**Algorithm 1: Bubble Sort**

Below is a method which will sort an array of integers in ascending order using the bubble sort. Please copy this method into your own personal file to test it with the array.  
**int** array[] = {5,3,7,8,2,1,9,4,10,6};

|  |
| --- |
| **public** **static** **void** BubbleSort(**int** a[])  {  **boolean** loopSomeMore;  **do**  {  loopSomeMore = **false**; // sets loop to false  **for**(**int** j = 0; j < a.length -1; j++)// check when j is less than the length of the array ‘a’, when true add 1 to i  {  **if**(a[j] > a[j+1]) // checks when array ‘a’ of index j is greater than the index in front of the same array, if true then switch places  {  **int** temp = a[j];  a[j] = a[j+1];  a[j+1] = temp;  loopSomeMore = **true**; // sets loop to true to check if there are more numbers to sort  }  }  }  **while**(loopSomeMore);  } |

1. Please add developer comments at the points indicated with a **“//”**. These comments should briefly outline what each line of code is for, and what it accomplishes.
2. In the box below, please provide a written description of how this sort moves through an array sorting it. This Description should highlight the process, it does not need to outline every single adjustment that is made.

|  |
| --- |
| Takes the first index value of the array and checks if it is greater than the index value directly after it. If it is true, it will switch their places then loop again |

1. Can you please copy and paste the sort method into the box below, but please make the adjustments necessary to sort the array in descending order instead of ascending. Please highlight your adjustments in **red bold text.**

|  |
| --- |
| **public** **static** **void** BubbleSort(**int** a[])  {  **boolean** loopSomeMore;  **do**  {  loopSomeMore = **false**; // sets loop to false  **for**(**int** j = 0; j < a.length -1; j++)// check when j is less than the length of the array ‘a’, when true add 1 to i  {  **if**(a[j] **<** a[j+1]) // checks when array ‘a’ of index j is **less** than the index in front of the same array, if true then switch places  {  **int** temp = a[j];  a[j] = a[j+1];  a[j+1] = temp;  loopSomeMore = **true**; // sets loop to true to check if there are more numbers to sort  }  }  }  **while**(loopSomeMore);  } |

**Algorithm 2: Insertion Sort**

Below is a method which will sort an array of integers in ascending order using the Insertion sort. Please copy this method into your own personal file to test it with the array.  
**int** array[] = {5,3,7,8,2,1,9,4,10,6};

|  |
| --- |
| **public** **static** **void** InsertionSort(**int** a[ ] )  {  **int** itemToInsert, j;  **boolean** keepGoing;  **for**(**int** k = 1; k < a.length; k++)  {  itemToInsert = a[k]; // sets the inserted variable to the be the second index of the array  j = k-1; // sets the next index variable to correlate with the for loop but starting at 0  keepGoing = **true**;  **while**((j >= 0) && keepGoing) {  **if** (itemToInsert < a[j] ) // checks if the inserted variable (the one directly in front of the selected array element) is less than the selected array element, if true when swap them {  a[j + 1] = a[j];  j--;  **if**(j == -1) {  [0] = itemToInsert;  }  }  **else** // moves the inserted variable up by 1  {  keepGoing = **false**;  a[j + 1] = itemToInsert;  }  }  }  } |
|  |

1. Please add developer comments at the points indicated with a **“//”**. These comments should briefly outline what each line of code is for, and what it accomplishes.
2. In the box below, please provide a written description of how this sort moves through an array sorting it. This Description should highlight the process, it does not need to outline every single adjustment that is made.

|  |
| --- |
| The first two numbers are selected and compared and sorted in ascending order, then the next number is selected and compared with the second number in the same way and then the second number is compared with the first number sort it in the same way again, then this is repeated by increasing the amount of numbers selected until it reaches the end of the array |

1. Can you please copy and paste the sort method into the box below, but please make the adjustments necessary to sort the array in descending order instead of ascending. Please highlight your adjustments in **red bold text.**

|  |
| --- |
| **public** **static** **void** InsertionSort(**int** a[ ] )  {  **int** itemToInsert, j;  **boolean** keepGoing;  **for**(**int** k = 1; k < a.length; k++)  {  itemToInsert = a[k]; // inserts an index value of k from the array ‘a’ into a new integer  j = k-1;  keepGoing = **true**;  **while**((j >= 0) && keepGoing) // checks if j is greater than or equal to 0 and if the reset are both true  {  **if** (itemToInsert < a[j] ) // checks if the inserted index of the array is less than the array ‘a’ of index j, if true switch the array index of j with the array index directly in front of j  {  a[j + 1] = a[j];  j--;  **if**(j == -1) {  [0] = itemToInsert;  }  }  **else** // if false, the sort has finished and reached the end of the array to which the final array index can be inserted at the end  {  keepGoing = **false**;  a[j + 1] = itemToInsert;  }  }  }  } |

**Algorithm 3: Selection Sort**

Below is a method which will sort an array of integers in ascending order using the Selection sort. Please copy this method into your own personal file to test it with the array.  
**int** array[] = {5,3,7,8,2,1,9,4,10,6};

|  |
| --- |
| **public** **static** **void** SelectionSort(**int** a[])  {  **int** min, minIndex;  **for**(**int** i = 0;i < a.length; ++i)  {  min = a[i]; // at i=0, the first index value will be assigned to a new integer as min  minIndex = i;  **for** (**int** j = i + 1; j < a.length; ++j) // j is set to be the index directly in front of the current index and the loop repeats while j is less than the index length of the array  {  **if** (a[j] < min) // at j=1, the array index value of index j (0) will be compared with the minimum integer, if the array is less than the minimum, the new minimum will be set to be the array index of j (0) and the index j will be stored to a new integer  {  min = a[j];  minIndex = j;  }  }  a[minIndex] = a[i]; // the array index value of the stored minimum index is set to be the array of index I  a[i] = min;  }  } |

1. Please add developer comments at the points indicated with a **“//”**. These comments should briefly outline what each line of code is for, and what it accomplishes.
2. In the box below, please provide a written description of how this sort moves through an array sorting it. This Description should highlight the process, it does not need to outline every single adjustment that is made.

|  |
| --- |
| The lowest number in the array is selected and moved to the front, then the first number is excluded from the search and the next smallest number is selected and moved to the second index of the array. |

1. Can you please copy and paste the sort method into the box below, but please make the adjustments necessary to sort the array in descending order instead of ascending. Please highlight your adjustments in **red bold text.**

|  |
| --- |
| **public** **static** **void** SelectionSort(**int** a[])  {  **int** min, minIndex;  **for**(**int** i = 0;i < a.length; ++i)  {  min = a[i]; // at i=0, the first index value will be assigned to a new integer as min  minIndex = i;  **for** (**int** j = i + 1; j < a.length; ++j) // j is set to be the index directly in front of the current index and the loop repeats while j is less than the index length of the array  {  **if** (a[j] > min) // at j=1, the array index value of index j (0) will be compared with the minimum integer, if the array is greater than the minimum, the new minimum will be set to be the array index of j (0) and the index j will be stored to a new integer  {  min = a[j];  minIndex = j;  }  }  a[minIndex] = a[i]; // the array index value of the stored minimum index is set to be the array of index I  a[i] = min;  }  } |