Q3

Djuro Radusinovic

171044095

1)

Reversing a string has a base case where the end of the string is reached and the memory for the string to be returned is reserved. Otherwise it is has a recursive call with the position of the string incremented. Also when the space is reached or the beginning of the string is reached another recursive helper method is called that returns the first word of a string passed. Substring method is not used because It has linear complexity and I don't know if it uses a loop inside or not. Smaller problems are finding these substrings.

2)

Determines if the word is elfish recursively. It has a helper method here again which has three parameters that are boolean(e,l,f) each of them indicates if the 'e', 'l' or 'f' occurred. Again position of the string is passed and incremented for each recursivecall. Base case is when the end of the string is reached and there it will return if all letters are found(e,l,f).

3)

This algorithm is used for sorting and is usually implemented using two loops. Instead of that I implemented two recursive calls. One which makes sure each element of the array is visited. The other helper recursive function finds the minimum index and places it at the beginning of the subarray passed. Of course it is also implemented recursively and the execution will stop when the end of the subarray is reached and the swapping of the element in the minimum index will be executed.

4)

For prefix evaluating I visited the string from right to left. Again, didn't use the substring method. Here I used a rule that for the string passed as a prefix expression every operator and operand are separated with a space. I am using a substring recursive method I have written to get the operator or the operand. I check if it is operator or the operand and add it to the stack if it is an operator, otherwise pop two operators and evaluate them and than add the result to stack. Base case would be when the beginning of the prefix's string's is reached. Result would be the only element of the string and it will be popped and returned as a result.

5)

Same as for the prefix evaluation with the difference that I inspected the string from left to right and the operator's in the stack changed position. Logic is the same and the base condition is when the end of the expression is reached.

6)

Printing these elements in clockwise took a method that has information about the upper limit, right limit, left limit and bottom limit. Problem is reduced in the was so that the subarray is always reduced using these four indicators. At the moment at which one of our indexes for the current element in the array is out of bounds we are returning from the method. Also there are indicators for the direction our array is moving when printing(it could have been a simple enum value but I decided to use boolean values yet again which won't change the result at all just would make the code a bit cleaner.

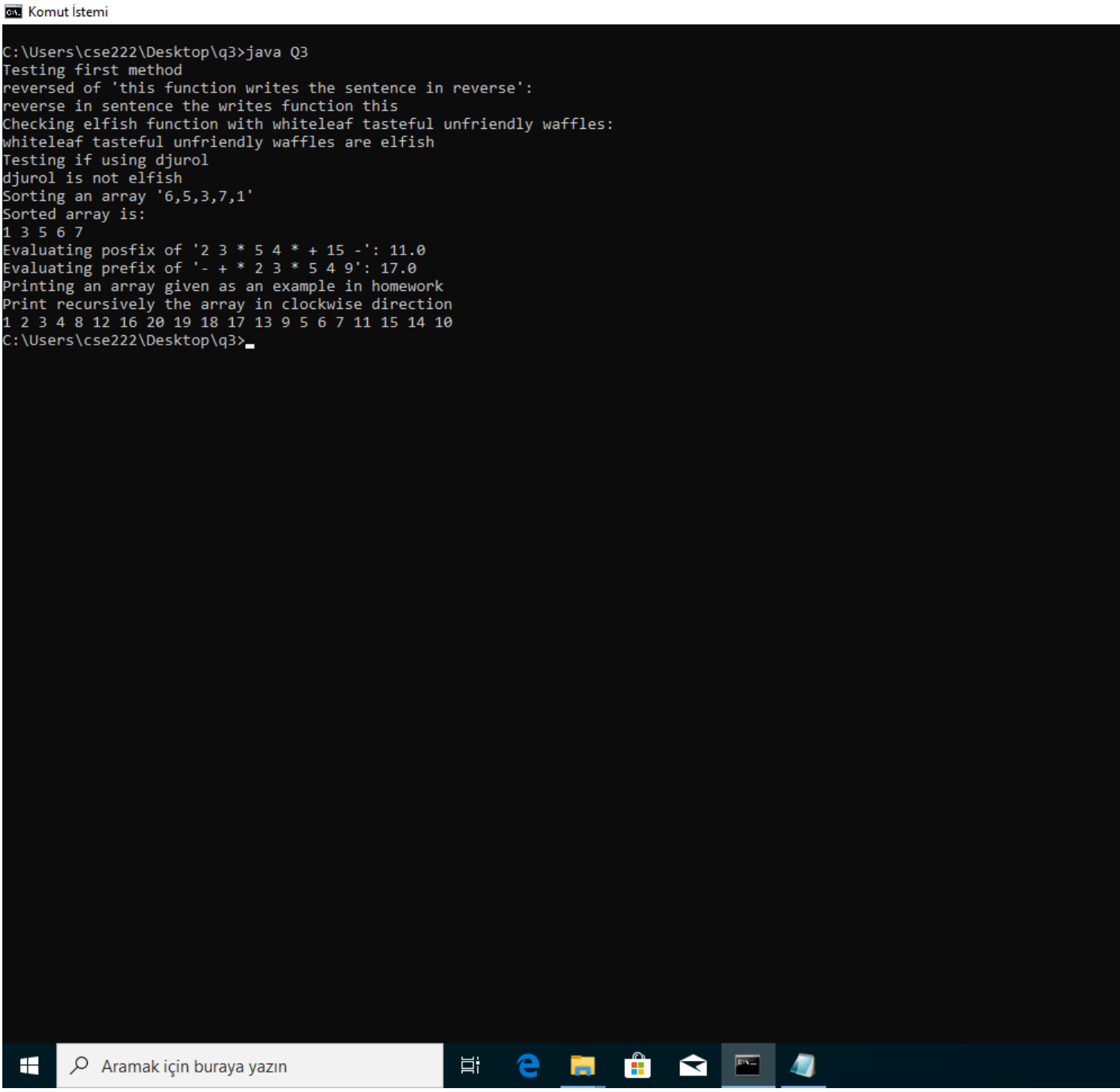
Probem solution approach

Since this part was not about object oriented programming but about recursion I just implemented all the methods at the same place and ran the main function so that I would test these methods. If oop was used the only thing I could have done is encapsulate helper recursive methods I have written. Main methods are the methods that have only one parameter and those are the ones that the user is supposed to call.

Testing

Test Scenario	**Expected Results**	**Actual Results**
Using reverse_rec method to reverse the string	Since input is 'this function writes the sentence in reverse' output should be 'reverse in sentence the writes function this'	As expected
Checking if the word is elfish using words 'tasteful unfriendly waffles are elfish'	They should be elfish	As expected
Checking if 'djurol' is elfish	Should not be elfish	As expected
Sorting an array '6,5,3,7,1'	Array should now be 1,2,5,6,7	As expected
Evaluating postfix string: 2 3 * 5 4 * + 15 -'	Result should be 11.0	As expected
Evaluating prefix of '- + * 2 3 * 5 4 9'	Result should be 17.0	As expected
Printing a 2d array clockwise Array given is: {1,2,3,4} {5,6,7,8} {9,10,11,12} {13,14,15,16} {17,18,19,20}	Output should be 1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10	As expected

Running command and results



I didn't make a class diagram since there is no op design here as it is just method implementation to test the knowledge on recursion.