Project 4 Additional FAQ

Please read this FAQ before posting questions on piazza.

Q1) for countFloatingPointValues(), are leading 0s valid for a float?

A1) Yes, all your floating-point algorithm needs to check for is no + or -, no commas and then all digit characters and an optional single decimal point. So yes, 0000076 would pass. 00.00.76 would not. 7600+00 would not. 76.0000 would pass.

Q2) For the shiftLeft function in our project, do we need to use & (i.e. int myFunc(string& array[ ])) to directly modify the contents of the array parameter in our function? If we do not need &, for future reference, how could we modify and return the contents of the given array parameter without changing the contents of the original array?

A2) You do not need to pass an array argument using the & operator to modify the contents of the original array. Since an array argument in a function is just the memory address, whenever you modify an element in the array argument, it essentially already acts like it is being passed by reference and the contents of the original array will reflect the modification.

Q3) For the count floatings, should the function consider a data entry such as "7." as a valid float, even though the decimal is superficial?

A3) The specs say that the decimal point is optional, and in the example on the very last page, "123", "456", "789" are all considered floating point numbers. "7." is also valid since it is composed of only numbers and an optional decimal point.

Q4) For the locateMaximum function, are we suppose to make sure that the items in the array are comprised of characters rather than digits? Or are digits fine for the comparison?

A4) You do not need to validate what the items in the array are. The strings can be composed of numbers, letters or special characters ("@", "#", etc).

Q5) Can I use cmath for Project 4?

A5) No, and you don't need to.

Q6) For our main function, as long as it compiles, can we turn in anything we want (the same as project 3)?

A6) Yes.

Q7) I'm confused by what the spec means when it says "return the number of times the placeholder *value* was used" if the amount to shift by is 0, and the placeholder value is the same as the value in the array, would that value in the array that's equal to the placeholder value still be counted? (ie array contains one value "foo" and placeholder value = "foo", amount to shift is 0, n is 1 -- would placeholder count be 1?)

A7) The placeholder count would be 0 since you never shifted left and never used the placeholder. The "foo" already being there is not due to our function, so we will not consider that in our placeholder count.

Q8) Should this be the correct thing to happen when n is smaller than the size of the array? We define an array name[5]={"Andrew","Bethany","Christie","Danny","Edward"}, and then call shiftLeft(name, 3, 2, "foo"). The function returns 2 and name is changed to {"Christie","foo","foo","Danny","Edward"}.

A8) When provided an n smaller than the actual array size, you should only manipulate those particular elements. So for:

```
string name[5]={"Andrew","Bethany","Christie","Danny","Edward"};
assert( shiftLeft(name, 3, 2, "foo") == 2 );
```

will wind up with {"Christie","foo","foo","Danny","Edward"}.

Q9) Q: Why should we use || for or and && for and? Wouldn’t and, or as words, it easier to read than || &&.

A9) Over the years, the C++ language standard, like any living language, has grown and evolved. It takes time for all the tool vendors to support the new things. I would recommend that you work with the pieces that have been around for a while. I like && and || myself over and and or.