

CSI 201 - Function Practice

CSI 201

A bunch of scenarios are presented to help you quickly test your function skills. All skills prior to this sheet may still be involved. For example, if the function asks you to average a vector, you must still be able to construct the function and average a bunch of numbers in a vector. The parameters that come into a function are often called the input of the function. The return of the function is often called the output of the function. In these descriptions, you'll see that I describe things coming into the function (parameters), the function receiving information (parameters) and input to the function (again parameters). These terms do not imply cin or console input. Likewise, you'll see that I describe things coming out of the function (returned), the output of the function (returned) or that the function computes something (it is implied the function then returns that value). As an example, if I were to describe the prompt for sqrt I might say: Write a function that gets a number and then computes the sqrt of that number. Or, write a function whose input is a number and whose output is the sqrt of that number. Sometimes the parameters and the return are implied and not spoken. Write a function that outputs "hello world" to the console. This is a function that has no return and no parameter list. What can be confusing here is that the word output is overloaded to mean, output to the console and output of the function. Look for the console keyword, or the user keyword to know when we're talking about interacting with the user (cout/cin) and not just talking about how the function should operate.

1 Call By Value *-Answers at the bottom*

In this section, all parameters should be call by value. This ensures (to the passer) that the incoming variables are untouched and unchanged.

1. Write a function to average three numbers.
2. Write a function that takes in a yard and outputs the number of feet in that yard. There are 3 feet in a yard.
3. Write a function that takes in Celcius and converts it into Fahrenheit (as output).
 $C = (F - 32) * 5 / 9$
4. Write a function that takes in a number n and computes the sum from 1 to n.
5. Write a function that takes in a number n and computes n factorial (which is the product of all the numbers from 1 to n).

6. Write a function that takes in a number n and computes the n th Fibonacci number. If n is 0 or 1, the Fibonacci number is 1. For all other values of n , the Fibonacci number is equal to the previous two Fibonacci numbers added together.

2 Call By Reference *- We didn't cover this SP18*

In this section, all parameters should be call by reference. This means that the variables declared by the function can actually be aliases of passed in variables. It also means, pure values cannot be based to that parameter. For example, if the integer is a call by reference variable, you cannot pass the literal "13" to that function directly. You must give it a variable with the value of 13 instead.

1. Write a function to average a vector of doubles. Use call by reference to avoid copying the entire list.
2. Write a function that takes in two vectors. The function should return true if the second vector is the reverse of the first vector. Otherwise, it returns false. Use call by reference to avoid copying the entire list.
3. Write a function that takes in two vectors. The function changes the second vector to make it the reverse of the first vector. Call by reference on the first parameter avoids a copy, on the second parameter it allows changes.
4. Write a function that takes in a vector and returns the size of that vector. (This is a silly task. Explain why.) Use call by reference to avoid copying the entire list.
5. Write a function that takes in two call by reference integers. The function then swaps the values in those integers.
6. Write a function that takes in a vector of strings, randomly selects two of those strings and returns the concatenation of those strings. (Do not srand in this function.)

① // Avg 3 #s CALL BY VALUE ANSWERS

```
// Assume integers
int avg(int a, int b, int c) {
    return (a+b+c)/3;
}
```

```
// Yardage to feet
int yardToFeet(int yards) {
    return yards * 3;
}
```

```
③ // F to C    C = (F-32) * 5/9
```

```
double CToF(double F) {
    return (F-32) * 5/9;
}
```

#3 - Flipped

```
// What about C to F?
// MUST USE ALGEBRA F = C * 9/5 + 32
double FToC(double C) {
    return C * 9/5 + 32;
}
```

```
④ // Sum from 1 to n
int sumOneToN(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum = sum + i;
    }
    return sum;
}
```

```
⑤ // Compute n!
int factorial(int n) {
    if (n < 0 || n > 13) {
        return 0;
    }
    int product = 1;
    for (int i = 2; i <= n; i++) {
        product = product * i;
    }
    return product;
}
```

```
⑥ // Fn = Fn-1 + Fn-2 F0 = F1 = 1
int fib(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    int f_n = 1; // current
    int f_m = 1; // previous
    for (int i = 1; i < n; i++) {
        int next = f_n + f_m;
        f_m = f_n;
        f_n = next;
    }
    return f_n;
}
```