CS 103 PR1 Twentyone

Mark Redekopp

Card Representation

- Unique integer for each card

cards

0	1	2	3	4		18				36					51

Card	Integer ID	Card	Integer ID	Card	Integer ID	Card	Integer ID
2-H	0	2-S	13	2-D	26	2-C	39
3-H	1	3-S	14	3-D	27	3-C	40
4-H	2	4-S	15	4-D	28	4-C	41
5-H	3	5-S	16	5-D	29	5-C	42
6-H	4	6-S	17	6-D	30	6-C	43
7-H	5	7-S	18	7-D	31	7-C	44
8-H	6	8-S	19	8-D	32	8-C	45
9-H	7	9-S	20	9-D	33	9-C	46
10-H	8	10-S	21	10-D	34	10-C	47
J-H	9	J-S	22	J-D	35	J-C	48
Q-H	10	Q-S	23	Q-D	36	Q-C	49
K-H	11	K-S	24	K-D	37	K-C	50
A-H	12	A-S	25	A-D	38	A-C	51

Shuffling

- Your code must follow this exact approach. You cannot take liberties to try to do something "equivalent"
  - Read the comment, what is n in our case?
  - If I want the random number to be between 0 and i, how do I achieve that?

cards

0	1	2	3	4										51

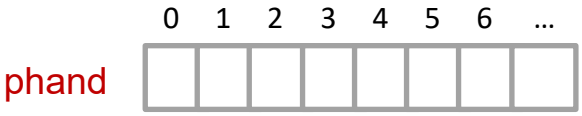
```
// To shuffle an array a of n elements (indices 0..n-1):
for i from n−1 downto 1 do
  j ← random integer such that 0 ≤ j ≤ i
      (i.e. use a modulus operation on the random number)
  exchange a[j] and a[i]
```

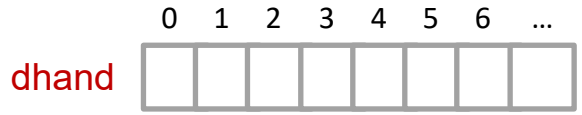
cards

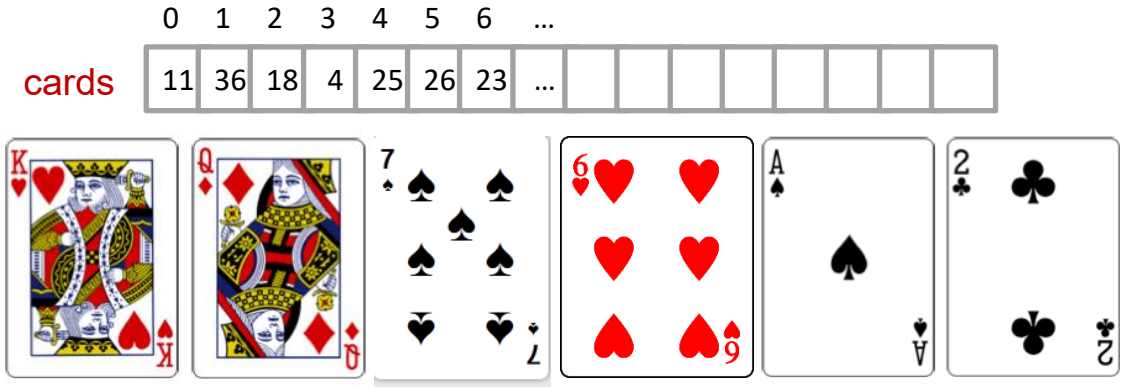
11	36	18	4	25	26	39								30

Dealing - Initial

- Dealing should not give out ALL cards at the start but happen as the game progresses
  - Deal 2 cards (in alternating fashion) to the player and dealer

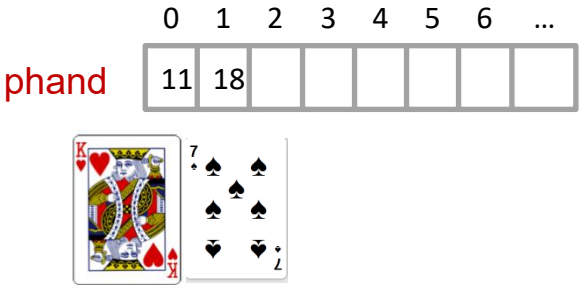




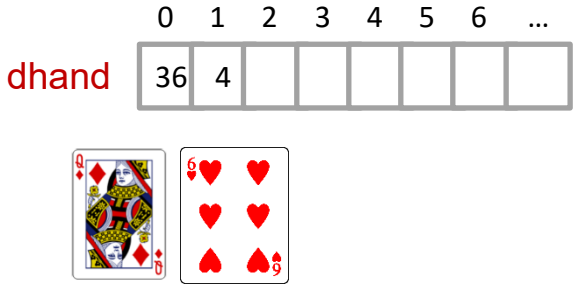


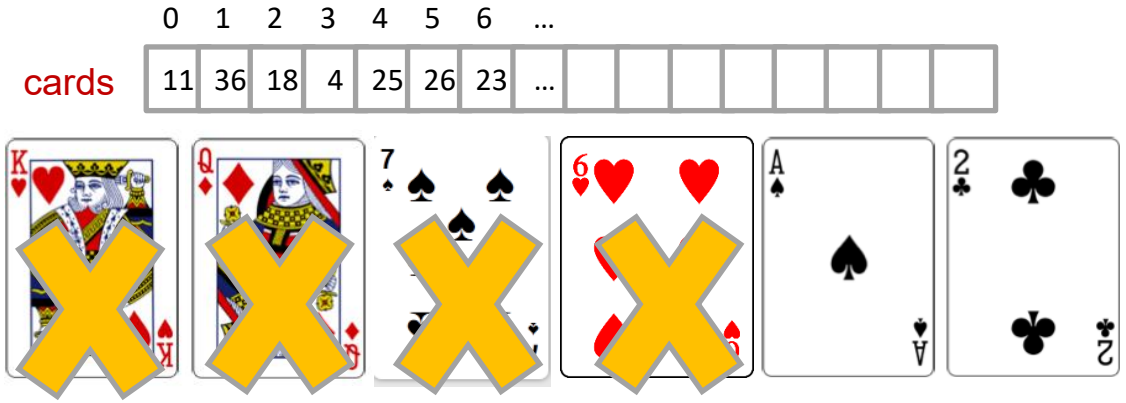
Dealing – Player's Turn

- Dealing should not give out ALL cards at the start but happen as the game progresses
  - Deal 2 cards (in alternating fashion) to the player and dealer
  - Then let player hit as many times as they want



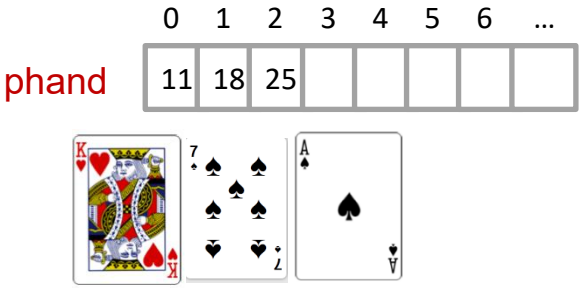
Choice: HIT



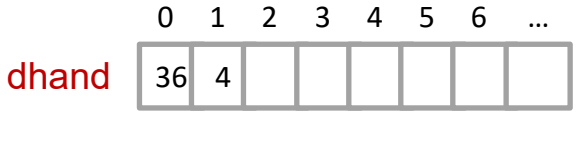


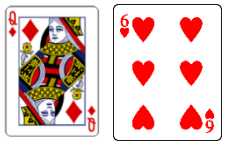
Dealing – Player Hit 1

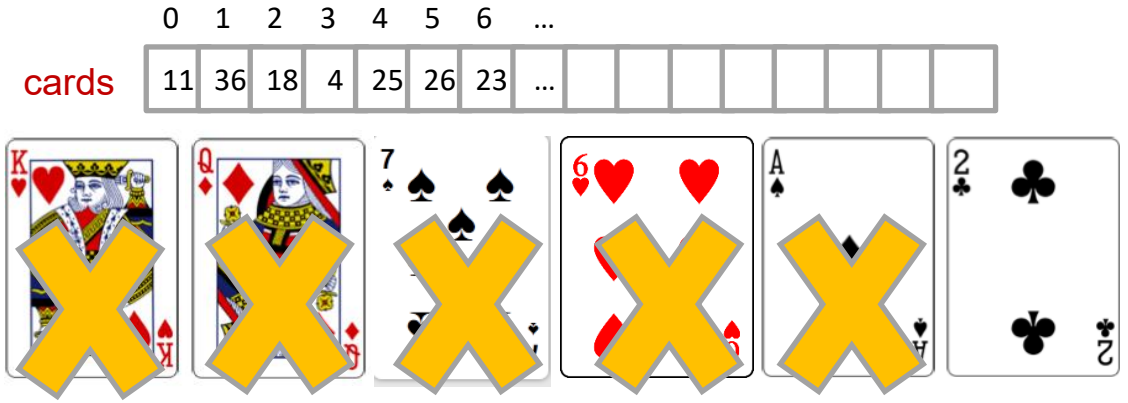
- Dealing should not give out ALL cards at the start but happen as the game progresses
  - Deal 2 cards (in alternating fashion) to the player and dealer
  - Then let player hit as many times as they want



Choice: HIT



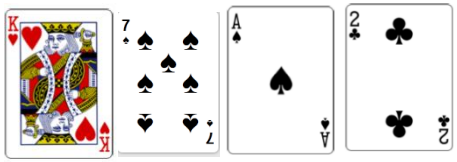




Dealing – Player Hit 2

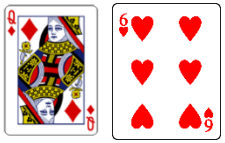
- Dealing should not give out ALL cards at the start but happen as the game progresses
  - Deal 2 cards (in alternating fashion) to the player and dealer
  - Then let player hit as many times as they want
  - Finally, let the dealer take cards

	0	1	2	3	4	5	6	…
phand	11	18	25	26				

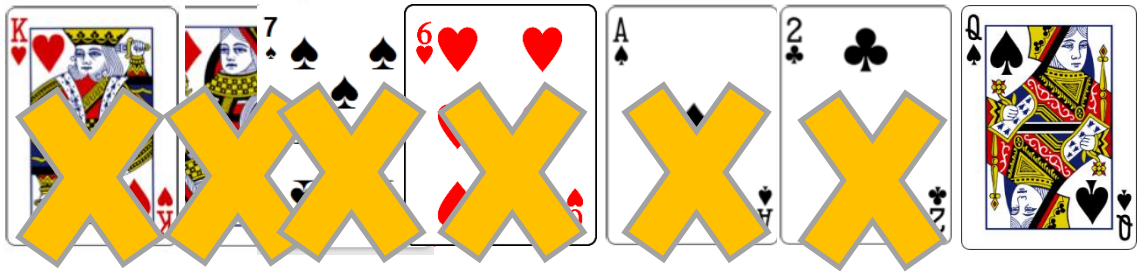


Choice: STAY

	0	1	2	3	4	5	6	…
dhand	36	4						



	0	1	2	3	4	5	6	…
cards	11	36	18	4	25	26	23	…

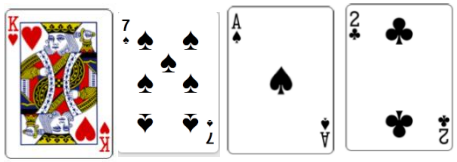


Dealing – Dealer's Turn

- Dealing should not give out ALL cards at the start but happen as the game progresses
  - Deal 2 cards (in alternating fashion) to the player and dealer
  - Then let player hit as many times as they want
  - Finally, let the dealer take cards

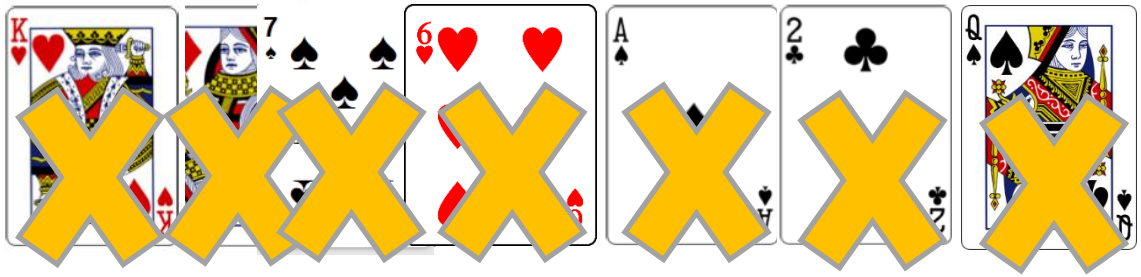
phand

0	1	2	3	4	5	6	…
11	18	25	26				



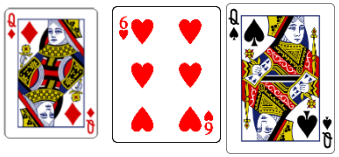
cards

0	1	2	3	4	5	6	…
11	36	18	4	25	26	23	…



dhand

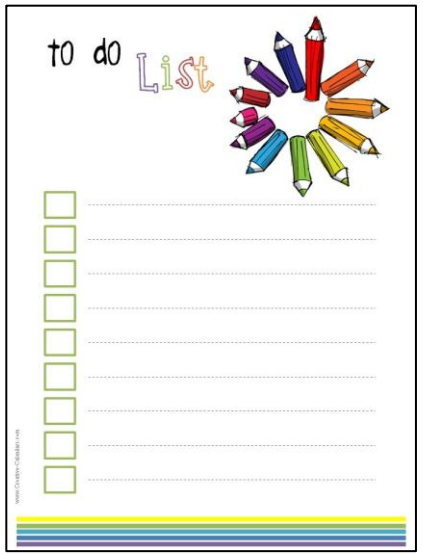
0	1	2	3	4	5	6	…
36	4	23					

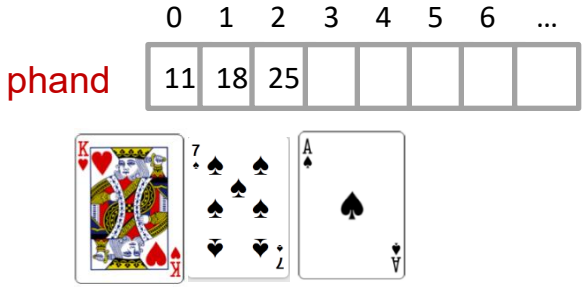


BUST

Using Arrays as Lists

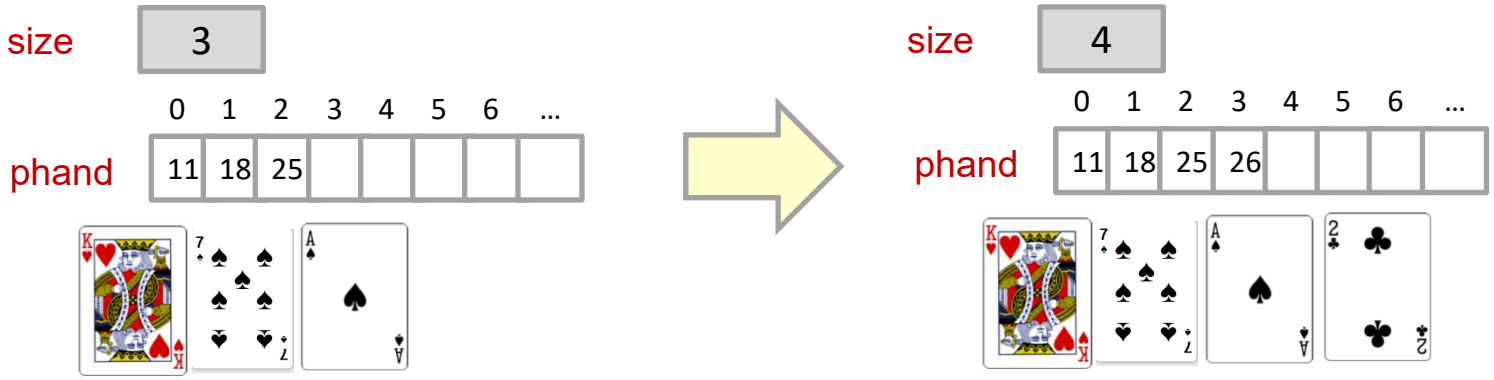
- A common programming structure is a "LIST". A list is like an array but how many elements we are actively storing can grow and shrink
  - Just like your to-do list; you add items and remove items as you finish them
- But recall, in C/C++ we must allocate an array of a FIXED size (cannot grow/shrink)
- So if our arrays are a fixed size how can we use them to mimic a "list"?





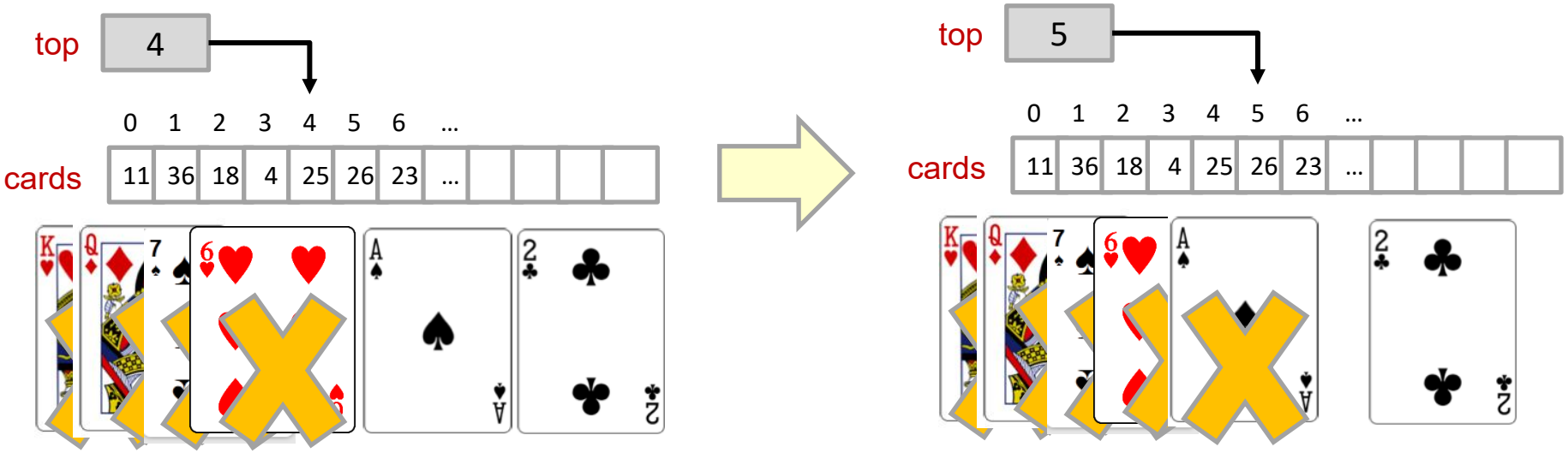
Using Arrays as Lists

- Common "List" implementation using fixed size arrays
  - Declare an array LARGE enough for the maximum number of items
  - Use a variable to track what portion is "used"



Using Arrays as Lists

- Common "List" implementation using fixed size arrays
  - Declare an array LARGE enough for the maximum number of items
  - Use a variable to track what portion is "used"
- For the deck we can use a variable to track where we should take the "NEXT" (or "top") card from



Coming Up With a Plan 1

- Start with a piece of paper and sketching out the general flow the game:

1. Shuffle
2. Deal
3. Let player "play" (until when?)
4. Let dealer "play" (until when?)
5. Summarize results
6. Repeat if they want to play again

Coming Up With a Plan 2

- Once you have a general game plan, go back and "refine" each part into more detail

1. Shuffle => Use function and code algorithm from writeup
2. Deal =>
   - Give first 4 cards (2 to each player) in alternating fashion
   - Display dealer and player hands
3. Let player "play" =>
   - Ask them if they want to hit or stay
   - If they hit, give them a card
   - Determine the hand's value
   - …
   - Repeat **until** … (what condition/)

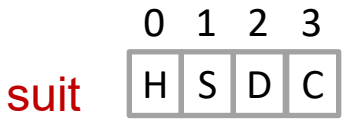
Coming Up With a Plan 3

- Only after you have a more refined plan, should you start to code!
- Look for places functions may be used (but no need to go overboard).

Using Lookup Arrays

- To print the card's suit and type or to get the card's value should require **NO if statements**
- Use arithmetic expression to convert the input card ID to the necessary index in the appropriate array
- **But how can you find that expression?**
  - **Write out several examples until you see the pattern.**

Using Lookup Arrays - Suit



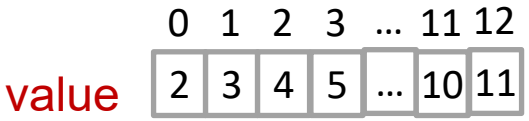
- Example: Write out some examples for converting the ID to the suit

ID	Needed index
0-12	0
13-25	1
26-38	2
39-51	3

index = f(id) ??

Card	Integer ID	Card	Integer ID	Card	Integer ID	Card	Integer ID
2-H	0	2-S	13	2-D	26	2-C	39
3-H	1	3-S	14	3-D	27	3-C	40
4-H	2	4-S	15	4-D	28	4-C	41
5-H	3	5-S	16	5-D	29	5-C	42
6-H	4	6-S	17	6-D	30	6-C	43
7-H	5	7-S	18	7-D	31	7-C	44
8-H	6	8-S	19	8-D	32	8-C	45
9-H	7	9-S	20	9-D	33	9-C	46
10-H	8	10-S	21	10-D	34	10-C	47
J-H	9	J-S	22	J-D	35	J-C	48
Q-H	10	Q-S	23	Q-D	36	Q-C	49
K-H	11	K-S	24	K-D	37	K-C	50
A-H	12	A-S	25	A-D	38	A-C	51

Using Lookup Arrays - Value



- Example: Write out some examples for converting the ID to the suit

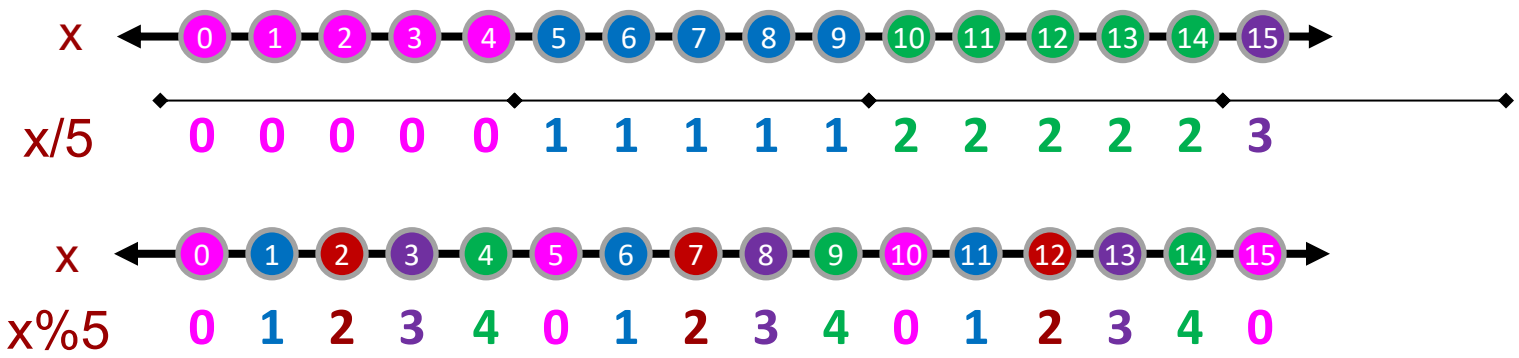
ID	Needed index
0,13,26,39	0
1,14,27,40	1
2,15,28,41	2
3,16,29,42	3
…	…

index = f(id) ??

Card	Integer ID	Card	Integer ID	Card	Integer ID	Card	Integer ID
2-H	0	2-S	13	2-D	26	2-C	39
3-H	1	3-S	14	3-D	27	3-C	40
4-H	2	4-S	15	4-D	28	4-C	41
5-H	3	5-S	16	5-D	29	5-C	42
6-H	4	6-S	17	6-D	30	6-C	43
7-H	5	7-S	18	7-D	31	7-C	44
8-H	6	8-S	19	8-D	32	8-C	45
9-H	7	9-S	20	9-D	33	9-C	46
10-H	8	10-S	21	10-D	34	10-C	47
J-H	9	J-S	22	J-D	35	J-C	48
Q-H	10	Q-S	23	Q-D	36	Q-C	49
K-H	11	K-S	24	K-D	37	K-C	50
A-H	12	A-S	25	A-D	38	A-C	51

Integer Division and Modulo Operations

- Recall integer division discards the remainder (fractional portion)
  - Consecutive values map to the **same values**
- Modulo operation yields the remainder of a division of two integers
  - Consecutive values map to **different values**
  - *x mod m* will yield numbers in the range [0 to m-1]
- Example:



Last Thought

- Start early (now) if you haven't already!

BACKUP

Dealing

- Dealing should not give out ALL cards at the start but happen as the game progresses