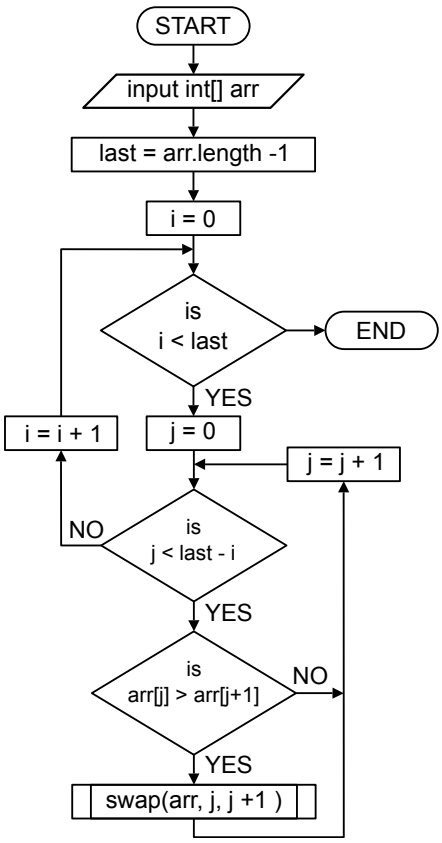
Bubble Sort Algorithm

The Java code for the *bubble sort* algorithm is given below and the corresponding flow chart is given to the right.

```
1  public static void bubbleSort(int[] arr) {
2     int last = arr.length - 1;
3     for (int i = 0; i < last; i++) {
4        for (int j = 0; j < last - i; j++) {
5           if (arr[j] > arr[j + 1]) {
6              // Swap adjacent if wrong order
7              int temp = arr[j];
8              arr[j] = arr[j + 1];
9              arr[j + 1] = temp;
10          }
11       }
12    }
13 }
```



The bubble sort algorithm repeatedly traverses the list, comparing each adjacent pair of elements. If a pair is in the wrong order, it swaps them. After each pass, the next largest unsorted element "bubbles up" to its correct position at the end, so later passes can ignore the already-sorted tail.

The table to the right shows an example of the *bubble sort* algorithm sorting a list with six elements. In the diagram, the dark gray array elements are the elements that were compared, and swapped when necessary.

For the first iteration of the outer loop, `i = 0` and the inner array index, `j`, loops from `0` to the length of the array `-2`, swapping elements as necessary. The `-2` is because we compare each element to the next adjacent element, so if we continued to the array length `-1`, the bounds of the array would be exceeded (we would compare `arr[5]` to `arr[5+1]`, and the last element in the array is `arr[5]`).

For the second iteration of the list, the inner array index, `j`, loops from `0` to the length of the array `-3`. Since the largest element of the array has already “bubbled” up to be positioned in the final location of the array, that element does not need to be checked.

The darker table boundaries show that for each iteration of the outer loop, the inner loop limits become shorter.

		Array						
i	j	4	2	8	6	0	5	
0	0	2	4	8	6	0	5	first iteration
	1	2	4	8	6	0	5	
	2	2	4	6	8	0	5	
	3	2	4	6	0	8	5	
	4	2	4	6	0	5	8	
1	0	2	4	6	0	5	8	second
	1	2	4	6	0	5	8	
	2	2	4	0	6	5	8	
	3	2	4	0	5	6	8	
2	0	2	4	0	5	6	8	third
	1	2	0	4	5	6	8	
	2	2	0	4	5	6	8	
3	0	0	2	4	5	6	8	fourth
	1	0	2	4	5	6	8	
4	0	0	2	4	5	6	8	5th
		0	2	4	5	6	8	

English Name: _____

Bubble Sort Algorithm

Complete the table with each iteration of the bubble sort algorithm.

i	j	**5**	**9**	**3**	**5**	**3**	**6**	**1**	**2**	
0	0	5	9	9	5	3	6	1	2	Outer loop first Iteration
	1	5	3	9	5	3	6	1	2	
	2	5	3	5	9	3	6	1	2	
	3	5	3	5	3	9	6	1	2	
	4	5	3	5	3	6	9	1	2	
	5	5	3	5	3	6	1	9	2	
	6	5	3	5	3	6	1	2	9	
1	0	3	5	5	3	6	1	2	9	Second
	1	3	5	5	3	6	1	2	9	
	2	3	5	3	5	6	1	2	9	
	3	3	5	3	5	6	1	2	9	
	4	3	5	3	5	1	6	2	9	
	5	3	5	3	5	1	2	6	9	
2	0	3	5	3	5	1	2	6	9	Third
	1	3	3	5	5	1	2	6	9	
	2	3	3	5	5	1	2	6	9	
	3	3	3	5	1	5	2	6	9	
	4	3	3	5	1	2	5	6	9	
3	0	3	3	5	1	2	5	6	9	Fourth
	1	3	3	5	1	2	5	6	9	
	2	3	3	1	5	2	5	6	9	
	3	3	3	1	2	5	5	6	9	
4	0	3	3	1	2	5	5	6	9	Fifth
	1	3	1	3	2	5	5	6	9	
	2	3	1	2	3	5	5	6	9	
5	0	1	3	2	3	5	5	6	9	Sixth
	1	1	2	3	3	5	5	6	9	
6	0	1	2	3	3	5	5	6	9	7th
		1	**2**	**3**	**3**	**5**	**5**	**6**	**9**	