

# CSE 240 Homework 2

Chen, Acuña

1/8/2021

---

## Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures, including programming paradigms, the structure of programming languages, and the differences between a macro and a procedure. By the end of the assignment, you should have

- exercised typing systems and operations of different typing systems.
- understood differences between the execution models of a macro and a function.
- gotten started with the programming environments Visual Studio and GCC.

This assignment is related to the outcomes 1-2 and 1-3 listed in the syllabus:

- learn strong vs. weak typing in computer programming languages
- understand the control structures of functional, logic, and imperative programming languages.
- understand the execution of functional, logic, and imperative programming languages.

**Optional Reading (6e):** Read chapter 1, chapter 2 (sections 2.1, 2.2, and 2.3), appendix (sections B.1 and B.2), and course notes (slides).

You are encouraged to ask and answer questions on the course Slack. (However, **do not share your answer.**)

## Pre-requisite

In addition to using GCC as describes in Homework 1, you will need to have installed Visual Studio on your computer (or otherwise have access to Visual Studio to complete the later questions in this assignment).

Note that installing Visual Studio and getting use to it may take a little time so this homework has fewer questions than the previous one.

## Programming Exercise (50 points)

1. Macros are available in most high-level programming languages. The body of a macro is simply used to replace a macro-call during the preprocessing stage. A macro introduces a "true inline" function that is normally more efficient than an "out-line" function. However, macros suffer from side-effects: unwanted, or unexpected modifications to variables. Macros should be used cautiously. The main purpose of the following programs is to demonstrate the differences between a function and a macro. Other purposes include demonstrating the differences between different programming environments, and learning different ways of writing comments, formatted input and output, variable declaration and initialization, unary operation ++, macro definitions/calls, function definitions/calls, if-then-else and loop structures, etc.

Start by running each of the functions below and understand their functionality. The code will need to manually be copied/typed into your editor from this document. **You can use either GNU gcc under Unix or Visual Studio to implement the code in this question.**

```
int subf(int a, int b) {
    return a - b;
}

int cubef(int a) {
    return a * a * a;
}

int minf(int a, int b) {
    if (a <= b) {
        return a;
    }
    else {
        return b;
    }
}

int oddf(int a) {
    if (a % 2 == 1) {
        return 1;
    }
    else {
        return 0;
    }
}
```

- 1.1 Write four macros to re-implement the given four functions. Name them: subm, cubem, minm, and oddm, respectively. [12 points]
- 1.2 Make a C file hw02q1.c that has the four functions and four macros defined in previous question. Write a main() function to test these functions and macros. Use the following test cases in the main() to call your functions and macros in this order and observe the results: [8 points]

```

a = 3, b = 6;
subf(a, b);
subm(a, b);
subf(a++, b--);
a = 3; b = 6;          // reset a,b values
subm(a++, b--);

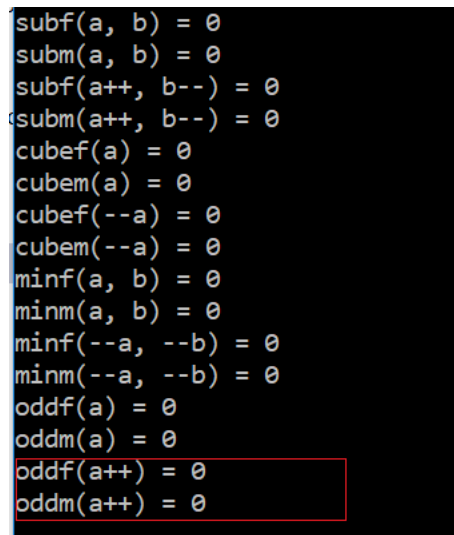
a = 3; b = 6;
cubef(a);
cubem(a);
cubef(--a);
a = 3; b = 6;
cubem(--a);

a = 3; b = 6;
minf(a, b);
minm(a, b);
minf(--a, --b);
a = 3; b = 6;
minm(--a, --b);

a = 2; b = 6;
oddf(a);
oddm(a);
oddf(a++);
a = 2; b = 6;
evenm(a++);

```

You must insert print statements to print answer of every function call and macro above, so that the expected output looks like the following:



```

subf(a, b) = 0
subm(a, b) = 0
subf(a++, b--) = 0
subm(a++, b--) = 0
cubef(a) = 0
cubem(a) = 0
cubef(--a) = 0
cubem(--a) = 0
minf(a, b) = 0
minm(a, b) = 0
minf(--a, --b) = 0
minm(--a, --b) = 0
oddf(a) = 0
oddm(a) = 0
oddf(a++) = 0
oddm(a++) = 0

```

Your output should have actual answers, not zeros! Take a screenshot of the output. Using an image editor (such as Paint) mark the lines in color where the function-macro pair gives different results, like `oddf(a++)` and `oddm(a++)` in figure above. Submit in a PDF file called `hw02q1.pdf`.

**For question parts 1.1 and 1.2, submit your program as hw02q1.c file and the screenshot file as hw02q1.pdf.**

2. You are given a file named hw02q2.c in the Canvas link for this assignment. All instructions are given in the form of comments in the file. **You are to run the file in both GCC and Visual Studio.** Observe the outputs and make changes as asked. **Please read all instructions very carefully, then complete and submit the updated file as hw02q2.c.** [30 points]

## What to Submit?

This homework assignment has multiple files. You are required to submit your answers in a compressed format (.zip). Make sure your compressed file is labeled correctly: lastname\_firstname2.zip. (All lowercase, do not put anything else in the name like "hw2".)

The compressed file **MUST** contain the following:

hw02q1.c  
hw02q1.pdf  
hw02q2.c

No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded.

**Submission preparation notice:** The assignment consists of multiple files. You must copy these files into a single folder for Canvas submission. We suggest that you double check your submission to Canvas: if you submitted an empty folder, an incomplete folder, or a wrong folder, you cannot resubmit after the submission linked is closed! We grade only what you submitted in the canvas. We cannot grade the assignment on your computer or any other storage, even if the modification date from your computer indicated that the files were created before the submission due dates. The contents of Canvas are what counts.