1. Driver/Scribe both do this separately:
Write down a list of any five numbers
Write down mathematical operators to go between each of those numbers (four operators total)

For example, 3+5*9/2-4.
Tell your partner one time what the operators are supposed to be (in turns one right after the other).
Wait 60 seconds
Tell them the five numbers. Ask for the correct result.
If you succeed in correctly remembering the operators, repeat the above with one more number and operator.
When you fail to memorize the values as they are passed to you, write down how many operations you were able to memorize (just write none if you were unable to succeed this the first time.)
Write down how you felt in the failed attempt about being able to understand and memorize your partner's intended code.

(2-4) * 9 / 9 + 1
Attempt 2:

3+8 * 21 / 3 - 5 + 1

This one caught me off guard , not because of the amount of operations and numbers, but because how much bigger the numbers are for the second scenario.

2. Say you have a list of numbers.

4 23 8 42 15 16

For all four answers, write down the swaps that you called.

If you have a method 'swap(int x)' that swaps two adjacent values (the value at index x and the one after it), how many swaps would it take to sort the list by ascending order?
swap(3); //4 23 8 15 42 16
swap(4); //4 23 8 15 16 42
swap(1); //4 8 23 15 16 42
swap(2); //4 8 15 23 16 42
swap(3); //4 8 15 16 23 42
5 swaps.

How about descending?
swap(0); //23 4 8 42 15 16
swap(1); //23 8 4 42 15 16
swap(2); //23 8 42 4 15 16

swap(3); //23 8 42 15 4 16
swap(4); //23 8 42 15 16 4
swap(1); //23 42 8 15 16 4
swap(2); //23 42 15 8 16 4
swap(3); //23 42 15 16 8 4
swap(2); //23 42 16 15 8 4
swap(0); //42 23 16 15 8 4
10 swaps.

Now what if you had a method that would let you swap(int x, int y) two items directly?
It will take less swaps.

How many swaps would it take?
Ascending:
swap(1, 3); //4 15 8 42 23 16
swap(1, 2); //4 8 15 42 23 16
swap(3, 5); //4 8 15 16 23 42
3 swaps.
Descending:
swap(0, 5); //16 23 8 42 15 4
swap(0, 2); //8 23 16 42 15 4
swap(0, 4); //15 23 16 42 8 4
swap(0, 3); //42 23 16 15 8 4
4 swaps.

Driver: Alan Xiao
Scribe: Froilan Zarate
Develop a plan and write pseudocode for this task.
Scribe: How do you plan to find duplicates?
- Create a for loop to check the element with the next element.

Scribe: What will you do when you find them?
- If they are equal, replace it with a filler, then at the end, remove all fillers.

Pay special attention to what happens at the beginning or end of the array list.
Scribe and Driver: Write pseudo code to describe how you would eliminate ALL duplicates, not just adjacent ones.
- for “i” equals 0 to the word length - 1:
    - if the word in the array of index “i” is equal to the word in the array of index “i + 1”
        - Remove the word in the array of index “i”
        - i decrease by 1 i.e. check the same index again

Step 1
Open project Lab11C and run the main() method of class Swap.
The code on lines 28 to 32 is intended to swap neighboring elements. For example,
51 42 2 22 73 77 4 93 15 37
is supposed to turn into
42 51 22 2 77 73 93 4 37 15
But as you can see, it doesn't work.
Now launch the BlueJ debugger. Put a breakpoint at line 28.
Click Step. And then keep clicking Step and observe the program behavior until you can tell why it fails to swap the values.

- It made both elements the number in the index i + 1. i.e. it made both numbers the same.

Tip: To see the contents of the array, double-click on it in the Local Variables pane.

Step 2
Discuss what you learned from observing the program.

- Replacing the element will not save its previous value.

How can you fix your program so that the swapping actually works?

- Create a temp int variable to save the first element.

Scribe: What did you decide?

- We created a temp int variable and stored the first element, replaced the first element with the second element, and replaced the second element with the temp element.

Step 3
Implement your fix and test it in BlueJ.

Submit it to Codecheck to pass all tests.
Driver: Put the fixed code in your lab report.

```java
public class Swap
{
    public static void main(String[] args)
    {
        // Create a random generator with a given seed
        int seed = 54321;
        Random generator = new Random(seed);

        // Make an array of even length between 10 and 20
        int length = 10 + 2 * generator.nextInt(6);
        int[] numbers = new int[length];

        // Fill the array with random numbers between 0 and 99
        for (int i = 0; i < length; i++)
            numbers[i] = generator.nextInt(100);

        // Display all array elements before swapping
        System.out.println(Arrays.toString(numbers));

        // Swap adjacent neighbors
        for (int i = 0; i < length; i = i + 2)
        {
           int temp = numbers[i];
           numbers[i] = numbers[i + 1];
           numbers[i + 1] = temp;
        }

        // Display all array elements after swapping
        System.out.println(Arrays.toString(numbers));
    }
}
```

saved

Step 1
Open project Lab11D and run the main() method of class Lab11D.
Note that a VisualArrayList is exactly like an ArrayList, except it shows you in slow motion what goes on inside.

Step 2
Come up with an algorithm for the following task. We want to swap the first half and the second half of the array list.
For example, A B C D E F should turn into D E F A B C.
You should assume that the array list has an even number of elements (not necessarily 6).
One solution is to keep removing the element at index 0 and adding it to the back.
Each step would look like this:
A B C D E F
B C D E F A
C D E F A B
D E F A B C
Write pseudocode for this algorithm.

- for i equal 0, while i is less than the size of the list / 2:

- Add the first element at the end
- Remove the first element
- Increment i by 1.

Scribe: Which actions need to be carried out in each loop iteration?

- The “add” and “remove” was needed

Scribe: How many iterations are there (for arbitrary even lengths, not necessarily 6)

- The amount of iterations is half the amount of elements in the list.

Step 3

Implement your pseudocode and run it.

Driver: Paste the main method into your lab report.

```
public static void main(String[] args)
{
    VisualArrayList<Picture> list = new VisualArrayList<Picture>();
    list.add(new Picture("a.jpeg"));
    list.add(new Picture("b.jpeg"));
    list.add(new Picture("c.jpeg"));
    list.add(new Picture("d.jpeg"));
    list.add(new Picture("e.jpeg"));
    list.add(new Picture("f.jpeg"));

    // your work here: swap the first and second parts of the array list.


    for (int i = 0; i < (list.size() / 2); i++)
    {
        list.add(list.get(0));
        list.remove(0);
    }
```

Step 4

When you run the code, you will find that there is a lot of movement in the array list. Each call to remove(0) causes n - 1 elements to move, where n is the length of the array. If n is 100, then you move 99 elements 50 times, (almost 5000 move operations). That's an inefficient way of swapping the first and second halves. (Learn more about time complexity at the end of the lab).

Come up with a better way in which you swap the elements directly. Hint: How do you swap the two elements A and D?

- We can swap the element in index “i” with the element in index “i + half the list”

A B C D E F

D B C A E F

D E C A B F

D E F A B C

Write pseudocode for this algorithm.

- for i equal 0, while i is less than the size of the list / 2:
    - Picture temp equals the picture in index i
    - Set the picture in index i, to the picture in index i plus half of the size of the list
    - Set the picture in index i plus half of the size of the list, to the temp picture.

Scribe: Which actions need to be carried out in each loop iteration?

- The “set,” “get”, and “size” method.

Scribe: How do you know the positions of the elements to be swapped?

- For the initial position “i”, it will be “i” plus the size of the array divided by 2.

Scribe: How many iterations are there (for arbitrary even lengths, not necessarily 6)

- The size of the array is divided by 2.

Step 5

Implement your pseudocode and run it.

Driver: Paste the main method into your lab report. Watch how much faster it runs. (This should be pretty obvious since the movement of the array elements takes time.)

```
public class Lab11D
{
    public static void main(String[] args)
    {
        VisualArrayList<Picture> list = new VisualArrayList<Picture>();
        list.add(new Picture("a.jpeg"));
        list.add(new Picture("b.jpeg"));
        list.add(new Picture("c.jpeg"));
        list.add(new Picture("d.jpeg"));
        list.add(new Picture("e.jpeg"));
        list.add(new Picture("f.jpeg"));

        // your work here: swap the first and second parts of the array list.



        for (int i = 0; i < (list.size() / 2); i++)
        {
            Picture temp = list.get(i);
            list.set(i, list.get(list.size() / 2 + i));
            list.set(list.size() / 2 + i, temp);
        }

    }
}
```

Step 6

Add two more letters and run the program again to double-check that it works with any even number of letters. Many more files are provided, and you could add even more letters to run the program