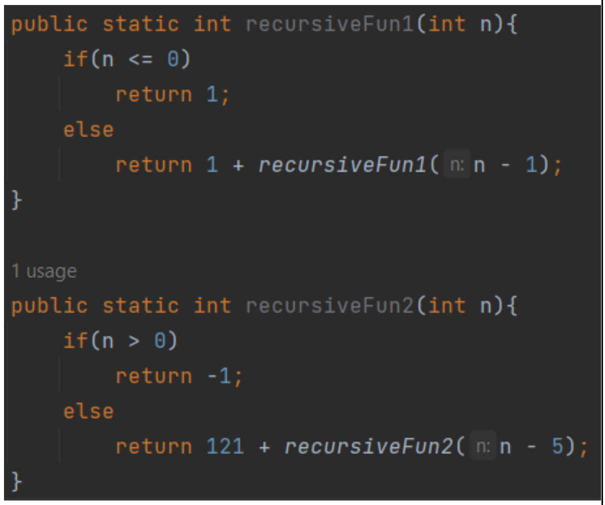
American University of Armenia

CS 121A&B - Data Structures

Spring 2024

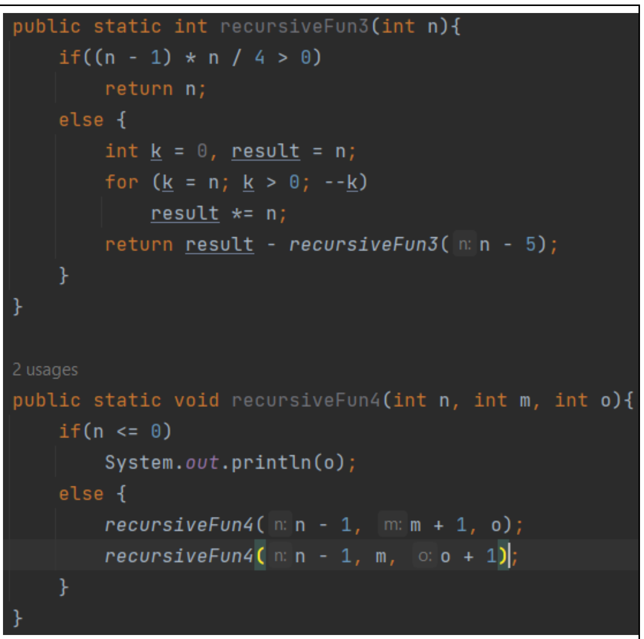
PSS 2

Problem 1. Find time and memory complexities for given functions.



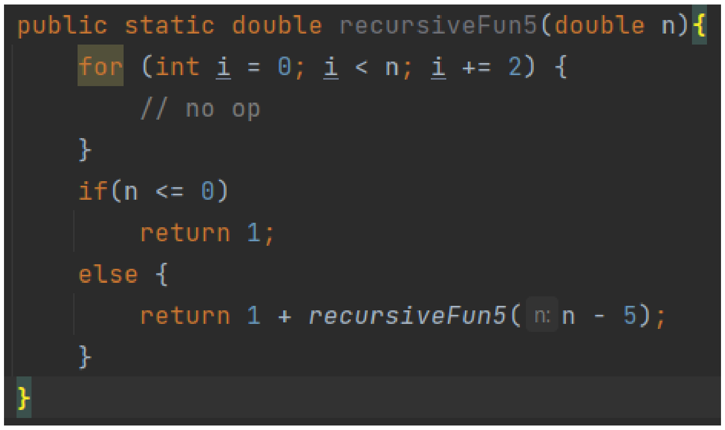
```
public static int recursiveFun1(int n){
    if(n <= 0)
        return 1;
    else
        return 1 + recursiveFun1( n: n - 1);
}

1 usage
public static int recursiveFun2(int n){
    if(n > 0)
        return -1;
    else
        return 121 + recursiveFun2( n: n - 5);
}
```



```
public static int recursiveFun3(int n){
    if((n - 1) * n / 4 > 0)
        return n;
    else {
        int k = 0, result = n;
        for (k = n; k > 0; --k)
            result *= n;
        return result - recursiveFun3( n: n - 5);
    }
}

2 usages
public static void recursiveFun4(int n, int m, int o){
    if(n <= 0)
        System.out.println(o);
    else {
        recursiveFun4( n: n - 1,  m: m + 1, o);
        recursiveFun4( n: n - 1, m,  o: o + 1);
    }
}
```



```
public static double recursiveFun5(double n){
    for (int i = 0; i < n; i += 2) {
        // no op
    }
    if(n <= 0)
        return 1;
    else {
        return 1 + recursiveFun5( n: n - 5);
    }
}
```

Problem 2. Reverse an input array of integers recursively. Give Big-Oh estimates of the running times and memory consumption.

Problem 3. In the following table, Specify the complexities of the highlighted sorting algorithms in terms of time {best, average, worst} cases and memory.

Table 1

Sorting	Time complexity		
	Best	Worst	Space
Insertion	Ω(?)	O(?)	O(?)
Selection	Ω(?)	O(?)	O(?)
Bubble	Ω(?)	O(?)	O(?)
Quick	Ω(?)	O(?)	O(?)
In-place Quick	Ω(?)	O(?)	O(?)
Merge	Ω(?)	O(?)	O(?)
Bucket	Ω(?)	O(?)	O(?)
Radix	Ω(?)	O(?)	O(?)

Problem 4. Modify the selection sort algorithm to place the smallest number in the middle, followed by the second smallest, and so forth. When you reach the edge of the array, position the next smallest number at index 0, then index 1, and so forth until you reach middle - 1 index. The visualisation in figure 2 shows us the result of the algorithm on figure 1 array.

19	5	23	11	13	52	4	44	6	28

Figure 1: Array of integers of size 10.

19	23	28	44	52	4	5	6	11	13

Figure 2: Sorted array using modified selection sort.

Problem 5. Modify the selection and bubble sort algorithms to a recursive approach.

Problem 6. Implement atoi(string to int) function recursively. Give Big-Oh estimates of the running times and memory consumption of that function.

Problem 7.

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

The test cases are generated so that the length of the output will never exceed 10^5.

Input: s = "3[a]2[bc]"
Output: "aaabcbc"

Input: s = "3[a2[c]]"
Output: "accaccacc"

Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"

Constraints:

- 1 <= lengthOfInputString <= 30
- s consists of lowercase English letters, digits, and square brackets '[]'.
- s is guaranteed to be **a valid** input.
- All the integers in s are in the range [1, 300].

Give Big-Oh estimates of the running times and memory consumption.