Submission Form

Please ensure you fully complete all FOUR questions on this form before submitting your assignment.

1. Please list the programming constructs you have used.

For loops While loops If statements else if statements else statements

2. Describe in simple language how your program works **and include the name of the type of sort this is.**

First thing we do is create the "numberList" array at a size of 10 values and the "suffix" array at a size of 4 with already initialized values. Using this it then calculates the integer "len" meaning length by taking the size of the array in bytes, and divides it by the size of an element in the array in bytes. We can then use this value for anything that needs to iterate the entire length of the array, as it will calculate based upon the array size. We then have a for loop that iterates for as long as the length of the array using our calculated length variable. I've used a FOR loop here as we are technically given the number of iterations we need to complete due to the array we're inputting into being a fixed size(regardless of what that size is), we just have to calculate the size as an integer beforehand for usage as a variable in our loops. The first thing we do is write our function prototype and then call a function called ordinalSuffix. This function essentially takes the last digit of the number of entry we're on (ie: entry no 1) and uses an if statement to give variable "x" a number corresponding to the correct position of the string we need in our suffix array, and then returns that value for use in our input line. So essentially, for almost all numbers ending in 1 we use the "st" suffix, which is position 0 in our array. So on our number 1 item, our x is set to 0 because we need to grab the "st" at position 0 from the array. We then use the x as the position in the array and get the correct suffix. There are exceptions to this, but they have been accounted for by having a second condition be true in the case of 11,12 and 13 who despite their ending number still make use of "th" as a suffix. The next line asks the user to input the item price, displaying a corresponding item number starting at 1 with the correct ordinal suffix using the array "suffix" and variable" x". While the user enters a number that is not a valid double data type it will be treated as invalid an error message will be received and the user will be asked to enter another value until 10 valid numbers are given. Each valid number entered is added to the numberList array. We then make use of an **insertion sort** on the array, I used insertion sort as the program was intended for 10 items and this would likely be the quickest sort for an array of this size, and should the array be expanded would still be sufficiently fast (though at much larger sizes a quicksort will be better).

The sort basically takes a temporary value - I called it temp as it's temporarily set as equal to a value in the array (starting at the second value as it takes the first value as the basis for the sort) using this it compares the first and second values and moves the second value if higher than the first, to the position ahead, if not it will remain where it is as the second value. Essentially it moves the positions of the values in the array as necessary to any place in the array after comparison hence 'insertion'. In the next pass it compares the 3rd value to both the first and second value in the array and inserts accordingly. In comparison to other sorts we compare each element with all previous sorted elements . This is rather than swapping adjacent values as in some other sorts. This iterates through the entire length of the array until it is fully sorted.

Finally it prints "Sorted list" followed by the whole array, which due to our sort will be in ascending order. It does this using a for loop and using our calculated length to ensure it prints all values. The use of setprecision here is perhaps unnecessary but is used so that a number of up to 15 characters can be displayed in full for accuracy and ease of reading, this ensures almost all numbers within reasonable bounds. It was used as without it when using cout scientific notation is used for any number greater than 6 characters, which in turn means that for any number greater than 999999 information can be lost if the user converts back from scientific notation, this something I wished to avoid.

3. Please describe below any additional features that you've included in your programme, if none what would you add if you had more time?

I've included additional input validation to ensure that the values entered could not be any form of data other than a double, which essentially means all the values entered will be numeric which was the only specification and therefore valid.

I also decided to add a function that adds an ordinal suffix , which is essentially the bit of text after a number like "st", "nd" etc. It uses an array of strings and an if statement. It's fairly simple but it's robust in the sense that it accounts for the exceptions to the rule using OR (IE: 11 ends in a 1 but still uses a "th" suffix) and works beyond the scope of the program, up into the hundreds. Just wanted to give another example of how an array could be used. I made this a function as it could potentially be applied to the output, all you would swap would be i for k, which is the counter in the output loop. Though I did not do that, that was because I did not have a specified situation for the output, but if given a large unsorted list and I wanted to know the order of finishers in a race for instance then having the suffixes would be nice (23rd, 34th etc).

In terms of additional features if I wanted to make the program as robust as possible in order to account for user error I could give the user the option to enter a specific character after an entry to remove the previous value and give them the chance to renter that value. This would be good because while it's not much of a problem for this program as it only has to account for 10 values so restarting the program isn't a huge problem, if somebody had a long list they wanted to enter sequentially(which is unnecessary in this instance as the order doesn't matter, but if they did) then making a mistake and having to re-enter the entire thing would be an annoyance, so having a remove previous value button is a convenience more than anything.

If allowed to fundamentally change the program I would probably use a vector as opposed to an array, due to the fact that the user at present would have to have access to the backend/code of the program in order to change how many items were entered by manually changing the array size due to the lack of variable size using arrays in standard C++, when using a vector which would give us access to variable size.

4. Please paste your code into the box below, ensuring it has been formatted correctly

```
#include <iostream>
using namespace std;
#include <iomanip>
int main() {
    double numberList[10]; //create our array
    string suffix[4]{ "st","nd","rd","th" }; //create second array

    int len = sizeof(numberList) / sizeof(numberList[0]); // get numeric value for the length of
the array for usage in calculations, mainly determining loop lengths.

    for (int i = 0; i < len; i++) {
        int ordinalSuffix(int, int);
        int x = ordinalSuffix(i, x);
        cout << "Enter " << i + 1 << suffix[x] << " number ";
        cin >> numberList[i];
        while (!cin) { //any non-double data type will set this off
            cout << "That is not a valid number, please enter a valid number." << endl;
            cin.clear(); //reset cin so it doesn't read as incorrect forever.
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); //clear entire input buffer so a
new input can be taken
            i--; // subtract one from the number of iterations as the incorrect value should not be
treated as valid, we only want 10 valid entries
        }

        //insertion sort being used here, likely the fastest for our 10 items.
        double temp = numberList[i]; //set temp to  value at pos i
        int j = i - 1; // first run through j is set to -1, which is array position 0. This is used
for our while loop.
        while (j >= 0 && temp <= numberList[j]) //while j is greater than or equal to 0 (not at the
first value in our array) and that the value of temp is lesser than or equal to numberList[j]which
is the previous value.
        {
            numberList[j + 1] = numberList[j]; //due to temp being lesser move item at j up by 1
position and set it to the array at the higher position.
            j--; //count j down to ensure while loop condition is met
        }
        numberList[j + 1] = temp; //ensure that temp is back to our current position in the array as
j was set to our current position-1 at the start.
    }
    //print array
    cout << "SORTED LIST" << endl;
    for (int k = 0; k < len; k++) {
        cout << numberList[k] << setprecision(15) << endl;
    }
}

int ordinalSuffix(int i, int x) {
    int lastDigit = (i + 1) % 10; // i+1 because the computer starts counting at 0, but people don't
so we don't want a 0th item.
    int exceptions = (i + 1) % 100; // check exceptions for 11,12 and 13 because they use "th"
despite their ending number
    if (lastDigit == 1 && exceptions != 11) {
        x = 0;
    }
    else if (lastDigit == 2 && exceptions != 12) {
        x = 1;
    }
    else if (lastDigit == 3 && exceptions != 13) {
        x = 2;
    }
    else { x = 3; }
    return x;
}
```