Name _____ Role (circle one) programmer/computer/project manager

Name _____ Role (circle one) programmer/computer/project manager

Name _____ Role (circle one) programmer/computer/project manager

Name _____ Role (circle one) quality control

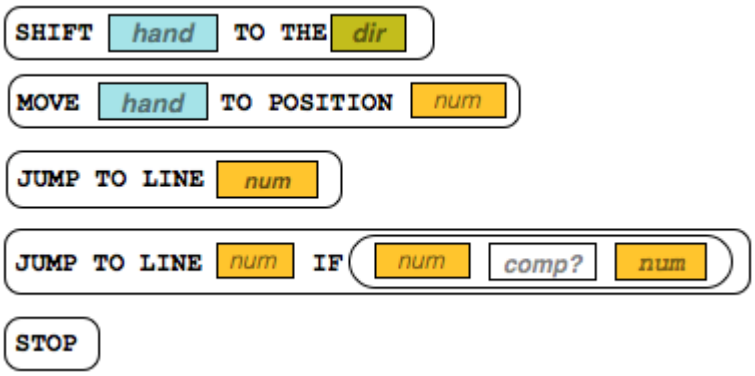
Card Search And Sort

Your Tasks (Mark these off as you go)

- ☐ Get familiar with the machine language commands
- ☐ Practice with the Machine Language Commands
- ☐ Have Ms. Pluska check of practice with machine language
- ☐ Complete challenges 1 thru 3
- ☐ Have Ms. Pluska check off your challenges 1 thru 3 before you continue
- ☐ Receive credit for the group portion of this lab
- ☐ Receive credit for the individual portion of this lab

☐ 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.

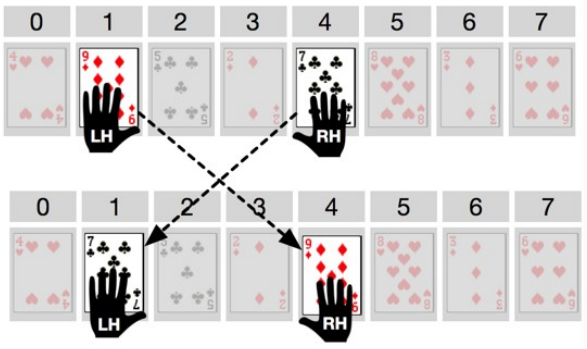


The 5 commands you can use are shown to the right. **See the *Reference Guide* provided for descriptions of what these commands do.**

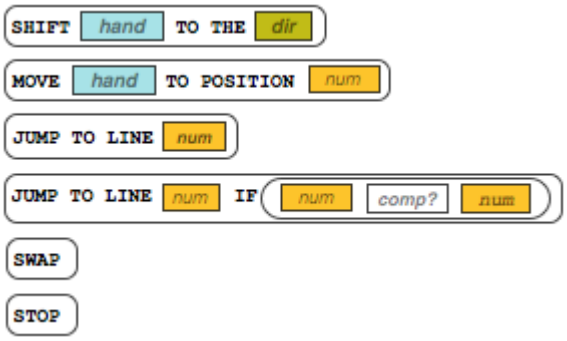
Some of these commands might seem unusual, but we can write programs with just these commands to control the "human machine's" hands to touch or pick up the cards, look at their values, and move left or right down the row of cards.

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.

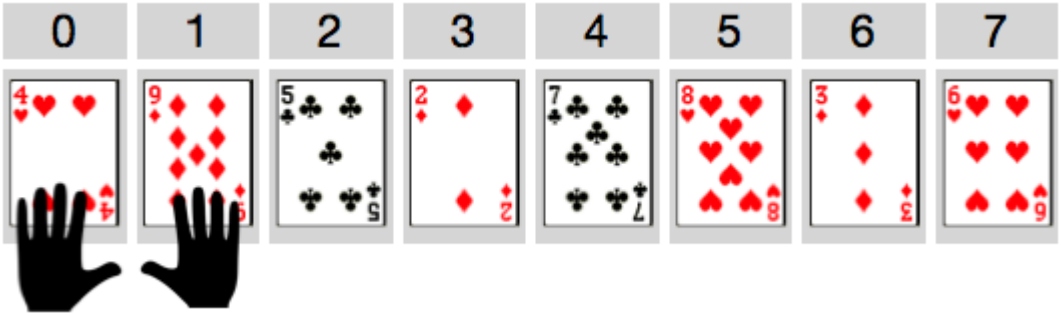


There are 6 commands total in the language available for you to use. These are shown below.



□ Practive with machine language commands

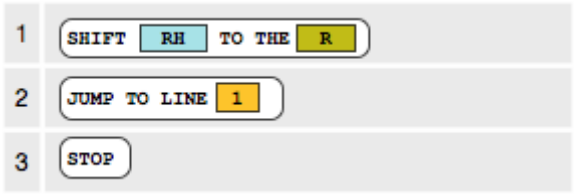
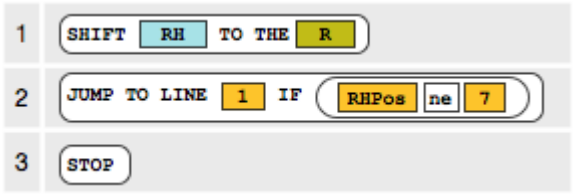
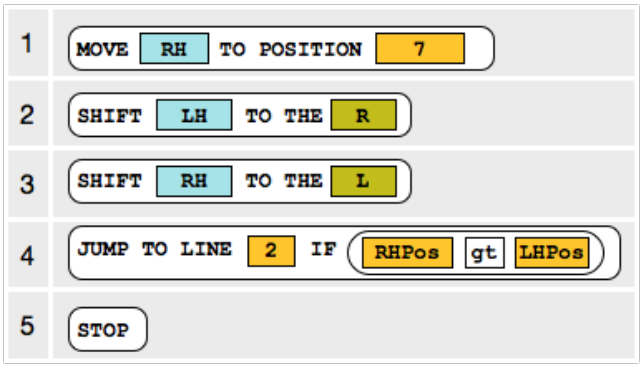
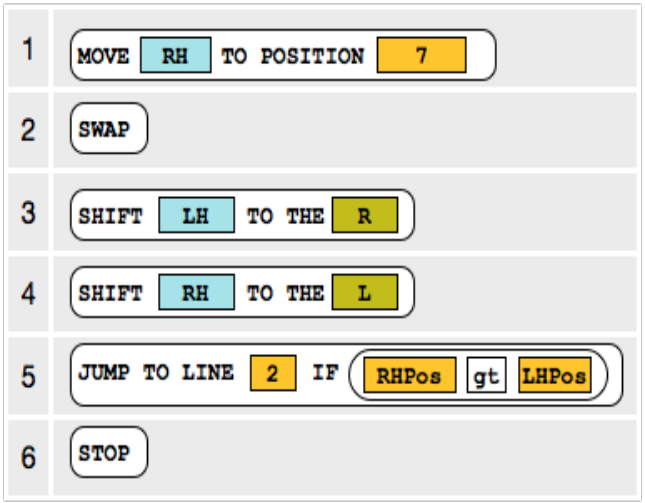
For this activity, you should assume this standard initial setup. Here is a diagram for an 8-card setup:



- 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?
	*Note: this one has a problem, can you find it?*
	
	
	

☐ Have Ms. Pluska check off practice with machine language

Before you continue have Ms. Pluska check off your practice with machine language exercises

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Complete the challenges 1 thru 3

Challenge 1: Card Search

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.

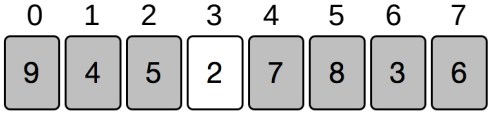
Once you have figured out an algorithm that works, write your algorithm on the paper provided.

Challenge 2: Min to Front

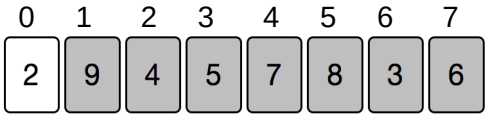
Using only the Human Machine Language design an algorithm to find the smallest card and move it to the front of the list (position 0). All of the other cards *must remain in their original relative ordering.*

END STATE: When the program stops, the smallest card should be in position 0. The ending positions of the hands do not matter, the ending positions of the other cards do not matter. *As a challenge:* try to move the min-to-front and have all other cards be in their original relative ordering.

Cards BEFORE:



Cards AFTER (may not be in this order)



Once you have figured out an algorithm that works, write your algorithm on the paper provided.

Challenge 3: Card Sort

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

□ Have Ms. Pluska check off Challenges 1 thru 3

Before you continue have Ms. Pluska check off your Card Search, Min to Front, and Card Sort challenges

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Receive Credit for the group portion of this lab

Make sure indicate the names of all group members, then submit this lab to the needs to be graded folder to receive credit for the group portion of this lab.

□ Receive Credit for the individual portion of this lab

To recieve credit for the individual portion of this lab you must implement the above methods in your CardCounter project.

Locate your CardCounter project, then locate the CardCounter class. In the CardCounter class you will write the following methods,

locateCard

In the CardCounter class write a method called locateCard. The locateCard method should search the dealtCardsArray for a Card. If the Card is found, the index of the card in the dealtCardsArray is returned, otherwise the method will return -1. The locateCard method should have the following signature,

```
public int locateCard(Card[] cArr, Card c);
```

minToFront

In the CardCounter class write a method called minToFront. The minToFront method should locate the card with the lowest value in the dealtCardsArray then move it to the front of the array. The minToFront method should have the following signature,

```
public void minToFront(Card[] cArr);
```

cardSort

In the CardCounter class write a method called cardSort. The cardSort method should accept an unordered array of cards, then return the sorted array. The cardSort method should have the following signature,

```
public Card[] cardSort(Card[] cArr);
```