasing value order - `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `J`, `Q`, `K`, `A`. To keep things simple, we will use a single symbol "`0`" (zero) to signify 10.

Hearts	Tiles	Clovers	Pikes
♥	♦	♣	♠

A card will thus be identified by its suit and value, e.g. `4H` to mean "four of hearts". In our game, the rank of a card is determined by its value, and in case of value parity by its suit, with `H` > `T` > `C` > `P`. Thus, `2P` is the lowest-ranked card of all, and `AH` is the highest-ranked. Write a program for a Turing Machine that, given as input a hand of cards from a single deck (hence, there can be no duplicates) composed of at least 3 and at most 52 cards, terminates leaving on the tape only the three top-valued cards in the hand, in descending order of rank.

Example

Input tape	Final tape
`4H8P8HKT6T`	`KT8H8P`
`JC4HAT9CQP5HJH8T2H5C`	`ATQPJH`
`2P2C2T2H`	`2H2T2C`
`2P3P4P`	`4P3P2P`
`9T0H5T3T0PAH`	`AH0H0P`

Exercise 7 Poker [11 points].

In the game of Poker, each player has a hand of 5 cards drawn from a French deck as described in the previous problem. The game has an elaborate structure of bets and card swapping, in several variants, but we will consider the simpler problem of identifying the strongest combination that a player has in hand. These combinations, in decreasing order of strength, are as follows:

Name	Description	Example
Straight flush	5 cards of consecutive values, all of the same suit	`4H 5H 6H 7H 8H`
Four of a kind	4 cards of the same value	`QH QC QT QP`
Full house	3 cards of one value and 2 cards of another value	`3C 3H 3P JH JT`
Flush	5 cards of the same suit	`3C 7C AC 8C JC`
Straight	5 cards of consecutive values	`9H 0C JH QT KC`
Three of a kind	3 cards of the same value	`5P 5H 5C`
Two pairs	2 cards of one value, 2 of another value	`4P 4H KH KT`
Pair	2 cards of the same value	`AH AP`
High card	1 card (the one with the highest rank in the hand)	`KH`

Write a program for a Turing Machine that, when given as input a Poker hand represented as a sequence of 5 cards, separated by spaces, terminates leaving as output the strongest combination in the hand, in the same format, with the cards in the same order they had in the original hand.

Example

Input tape	Final tape
`4H 8P 8H KT 6T`	`8P 8H`
`JC 4H AT 0C QP`	`AT`
`2P 2C 2T 2H AH`	`2P 2C 2T 2H`
`3C 3P 3H AH AC`	`3C 3P 3H AH AC`
`6C 2P 3P 4P 5H`	`6C 2P 3P 4P 5H`
`3P JH 0P JP JC`	`JH JP JC`

Exercise 8 Undercut [12 points].

In this game, each of two players simultaneously chooses a number between 0 and 5 (as in "odds and evens"). If the two numbers are consecutive (e.g, player A chooses 3 and player B chooses 4), then the player who chose the lesser of the two numbers gets as many points as the sum of the two numbers (e.g., player A gets 7 points, player B 0 points). In all other cases, (e.g., player A chooses 2 and player B chooses 5) each player gets as many points as the number they chose (e.g., player A gets 2 points and player B gets 5 points). The game is played repeatedly until a player reaches a score of 50 or more, upon which the player with the highest score is declared the winner and the game ends. It is possible that the two players cross the 50 points mark in the same round and end up with exactly the same final score, in which case the game ends in a draw.

Write a Turing Machine program that, when given as input the current scores of two players A and B (each represented as a two-digits decimal number, separated by `/`), followed by a symbol `=` and then the numbers called by the two players (each of them between `0` and `5`), terminates leaving on the tape the updated scores, separated by `/`, followed by `=`, and if the game is finished, one of the strings "`A WINS`" or "`B WINS`" or "`DRAW`", according to the outcome. If the game is not finished yet, nothing follows the `=` symbol.

Example

Input tape	Final tape
`00/00=35`	`03/05=`
`00/00=34`	`07/00=`
`12/15=55`	`17/20=`
`12/15=01`	`13/15=`
`17/20=54`	`17/29=`
`48/30=12`	`51/30=A WINS`
`30/48=52`	`35/50=B WINS`
`49/45=15`	`50/50=DRAW`

Exercise 9 An AI for Undercut [20 points].

A strategy to win a game of Undercut is to correctly predict what number the opponent will choose next. If we predict the opponent will choose 1, our best move is to choose 5, gaining a +4 advantage. If we predict the opponent will choose 5, our best move is to choose 4, gaining a +9 advantage, etc. Humans tend to be repetitive. For example, if we have noticed that after playing a 5, and losing 9 points against our 4, our opponent has in the past most often chosen a 1, we can then predict he will choose a 1 again, and thus play a 5 in order to maximize our win.

We want to develop an automated Undercut player (sometimes called an "artificial intelligence", or AI) that will work based on this principle. In particular, let us assume that the program has a record of all the numbers played by the two players in a match so far, as a sequence of numbers (in pairs: player A first, player B second). The AI plays as player A. The AI has to check which number has been played most often by player B in the past right after the pair of numbers that have just been played, and then choose a number (in 0-5) that will maximize player A's expected gain. In case multiple numbers have been chosen by the opponent the same number of times, the AI will predict the highest. If multiple numbers would produce the same advantage gain for the given prediction, the AI will select the lowest (for example, "12" makes A gain 3 points and B 0, while "52" makes A gain 5 points and B 2: in both cases the advantage gained by A is 3 points, so playing 1 is preferred by the AI). If the pair just played has never appeared in the past, the AI will select 3.

Write a program for a Turing Machine that implements the AI playing as player A, having as input a non-empty sequence of pairs played so far (thus, an even number of symbols in `0-5`) separated by spaces, and terminates leaving on the tape player A's best next move.

Example

Input tape	Final tape
`32 54 35 25 32 51 54 14 32 15 54 35`	`4`
`32 54 35 25 32 51 54 14 32 15 54 54`	`3`
`32 54 35 25 32 51 54 14 32 15 54 32`	`4`
`32 54 35 25 32 51 54 14 32 15 54 50`	`3`
`32 54 35 25 32 51 54 14 32 15 54 25`	`1`
`55`	`3`
`55 31`	`3`
`55 31 31`	`5`
`55 31 31 55`	`5`
`55 31 31 55 54`	`3`
`55 31 31 55 54 31`	`4`
`55 31 31 55 54 31 54`	`5`
`55 31 31 55 54 31 54 54`	`3`

Exercise 10 Connect Four [25 points].

The game Connect Four is played on a grid of 6 rows and 7 columns. Two players play the game, A and B; each of them, in turn, can place one of their tokens (which we will denote with symbols `A` and `B` respectively) on the grid. Player A moves first. Tokens can only "slide down from the top", e.g. they can only be placed in a free cell on the bottom row, or immediately on top of another token. The first player to form a straight line (horizontal, vertical or diagonal) of at least 4 tokens of the corresponding type, wins the game.

We want an AI playing as player B, that looking at the state of the board (immediately after player A's move), if there is a risk that player A will win on the next round, will suggest a move that will prevent that win (a "blocking move"), unless B can win on this round (a "winning move"), in which case the AI should suggest the winning move instead.

Write a program for a Turing Machine that, given an encoding of the board (with `A` and `B` denoting placed tokens, `#` denoting a free cell, and each row of the board, from top to bottom, consecutively placed with no separator on the tape) will terminate leaving as only output the index of the column (`1`-`7`, numbered left-to-right) of the winning move if there is one, or of a blocking move if there is one (and there are no winning moves), or `0` otherwise. In case multiple winning or blocking moves are possible (e.g., a line of three tokens can be extended on both ends), the program will output the index of the lowest-numbered column which constitutes a winning or blocking move.

Note: In the following example we split the input on multiple rows for clarity, but all symbols will be consecutive on the tape, one row at a time from top to bottom. For example, the input tape of the first example is `##################################B#AAA##B` .

Example

Input tape	Final tape / *Comment*

`#######`
`#######`
`#######`
`#######`
`######B`
`#AAA##B`
	`1`
	There are no winning moves and 2 blocking moves, 1 and 5. 1 is the lowest-numbered one.

`#######`
`#######`
`#######`
`##A###B`
`##AA##B`
`#AABABB`
	`7`
	7 is a winning move.

`#######`
`#######`
`#######`
`##A####`
`##A###B`
`#AABABB`
	`3`
	There are no winning moves and 1 blocking move, in column 3.

`#######`
`#######`
`#######`
`##B###A`
`##A###B`
`#AABABB`
	`0`
	There are no winning moves and no blocking moves; in fact, no player has 3 tokens already aligned.

`#######`
`#######`
`#######`
`##BAA#A`
`#BABBAB`
`#AABABB`
	`5`
	There are no winning moves and 1 blocking move, in column 5.