Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems for the time remaining in today's class.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your graders will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Problem Solving Team Members

If you are working digitally, record the name of each of your problem solving team members here.

Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

Ryan Robison
Jackson Palmer

Jump Search

The jump search algorithm has characteristics in common with both linear and binary searches. Like a linear search, it moves in only one direction, from index 0 to the end of the array. Like a binary search, jump search works only with sorted arrays and checks values at array indexes to determine if the target could possibly exist in a subrange.

A jump search begins by computing a block size of n where n is the length of the array. The block size determines the number of indexes in a subrange. A block search is conducted to determine if a block *could possibly* contain the target value. If such a block is found, a linear search is performed on the indexes within the block.

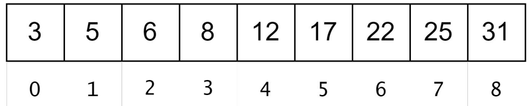
For each block, the search checks the value at the last index in the block

- If the value is the target, it has been found and index is returned
- If the value is greater than the target, the target, if it exists, must be in this block
  - A linear search is then conducted from the starting index of the block to the next to last index in the block and returns the result of the linear search
- Otherwise, the value is less than the the target, the next block is searched

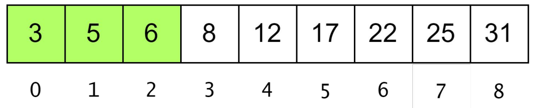
If the end of the array is reached and the target is not found, `None` is returned

Jump Search Example

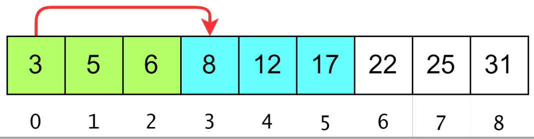
Given the array to the right, let's search for a target of 25
The array's `length` is `9`, therefore the block size is √9 or `3`



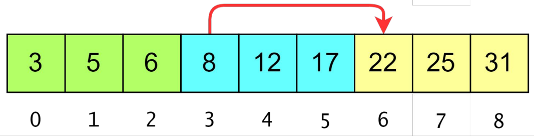
The first block to search is indexes `[0]` - `[2]` (inclusive)



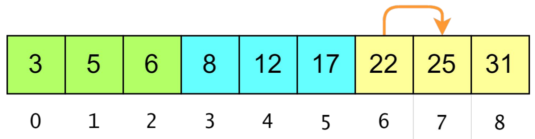
The last index in the block, `[2]`, has a value of `6`. `6` is not our target, `25`, and is also less than the target, so we jump to the next block of 3 indexes, `[3]` - `[5]` (inclusive)



The last index in the block, `[5]`, has a value of `17`. `17` is not our target, `25`, and is also less than the target, so we jump to the next, and last, block of 3 indexes, `[6]` - `[8]` (inclusive)



The value at the last index, [8], in the block, 31, is not our target, `25`, but it is greater than the target, so *if* the target exists in the array, it must be in this block, indexes `[6]` - `[8]` (inclusive)
A linear search begins from the start index of the block, `[6]` to the next to last index in the block, `[7]`
The value at index `[6]`, `22`, does not match our target, so we check the next index
The value at index `[7]`, `25` is our target, so it is found and we return our index `[7]`!!!



Problem 1A

Given the following array and target, fill in the block size, block search values, and linear search values for each iteration

1	3	4	11	17	21	26	31	32	37	40	43	58	62	65	77	91
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Block Size: 4

target = 91

Block Search					
iteration	0	1	2	3	4
block start	0	4	8	12	16
block end	3	7	11	15	16
value	11	31	58	77	91

Linear Search					
iteration	0	1	2	3	4
index	16				
value	91				

Problem 1B

Given the following array and target, fill in the block size, block search values, and linear search values for each iteration

1	3	4	11	17	21	26	31	32	37	40	43	58	62	65	77
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Block Size: 16

target = 21

Block Search					
iteration	0	1	2	3	4
block start	0	4			
block end	3	7			
value	11	31			

Linear Search					
iteration	0	1	2	3	4
index	6	5			
value	26	21			

Problem 1C

Given the following array and target, fill in the block size, block search values, and linear search values for each iteration

1	3	4	11	17	21	26	31	32	37	40	43	58	62
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Block Size: 4

target = 5

Block Search					
iteration	0	1	2	3	4
block start	0				
block end	3				
value	11				

Linear Search					
iteration	0	1	2	3	4
index	0	1	2	Not Found	
value	1	3	4		

Problem 2

Together with your team, write the pseudocode (high-level steps) for the Jump Search Algorithm.

```
Take in parameters of array and target and blocksize and block start with those last two set to none and
zero
If blocksize = none Set the block size to square root of length of the array

If array of index start + blocksize is less than target
Then call function with array, target, blocksize, start+blocksize
Else if array of index start + blocksize == target
        Return array of that index
Else if array of index is greater than target
For ind in range(blocksize)
        If array[ind+start] == target
Then return array ind+start
After that runs if nothing is found return None
```

Problem 3

Referencing your pseudocode from the previous problem, calculate the expected time complexity of the Jump Search Algorithm.

Hint: break the analysis into two parts, Block Search and Linear Search, and combine the complexity of each.

We dont know e

Problem 4

A test that is written *after the fact* to test code that is already working is called a *characterization test*. We use the test to characterize the current behavior of the code before making modifications so that we can make the modifications safely.

For the linear search that we wrote in class, list four test conditions that you would write characterization tests for. Do not write the actual tests.

Example: Empty array
1. Having target be lower than start
2. Having target higher than end
3. Having target in array
4. Having array be empty

Problem Solving 5

Given the following array and targets, fill in the start, end, mid, and values for each iteration

3	8	12	14	17	45	46	55	57	59	60	61	70	79	92	97	99
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

target = 99							
iteration	0	1	2	3	4	5	6
start	0	9	13	15	16		
end	16	16	16	16	16		
mid	8	12	14	15	16		
value	57	70	92	97	99		

target = 2							
iteration	0	1	2	3	4	5	6
start	0	0	0	0			
end	16	7	2	0			
mid	8	3	1	0	Not		
value	57	14	8	3	Found		