Weekly Planner: AP CSP week of 2.3.20

BIG IDEA for the week:

3- ALGORITHMS AND PROGRAMMING

Day	
Mon 2.3	**Unit 3, lesson 2. Teams should complete all videos and activities in unit 3 lesson 2&3.** (don't panic if you can't make it through lesson 3) ***HW= U3L2&3 homework**(due tues night) **on schoology, thoroughly research and answer the warm up question from this morning**
Tues 2.4	***HW= U3L2&3 homework**(due tues night) **on schoology, thoroughly research and answer the warm up question from this morning 2/3<--DUE TONIGHT!**
Wed 2.5	**U3 L4 & 5**
Thurs 2.6	**U3 L6 and L7** https://www.youtube.com/watch?v=pGnDInYzpes abstraction vs not abstraction https://studio.code.org/s/csp3-2019/stage/7/puzzle/11?section_id=2600619
Friday	**https://studio.code.org/s/applab-intro/stage/1/puzzle/1** https://www.youtube.com/watch?v=tDnoxkOSfQw (metamorphus) **U3 L8 and L9**

https://www.google.com/search?q=how+to+use+app+lab&rlz=1C1GKLA_enUS808US808&oq=how+to+use+app+lab&aqs=chrome..69i57j0l7.3266j0j8&sourceid=chrome&ie=UTF-8#kpvalbx=_HLg8XrTRHbOpytMP1sqOwA015

Warm up activities!

https://www.flippity.net/rp.asp?k=12CjhCA0jTq2jUcyMPs_W44wjadnS8YRDn-2paoY9M0s

Monday 2.3.20-

https://evanscсса.weebly.com/

Explain HOW and I.P address can be blocked using terms such as TCP packets, DNS, router, and ping.

You will need to go back and look at "warriors of the net" video and others:

https://www.youtube.com/watch?v=7_LPdttKXPc
https://www.youtube.com/watch?v=EOYe71RWMvk&t=6s
https://www.youtube.com/watch?v=Lq7qdOsGw7E
https://www.youtube.com/watch?v=m_6Aztlq4yM

Tuesday 2.4.20-

Make a VENN diagram describing the similarities and differences between programming language and our own “natural” language.

Wednesday 2.5.20-

Programming languages have some similarities and differences to the "natural" language you use in everyday speech. Select the two true statements about programming languages:

A. Ambiguities in natural language necessitate the creation of programming languages for controlling a computer

B. Compared to the number of words in a natural language, the number of defined words in a programming language is very small.

C. The number of defined words in a programming language is about the same as the number of words in a natural language.

D. There are typically many possible ways to interpret an instruction written in a programming language.

Thursday 2.6.20-

Copy down the following commands:

```
Line 1: REPEAT 4 TIMES

Line 2: {

Line 3:      MOVE_FORWARD()

Line 4:      ROTATE_RIGHT()

Line 5:      MOVE_FORWARD()

Line 6:      MOVE_FORWARD()

Line 7:      MOVE_FORWARD()

Line 8:      ROTATE_RIGHT()

Line 9: }
```

On the back of this sheet, draw the movement of the turtle from this code.

Friday 1.31.19- Unit 3

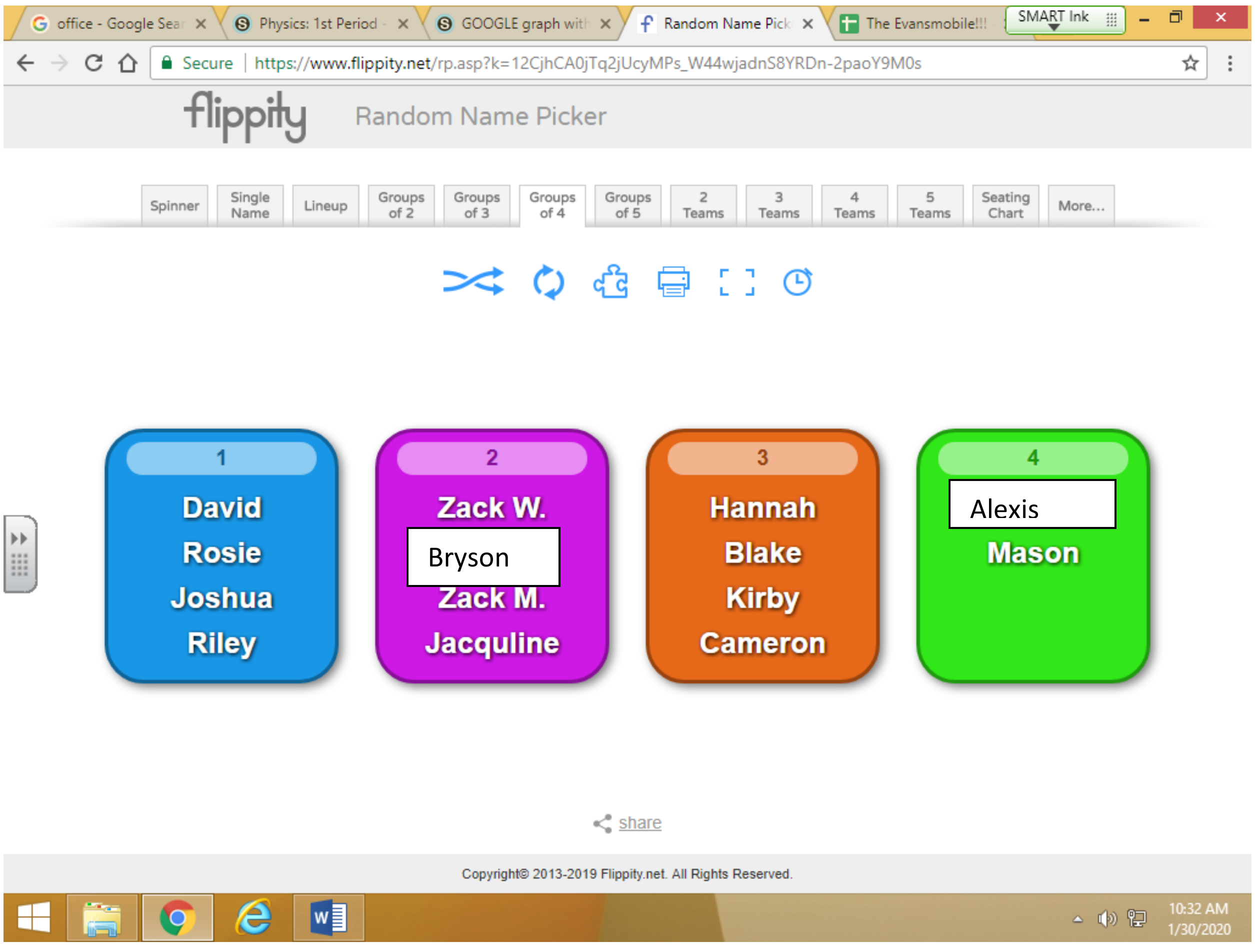
FUNCTIONS: https://docs.code.org/applab/functionParams_none/

Which of the following statements about writing functions and Top-Down Design is NOT true?

- Writing functions helps manage complexity in a program.
- Top-Down Design leads to programs which feature multiple layers of abstraction.
- Two programmers solving the same problem using Top-Down Design should arrive at identical programs.
- Top-Down Design relies upon identifying subproblems of a larger problem.
- Top-Down Design assists in identifying the layers of functions that will be used to solve a programming problem.

cheese.

adrienne.evans@sccnc.edu



Dear David, Bryson, and Kirby,

CONGRATS on being selected as your team captain! Please refer to our website https://evanscca.weebly.com/ on the AP page to see what your team will be doing today, 2/3/2020.

Your job is to make sure your team knows what to do and that they stay roughly on the same activities all period (no pair should move ahead without helping the other). You will most likely have to have them read the directions out loud. You also should be watching the videos at the same time so you can discuss if needed.

Bryson is in charge of the cards for every team in case your team needs some.

HAVE FUN!

-Ms. Evans

Programming languages have some similarities and differences to the "natural" language you use in everyday speech. Select the two true statements about programming languages:

A. Ambiguities in natural language necessitate the creation of programming languages for controlling a computer

B. Compared to the number of words in a natural language, the number of defined words in a programming language is very small.

C. The number of defined words in a programming language is about the same as the number of words in a natural language.

D. There are typically many possible ways to interpret an instruction written in a programming language.

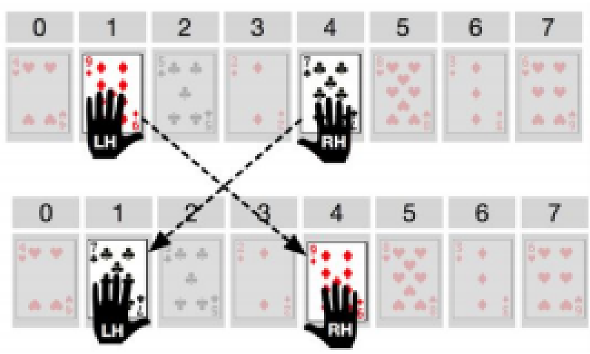
Human Machine Language - Part 2

This one goes with lesson 3. Only do if you have time.

We're going to add one command to the Human Machine Language called **SWAP** - see description below. All of the other commands are still available to you. So, there are 6 commands total in the language now.

SWAP

Swap 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.



The human machine action is: pick up the cards, exchange the cards in hand, and return hands to original position in the list with the other card.

Human Machine Language Reference

- SHIFT *hand* TO THE *dir*
- MOVE *hand* TO POSITION *num*
- JUMP TO LINE *num*
- JUMP TO LINE *num* IF (*num* *comp?* *num*)
- SWAP
- STOP

Try an example with Swap

Trace the program below with a partner and describe what it does.

1	MOVE RH TO POSITION 7
2	SWAP
3	SHIFT LH TO THE R
4	SHIFT RH TO THE L
5	JUMP TO LINE 2 IF (RHPos gt LHPos)
6	STOP

What does this program do?

Challenge: 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:

0	1	2	3	4	5	6	7
9	4	5	2	7	8	3	6

Cards AFTER (may not be in this order)

0	1	2	3	4	5	6	7
2	9	4	5	7	8	3	6

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

(If you need more lines, just keep going).

Optional Challenges

This list of challenges is given in no particular order. Find one that intrigues you and try it out.

For all of these challenges make the following assumptions:

- Cards start randomly valued, and randomly ordered, and are dealt from an infinitely large deck. I.e. you could face a row of all one value, or there could be seven 2s and one 6, and so on.
- Algorithms should work in principle for any number of cards, and any values that are comparable.
- Algorithms must STOP and be in the END STATE given in the challenge description.

Challenge Description	Example
Search for 2 or a 10 Search the list and stop when find EITHER a 2 OR a 10 (you could substitute 2 and 10 for any other two values if you like). END STATE: the left hand should be touching the first 2 or 10 encountered in the list. End state does not matter if there is no 2 or 10, but the program should stop.	BEFORE: 4 3 7 5 10 4 7 2 2 AFTER: 4 3 7 5 10 4 7 2 2 ^ (under 10) LH
Hi-Lo Find the min and max values in the list and move them to the first and last positions, respectively. END STATE: The card with lowest value in the list is in position 0, and the card with the highest value is in the last position (position 7 if there are 8 cards). The end state of the hands does not matter, the positions of the other cards does not matter.	BEFORE: 4 3 7 5 **10** 4 7 **1** 2 AFTER: **1** 4 3 7 5 4 7 2 **10**
Search for 2 and a 10 Search the list and stop once you have found BOTH a 2 AND a 10. END STATE: the left hand should be touching a 2 and the right hand should be touching a 10. End state does not matter if there is not both 2 and a 10.	BEFORE: 4 3 7 5 10 4 7 2 2 AFTER: 4 3 7 5 10 4 7 2 2 ^ (under 10) ^ (under last 2) RH LH
Sort Get the cards into sorted order from least to greatest. END STATE: end state of the hands does not matter, but cards should be in ascending order, and the program should stop.	BEFORE: 4 3 7 5 10 4 7 2 2 AFTER: 2 2 3 4 4 5 7 7 10
Partition Call the last card in the list the *pivot value*. Arrange the list so that the all of the cards less than the pivot value are to the left of it and all the cards greater than the pivot are to the right. (Cards equal to the pivot can go to the left or right of it, your choice). END STATE: The pivot value is in the middle of the list somewhere with all values less than it to the left, and all values greater than it to the right. The ordering of the cards to the left and right do matter. The end state of the hands does not matter.	BEFORE: 8 7 4 2 3 7 5 1 **6** AFTER: 4 2 3 5 1 **6** 8 7 7 BEFORE: 5 4 3 5 4 3 5 4 **1** AFTER: **1** 5 4 3 5 4 3 5 4 BEFORE: 5 4 3 5 4 3 5 4 **7** AFTER: 5 4 3 5 4 3 5 4 **7**
Count Count the number of 2s in the list and set the right hand so that its position number is equal to the number of 2s in the list. END STATE: the position number of the right hand is equal to the number of 2s in the list. If the count is higher than the possible position numbers (i.e.	BEFORE: 4 **2** 7 5 10 4 7 **2** **2** AFTER: 4 2 7 5 10 4 7 2 2 RH BEFORE: 4 3 7 5 10 4 7 1 1 AFTER: 4 3 7 5 10 4 7 1 1