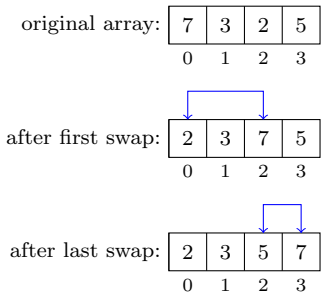
Lab 05 - Arrays & Sorting

Instructions:

- **The lab requires writing a complete cpp file within an hour. It requires completing set of tasks.**
- **Accompanying this file is a template cpp file that you must modify. You cannot include additional libraries to or remove any libraries from the file. All other modifications are allowed.**
- **Your submission must be submitted to the Labs directory of your github repository and/or as an attachment on Google classroom under the Lab05 assessment. The file must remain a cpp file.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the lab.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, AT THE BEGINNING OF YOUR SUBMISSION, ADD A COMMENT THAT CONSISTS OF YOUR NAME AND THE DATE

Your objective is to write an in-place sorting algorithm function, which is an algorithm that sorts an array with little or no additional space [extra variables or arrays], and generate a trace table of a caller of the function. The sorting algorithm you will define is known as the selection sort algorithm. It finds the index of the element that contains the minimum value from the unsorted portion of the array, and then, moves the value to the end of the sorted portion of the array by swapping the values of the elements only if they are different [have different indices]. It repeats the above process until the unsorted portion is empty [note: the end of the sorted portion and the beginning of the unsorted portion is the same element for each cycle of the selection sort]. For instance, the diagram below shows the swaps performed using the selection sort on the array below



To accomplish the objective, complete the following tasks:

- ☐ define a void function named `Print()` that takes a double array parameter and an int parameter respectively. Given that the int parameter represents the size of the array parameter, the function displays the values of each element of the array parameter with a space between each value on their own line.

- ☐ define a void function named `Swap()` that takes two double reference parameters. It swaps the values of the parameters.

- ☐ define a void function named `SwapIfDifferent()` that takes a double array parameter and two int parameters respectively. Given that the int parameters are valid indices of the array parameter, if the int parameters are not equal, the function swaps the values of the elements of the array parameter whose indices are equal to the values of the int parameters. Otherwise, it does nothing.

- ☐ define an int function named `MinimumIndex()` that takes a double array parameter and two int parameters respectively. Given that the first int parameter represents the size of the array parameter and the second int parameter is a valid index of the array parameter, the function returns the index of the element of the array parameter that has the minimum value of the elements with indices ranging from the value of the second parameter to the size of the array parameter.

- ☐ define a void function named `SelectionSort()` that takes a double array and an int parameter respectively. Given that the int parameter represents the size of the array parameter, the function calls both the functions `MinimumIndex()` and `SwapIfDifferent()` for each index of the array parameter in order [starting with 0 and ending with the size of the array minus 1]. You must perform all operations correctly so that the array is sorted when the function terminates.

- ☐ in the main function, initialize a double array to $\{8, 2, 9, 5, 1\}$, call the `Print()` function with the array as the argument. Afterwards, call `SelectionSort()`, and then, `Print()` with the array as their respective arguments.

- ☐ write a trace table list as a comment after the main function of the function caller `SelectionSort(a,5)` where `a` is an array equal to {`8,2,9,5,1`}