Name _____ Period _____

Card Search and Sort

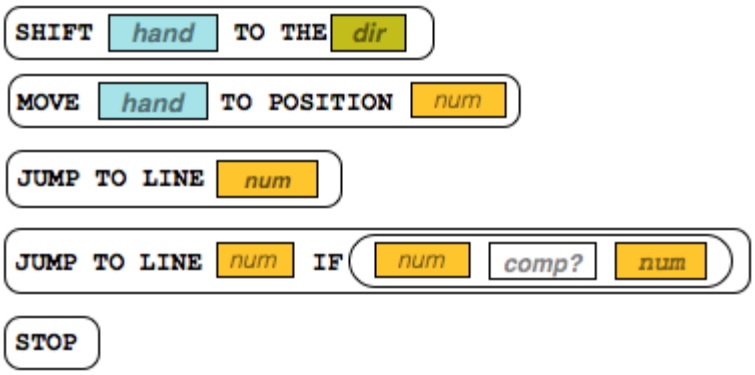
Your Tasks (Mark these off as you go)

- ☐ Get familiar with the machine language commands
- ☐ Practice with the Machine Language Commands
- ☐ Write an algorithm to search for a card
- ☐ Write an algorithm to bring the min card to the front
- ☐ Write an algorithm to sort the cards from low to high
- ☐ Write an algorithm to find a card in an ordered list
- ☐ Receive credit for this lab guide

☐ Get familiar with the machine language commands

Here are the beginnings of a low-level language you can use to create programs for a "Human Machine" to solve problems with playing cards.

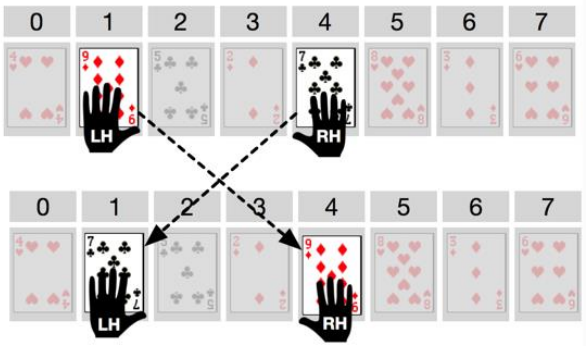
The 5 commands you can use are shown to the right. **See the *Reference Guide* provided for descriptions of what these commands do.**



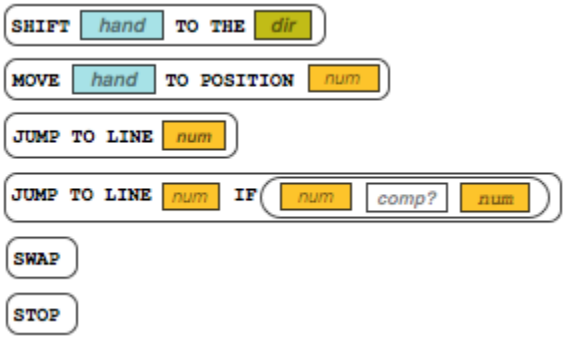
Some of these commands might seem unusual, but we can write programs with just these commands to control the "human machine's" hands to touch or pick up the cards, look at their values, and move left or right down the row of cards.

In addition to the above commands, we have an additional command called **SWAP** - see description below.

The SWAP command swaps the positions of the cards currently being touched by the left and right hands. After a swap the cards have changed positions but hands return to original position.

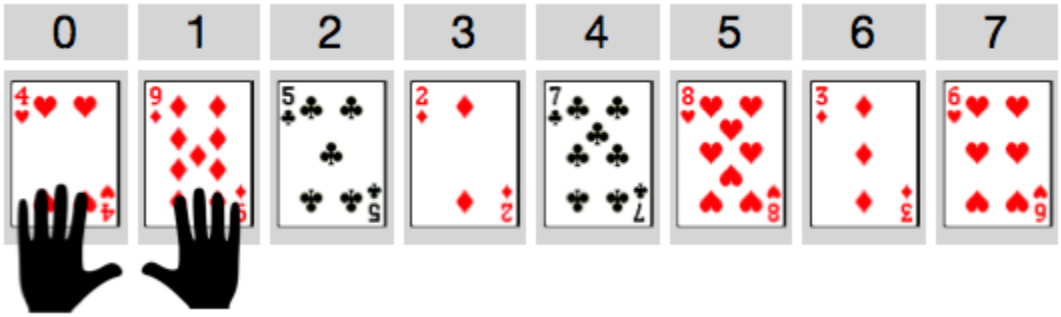


There are 6 commands total in the language available for you to use. These are shown below.



☐ Practice with machine language commands

For this activity, you should assume this standard initial setup. Here is a diagram for an 8-card setup:

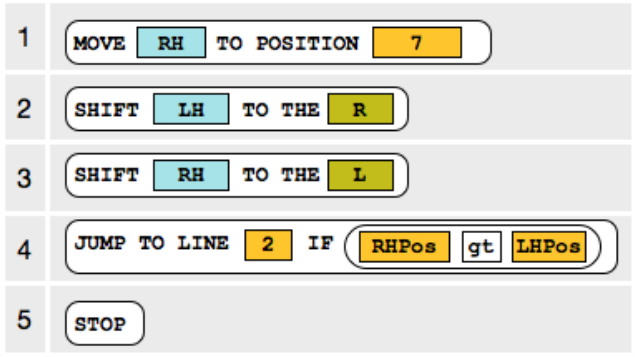
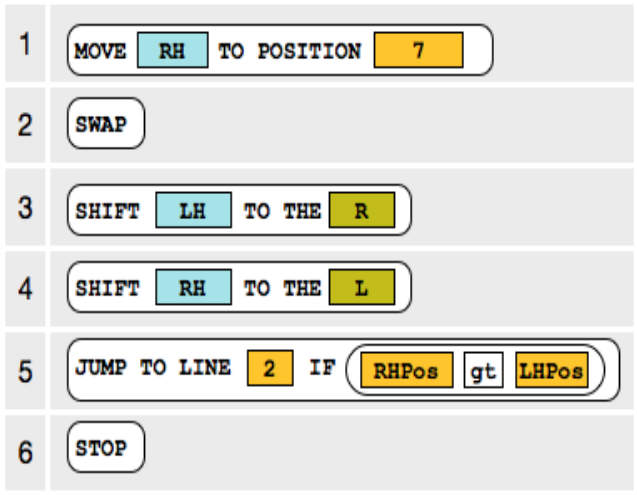


- There will be some number of cards with random values, lined up in a row, face up.
- Positions are numbered starting at 0 and increasing for however many cards there are.
- The left and right hands start at positions 0 and 1 respectively.

Get to know the Human Machine Language by acting out the following examples with a partner. For each of the examples you should:

- Lay out a row of **8 cards** randomly in front of you to test out the program.
- Have one partner read the instructions in sequence starting at line 1, and the other partner act out each command as the human machine.
- Use the ***code reference*** to answer your questions and verify you're interpreting the code correctly.
- Give a brief description of what the program does, or its ending state.

Example Program	What does it do?
1 SHIFT RH TO THE R 2 JUMP TO LINE 1 3 STOP	*Note: this one has a problem, can you find it?* RH goes out of bounds
1 SHIFT RH TO THE R 2 JUMP TO LINE 1 IF RHPos ne 7 3 STOP	The rh ends at position 7

	Shifting the rh and lh until they cross
	shifting the lh and rh, and swapping, until they cross

□ Write an algorithm to search for a card

Using only the Human Machine Language design a program to find the card with a specified value. When the program stops, the left hand should be touching the specified card. Your program must stop if one of the following conditions is met,

- The left hand is touching the specified card
- The card is not found

Use the hand position values to check whether the position is 0 or the largest position in the list - you can assume that you know how big the list is ahead of time. For example, if the last position is 7, then the comparison: **IF RHPos eq 7** would tell you that the right hand was as the end of the list.

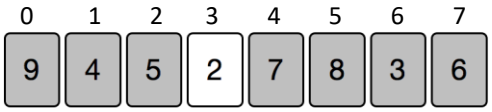
Once you have figured out an algorithm that works, write your algorithm below.

☐ Write an algorithm to bring the min card to the front

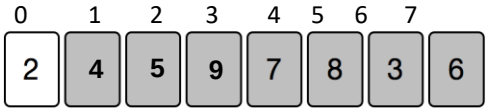
Using only the Human Machine Language design an algorithm to find the smallest card and move it to the front of the list (position 0). All of the other cards *must remain in their original relative ordering.*

END STATE: When the program stops, the smallest card should be in position 0. The card that was at index 0 is at the minimum card's spot. The position of all other cards is unchanged.

Cards BEFORE:



Cards AFTER (may not be in this order)



Once you have figured out an algorithm that works, write your algorithm below.
Navigate to the machine language simulator below to test out your code.
https://timberlinecs.github.io/HML/?swap=

☐ Write an algorithm to sort the cards from low to high

Using only the Human Machine Language design an algorithm to sort the cards from low to high.

END STATE: When the program stops, the cards should be sorted from smallest to highest. For each iteration, only the smallest card and the card in front should be swapped - the position of all other cards should remain unchanged.

Once you have figured out an algorithm that works, write your algorithm below.
I5RlLI7reaSrrJ1WJ2I11leaSlrMrlJ1X ➤ 1: JUMP TO LINE 5 IF Right Hand Card < Left Hand Card 2: JUMP TO LINE 7 IF Right Hand Position = Max Position 3: SHIFT Right Hand TO THE Right 4: JUMP TO LINE 1 5: SWAP 6: JUMP TO LINE 2 7: JUMP TO LINE 11 IF Left Hand Position = Max Position 8: SHIFT Left Hand TO THE Right 9: MOVE Right Hand TO POSITION Left Hand Position 10: JUMP TO LINE 1 11: STOP
Navigate to the machine language simulator below to test out your code.
https://timberlinecs.github.io/HML/?swap=

☐ Write an algorithm to locate a card in an ordered list

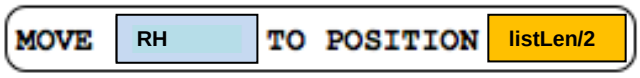
Previously you wrote an algorithm to locate a card in an unordered list. To do this required that you apply a linear search – that is, you started at the beginning of the list and compared each card to the card you were looking for until you found it. If you got to the end of the list, then you concluded that the card was not in the list.

What are situations where a linear search would be impractical?
if the list is really long and 1. the target is not in the list 2. the target is at the end of the list

Now that the list is ordered, how might you locate the item in the list using a different, more efficient algorithm?

Just like before, you can assume that you know how big the list is ahead of time. For example, if the last position is 7, then the comparison: **IF RHPos eq 7** would tell you that the right hand was at the end of the list.

If we know the list has a length of 7, then half the list has a length of 3 (if we apply integer division). This allows us to move the hand to a specific location in the list. For example,



Once you have figured out an algorithm that works. Take a picture of your algorithm and paste it in the box below. Or you can type the commands instead.

□ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.