

# CSE 240 Homework 2, Spring 2020 (50 points)

Due Saturday, February 2, 2020 at 11:59PM, plus a 24-Hour grace period

---

## 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.

**Reading:** 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 expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

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

## Pre-requisite

See Homework 1. Install Visual Studio on your computer or use computers in BYENG214 lab.

## Programming Exercise (50 points)

1. Review the lecture slides which discuss Very Simple Programming Languages (VSPL). Next, observe the VSPL defined below and identify which sequences are valid.

```
<letter>      ::= a | b | c | d | e
<LETTER>      ::= V | W | X | Y | Z
<number>      ::= 0 | 1 | 2 | 3 | 4
<letters>     ::= <letter> | <letter> <letters>
<LETTERS>     ::= <LETTER> | <LETTER> <LETTERS>
<numbers>     ::= <number> | <number> <numbers>
<sequence>    ::= <letters> <LETTERS> <numbers> | <LETTERS> <letters> <numbers>
```

Which of the following are valid sequences? You must clearly identify for each of the following sequences, which are valid and which are invalid. Each sequence is worth 1 point. Submit your answer as hw02q1.pdf. [10 points]

1. CSE204
  2. ebcXYZ125
  3. XYZcde344
  4. VWXa10
  5. edc135790Z
  6. dYaZeWkV
  7. Zbad00
  8. aZ21
  9. XYZabc23
  10. Ey701
2. Read text section 1.4.2. 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 the side-effect, 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 definition/call, function definition/call, if-then-else and loop structures, etc.

Observe each of the functions below and understand their functionality. 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;
    }
}

```

- 2.1 Write four macros to re-implement the given four functions. Name them: subm, cubem, minm, and oddm, respectively. [10 points]
- 2.2 Make a C file hw02q2.c having 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: [5 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);

```

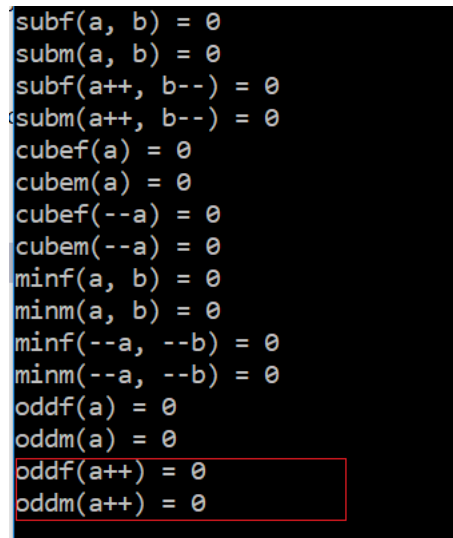
```

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

a = 2; b = 6;
oddf(a);
oddm(a);
oddf(a++);
a = 2; b = 6;
oddm(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. 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 `hw02q2.pdf`.

For questions 2.1 and 2.2, submit your program as `hw02q2.c` file and the screenshot file as `hw02q2.pdf`

3. You are given a file named `hw02q3.c`. 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 `hw02q3.c`. [25 points]

## Grading and Rubrics

Each sub-question (programming tasks) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each sub-question, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% - 10%	0
Each sub-question	Meeting all requirements, well commented, and working correctly in all test cases	Working correctly in all test cases. Comments not provided to explain what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed compilation, showing some effort, but the code does not implement the required work.	No attempt

## What to Submit?

This homework assignment will have multiple parts. You are required to submit your solutions in a compressed format (.zip). Make sure your compressed file is label correctly - lastname\_firstname2.zip. (All lowercase, do not put anything else in the name like "hw2".)

The compressed file MUST contain the following:

hw02q1.pdf

hw02q2.c

hw02q2.pdf

hw02q3.c

No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded. (Even if the assignment is submitted late.)

Submission preparation notice: The assignment consists of multiple files. You must copy these files into a single folder for canvas submission. To make sure that you have all the files included in the zip file and they work after unzip operation, you must test them before submission. You must also download your own submission from the canvas. Unzip the file on a different machine, and test your assignment and see if you can open and test the files in a different location, because the TA will test your application on a different machine. If you submitted an empty project folder, an incomplete project 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 indicated that the files were created before the submission due dates. The canvas submission may take a few minutes. Be patient and wait for it to complete.

## Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

## Late submission deduction policy

- No penalty for late submissions that are received within 24 hours after the deadline (before Sunday midnight);
- 10% grade deduction for every day it is late after the grace period (After Sunday);
- No late submission after Tuesday at 11:59PM.

## Academic Integrity and Honor Code.

You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in public websites before the assignment is due will be considered cheating.