**Reflection**

The program is designed to load bids from an Excel spreadsheet and then, based on user input, either sort them using a selection sort or a quick sort. Initially, the program encountered issues opening the Excel file, but adding it directly to the solution folder allowed for a relative path to be utilized, enabling the file to open correctly. The partition function works in conjunction with the quickSort function; it rearranges the bids by comparing them to the middle of the range, or the "pivot." By using the partition function to return the index of the middle element, quickSort is then called recursively until all the bids are sorted from smallest to largest title. The selectionSort uses nested for loops to iterate through the bids and identify the smallest elements. After identifying the smallest elements, they are rearranged using the swap functionality. Finally, there is a switch method for the menu, allowing users to enter a choice and then return to the menu once the choice has been executed.

**Pseudocode**

Function PARTITION(bids, begin, end)

Set middlePoint = begin + (end - begin) / 2

Set pivot = bids[middlePoint].title

Set done = false

While NOT done:

While bids[begin].title < pivot:

Increment begin

While bids[end].title > pivot:

Decrement end

If begin >= end:

Set done = true

Else:

Swap bids[begin] and bids[end]

Increment begin

Decrement end

Return end

Function QUICKSORT(bids, begin, end)

If begin >= end:

Return // Base case: already sorted

Set mid = PARTITION(bids, begin, end)

// Recursively sort left partition

QUICKSORT(bids, begin, mid)

// Recursively sort right partition

QUICKSORT(bids, mid + 1, end)

Function SELECTIONSORT(bids)

For i = 0 to size of bids - 1:

Set min = i

// Find the smallest element in the remaining unsorted section

For j = i + 1 to size of bids:

If bids[j].title < bids[min].title:

Set min = j

// Swap the smallest found element with the current position

Swap bids[i] and bids[min]