# OLQ4

## Study Guide

Questions from previous OLQs

Questions from Homework 1 and Homework 2

Questions about Coding Assignment 1

Identifying big O, big theta and big omega graphically.

OLQ4 will have an Insertion Sort question like the questions in Homework 2 – Merge Sort will be on OLQ5. Please see the "Insertion Sort Review for OLQ4" video under the Review Materials module in Canvas.

Conditional Compile

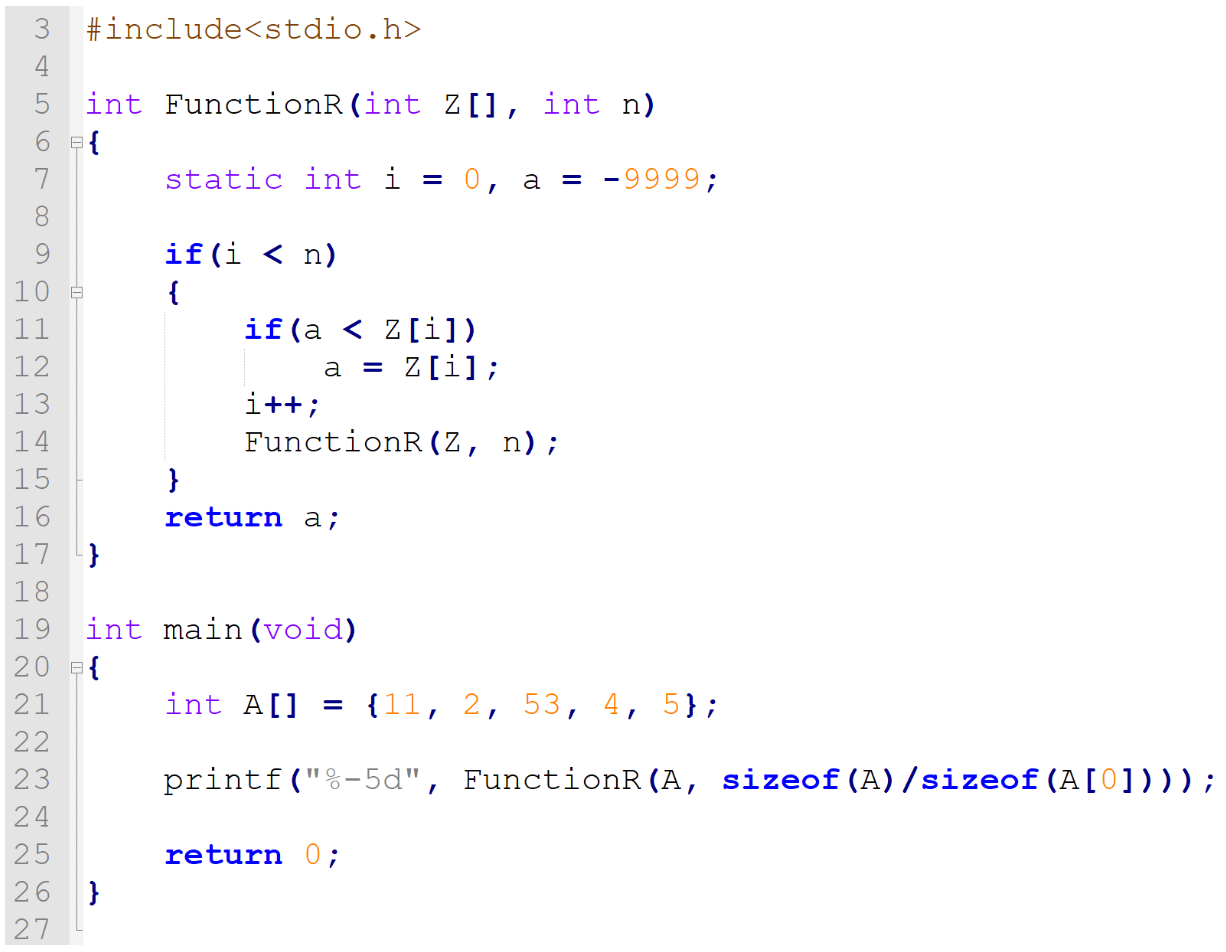
How is the code affected by using #ifdef, #elif, #else, #endif with -D on the command line with gcc?

Questions about recursion

What makes a function recursive? When/how does a recursive function know to end? What is a pro (positive feature) of using recursion and what is a con (negative feature) of using recursion?

You will be given a recursive program. You will be given input and will need to follow the recursion to determine the output.

Here is an example problem – we did one like this in class on Thursday.



```c
#include<stdio.h>

int FunctionR(int Z[], int n)
{
    static int i = 0, a = -9999;

    if(i < n)
    {
        if(a < Z[i])
            a = Z[i];
        i++;
        FunctionR(Z, n);
    }
    return a;
}

int main(void)
{
    int A[] = {11, 2, 53, 4, 5};

    printf("%-5d", FunctionR(A, sizeof(A)/sizeof(A[0])));

    return 0;
}

```

main() calls FunctionR() with two parameters – the array and the number of elements in the array

1st call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is initialized to 0 once (static variables are only initialized once and maintain their value between function calls)

a is initialized to -9999 once (static variables are only initialized once and maintain their value between function calls)

if (i < n) => (0 < 5) => true

if (a < Z[i]) =>Z[i]=Z[0]=11 so (-9999 < 11) => true

a = Z[i]  so a is set to Z[0] which is 11

i is incremented from 0 to 1

FunctionR() is called with the array and n (which is 5)

When 2nd call to FunctionR() returns, a will be returned to main()

2nd call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is 1

a is 11

if (i < n)  => (1 < 5) => true

if (a < Z[i]) => Z[i]=Z[1]=2 so (11 < 2) => false (a stays at 11)

i is incremented from 1 to 2

FunctionR() is called with the array and n (which is 5)

When 3rd call to FunctionR returns, a will be returned to 1st call of FunctionR

3rd call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is 2

a is 11

if (i < n) => (1 < 5) => true

if (a < Z[i]) => Z[i]=Z[2]=53 so (11 < 53) => true

a = Z[i]  so a is set to Z[2] which is 53

i is incremented from 2 to 3

FunctionR() is called with the array and n (which is 5)

When 4th call to FunctionR returns, a will be returned to 2nd call

4th call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is 3

a is 53

if (i < n) => (3 < 5) => true

if (a < Z[i]) => Z[i]=Z[3]=4 so (53 < 4) => false

i is incremented from 3 to 4

FunctionR() is called with the array and n (which is 5)

When 5th call to FunctionR returns, a will be returned to 3rd call

5th call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is 4

a is 53

if (i < n) => (4 < 5) => true

if (a < Z[i]) => Z[i]=Z[4]=5 so (53 < 5) => false

i is incremented from 4 to 5

FunctionR() is called with the array and n (which is 5)

When 6th call to FunctionR returns, a will be returned to 4th call

6th call to FunctionR

Z = {11,2,53,4,5} and n = 5

i is 5

a is 53

if (i < n) => (5 < 5) => false

return a which is 53

6th call returns a value of 53 to 5th call. 5th call returns a value of 53 to 4th call. 4th call returns a value of 53 to 3rd call. 3rd call returns a value of 53 to 2nd call. 2nd call returns a value of 53 to 1st call. 1st call returns a value of 53 which is printed to the screen by main().

Note that the return value of `FunctionR()` here does not matter until the recursion returns to `main()`. The return value of `FunctionR()` is printed in `main()`. None of the other calls to `FunctionR()` actually capture the return value of the recursive call. As each value is examined, if it is the current max, then it is stored in a. Variable `a` is static and is maintained between calls to `FunctionR()`. The final recursive step going back to `main()` returns the value of a to be printed. This recursion takes advantage of `a` being `static`.

The actual answer you would write on the quiz in this case is

```
53
```

Please read the directions carefully. If the directions state that you must show your work on the quiz, then you will need to write down how you arrived at your answer – not just write down the answer.