CSE 240 Homework 1 (50 points)

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

- Learned a brief history of programming languages and the characteristics of the languages.
- 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:

- Students will understand the control structures of functional, logic, and imperative programming languages.
- Students will 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 or code** in the course discussion board.)

Pre-requisite

1. Subscribe to the General UNIX Cluster Server. You need to visit the ASU Computer Accounts Self-Sub website:

   http://www.asu.edu/selfsub

   You can login using your ASUAD User ID & password. It will list your current subscriptions and what other options are available for you to subscribe. Add **General** Computing Server if you don't have it.

   Your new General UNIX account may be ready immediately or may take a few days to become available. You can then log onto the "general.asu.edu" server using your ASUAD User ID and your password.

   You need a secure telnet client like SSH to access the general server. You can download the personal version of a windows SSH client from My Apps in My ASU.

   You will use ASU general account and SSH for C/C++ and Prolog programming in this course. You could create the general account to create your personal web page in the folder called www.

   In this course, you will use GNU gcc or g++ compilers to run C and C++ programs. You will also use Microsoft Visual Studio IDE (Integrated Development Environment) to do C/C++ assignments. Later in the semester, you will be required to use GNU Prolog under the general server to run all your Prolog programs. The course is designed to give you experience of using different kinds of programming environments!

2. Getting Started with GNU GCC on Unix

   To do the C and C++ assignments using GNU GCC, you need to have basic Unix knowledge. If you are not familiar with basic Unix commands, you need to read text appendix B, sections B.1 and B.2.1.

   Enter the "general.asu.edu" server by using the secure shell client SSH to connect to *general.asu.edu*. You can download a personal version of the client from My Apps in My ASU.

   Make a new directory (e.g. MyDir) by using the command: *mkdir MyDir*, use this to store all of your C and C++ programs in a specific directory. You may create subdirectories in this directory.

   Enter the directory by using Change Directory command: *cd MyDir*

   Please note that you can remotely connect (SSH) to the general server from anywhere, at any time, that is, you can do the assignment at home.

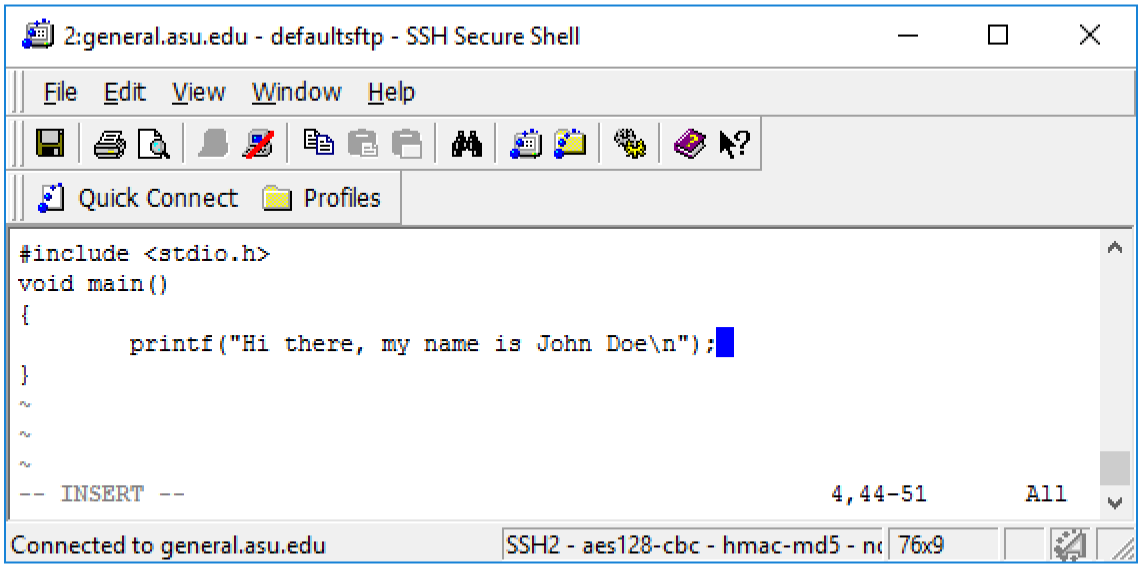
   To write a GNU GCC program on a Unix server, you can either use a Unix editor, e.g., *vi* to upload (using the same SSH software that includes FTP) a pre-edited file into your Unix directory *MyDir*. The name of a C program should have an extension *.c* and the name of a C++ program should have an extension *.cpp*

Programming Exercise (50 points)

1. Follow the Textbook Appendix B or Unix tutorial PPT given in the homework folder to complete the following tasks: [5 points]

1.1 Create a new directory called CSE240. Use cd CSE240 to enter the new directory. Use pwd command to find the path from the Unix root directory to your current directory.

1.2 Use vi to create a new program called hello.c and enter the following code into the program. You must use your name to replace the name "John Doe".


2:general.asu.edu - defaultsftp - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to general.asu.edu
SSH2 - aes128-cbc - hmac-md5 - n
76x9

```
#include <stdio.h>
void main()
{
        printf("Hi there, my name is John Doe\n");
}
~
~
~
-- INSERT --                                        4,44-51       All
```

Use gcc hello.c -o hello to compile the program. Fix any compilation errors if any.

1.3 Use ls, ls -l, and ls -al, respectively, to list files in the directory. Use command ./hello to executed the compiled code. What is the output?

Submit the screenshots containing the answers to questions 1.1, 1.2, and 1.3.

In the following two questions, you will use two different programming environments (IDEs) to edit, debug, and execute a simple C program. The basic part of C program is similar to Java. The purpose of the homework is to learn the programming environments. Read textbook Section 2.2.3, textbook section B.2, and the tutorial in section I of this document.

2 Use ASU General Unix Server and the text editor vi to enter the program on textbook pages 47-48 (the code is given on the next page). Use gcc and its debugger to debug the program.

2.1 Follow the given tutorial to compile, debug (find and fix any syntax errors), and execute the program and fix any semantic errors, for example, by adding "break" statements in the required places and by fixing incorrect characters copied into the programming environment, if any, and by type changing to print a floating point number. The code is supposed to perform one of the operations in each switch case. Submit the corrected program. [5 points]

```
/* This C program demonstrates the switch statement without using breaks. */
/* The program is tested on MS Visual C++ platform                         */
#include <stdio.h>
#pragma warning(disable : 4996) // Remove this line in Unix GCC environment

void main() {
        char ch;
        ch = '+';
        int f, a = 10, b = 20;
        printf("ch = %c\n", ch);
        switch (ch) {
        case '+': f = a + b; printf("f = %d\n", f);
        case '-': f = a - b; printf("f = %d\n", f);
        case '*': f = a * b; printf("f = %d\n", f);
        case '/': f = a / b; printf("f = %d\n", f);
        default: printf("invalid operator\n");
        }
        ch = '-';
        printf("ch = %c\n", ch);
        switch (ch) {
        case '+': f = a + b; printf("f = %d\n", f);
        case '-': f = a - b; printf("f = %d\n", f);
        case '*': f = a * b; printf("f = %d\n", f);
        case '/': f = a / b; printf("f = %d\n", f);
        default: printf("invalid operator\n");
        }
        ch = '*';
        printf("ch = %c\n", ch);
        switch (ch) {
        case '+': f = a + b; printf("f = %d\n", f);
        case '-': f = a - b; printf("f = %d\n", f);
        case '*': f = a * b; printf("f = %d\n", f);
        case '/': f = a / b; printf("f = %d\n", f);
        default: printf("invalid operator\n");
        }
        ch = '/';
        printf("ch = %c\n", ch);
        switch (ch) {
        case '+': f = a + b; printf("f = %d\n", f);
        case '-': f = a - b; printf("f = %d\n", f);
        case '*': f = a * b; printf("f = %d\n", f);
        case '/': f = a / b; printf("f = %d\n", f);
        default: printf("invalid operator\n");
        }
        ch = '%';
        printf("ch = %c\n", ch);
        switch (ch) {
        case '+': f = a + b; printf("f = %d\n", f);
        case '-': f = a - b; printf("f = %d\n", f);
        case '*': f = a * b; printf("f = %d\n", f);
        case '/': f = a / b; printf("f = %d\n", f);
        default: printf("invalid operator\n");
        }
}
```

You may create the file on your desktop computer and transfer the file into Unix using PuTTY or SSH.

2.2 Instead of repeating the same code five times, you can use a loop to reduce the code length. Modify the program. Use a for-loop to take 5 user inputs. Use an input statement such as scanf("%c", &ch); to replace the assignment statement ch = '+'; You can use either GCC or Visual Studio to complete this question. Please refer to the expected output word document for further clarification. Submit the rewritten program. [5 points]

3 Use Microsoft Visual Studio as the text editor and programming environment to execute the same program given above.

3.1 Compile, debug (find and fix any syntax errors), and execute the program and fix the semantic errors. Discuss and explain differences in compiling, editing, debugging, and executing that you observed in using GCC and Visual Studio. You **must** discuss and explain differences in each of the 4 tasks underlined above to receive all 5 points. [5 points]

3.2 Instead of repeating the same code five times, you can use a loop to reduce the code length. Modify the program. Use a **while**-loop to take 5 user inputs. Use an input statement such as ch = fgetchar(); to replace the assignment statement ch = '+'; You can use either GCC or Visual Studio to complete this question. Please refer to the expected output word document for further clarification. Submit the rewritten program. [5 points]

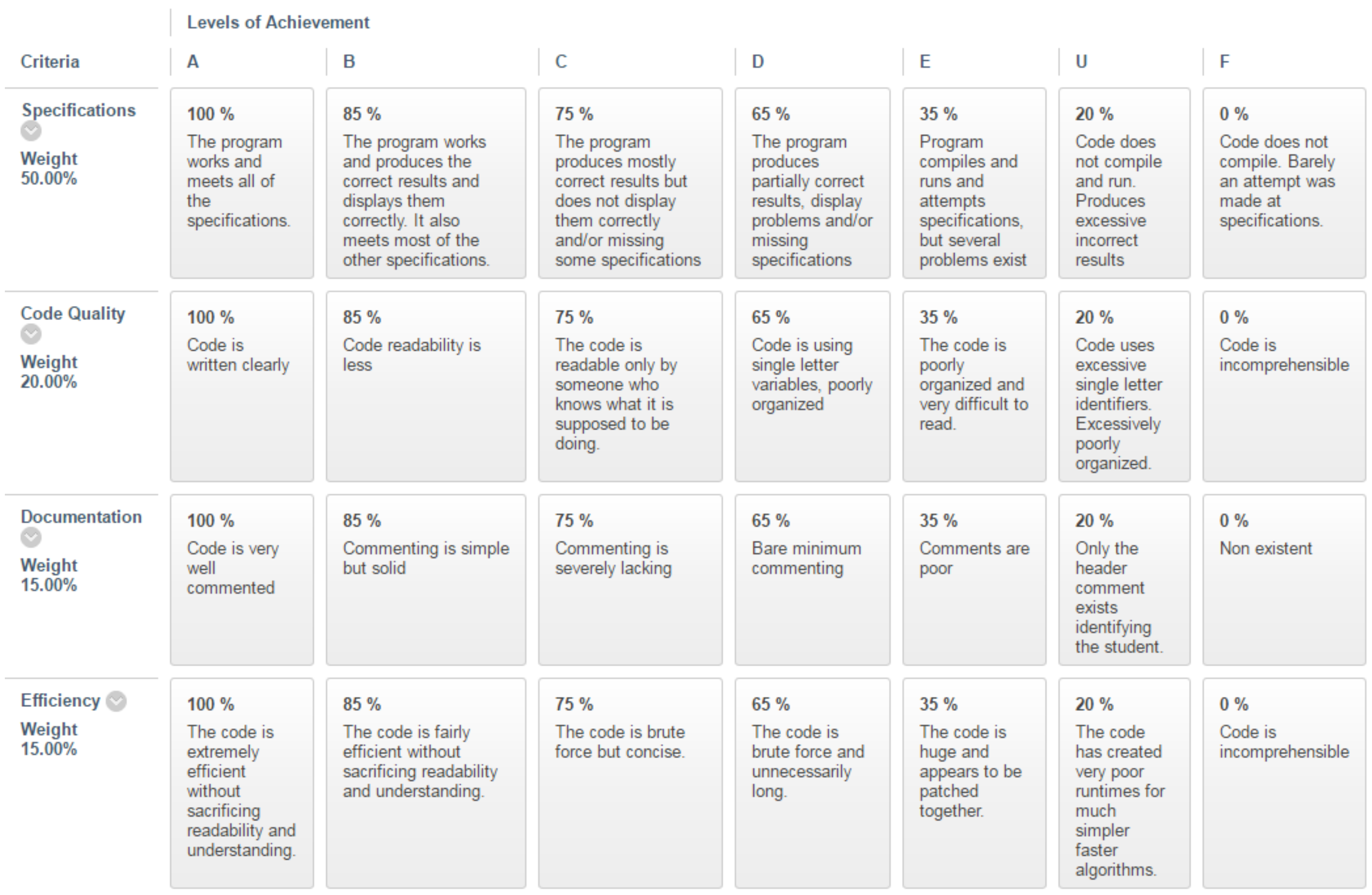
Note, you may need to use fflush(stdin) to flush the newline character left behind by a previous operation before using fflush(stdin).

4. You are given a file named "hw01q4.c". All instructions are given in the form of comments in the file. You are to correct the errors and identify which error type they are. Please read all instructions very carefully, then complete and submit the updated file. [25 points]

Grading of Programming Assignment

The TA will grade your program following these steps:

(1) Compile the code. If it does not compile you will receive a U on the Specifications in the Rubric
(2) The TA will read your program and give points based on the points allocated to each component, 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.



	Levels of Achievement						
Criteria	A	B	C	D	E	U	F
Specifications Weight 50.00%	100 % The program works and meets all of the specifications.	85 % The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	75 % The program produces mostly correct results but does not display them correctly and/or missing some specifications	65 % The program produces partially correct results, display problems and/or missing specifications	35 % Program compiles and runs and attempts specifications, but several problems exist	20 % Code does not compile and run. Produces excessive incorrect results	0 % Code does not compile. Barely an attempt was made at specifications.
Code Quality Weight 20.00%	100 % Code is written clearly	85 % Code readability is less	75 % The code is readable only by someone who knows what it is supposed to be doing.	65 % Code is using single letter variables, poorly organized	35 % The code is poorly organized and very difficult to read.	20 % Code uses excessive single letter identifiers. Excessively poorly organized.	0 % Code is incomprehensible
Documentation Weight 15.00%	100 % Code is very well commented	85 % Commenting is simple but solid	75 % Commenting is severely lacking	65 % Bare minimum commenting	35 % Comments are poor	20 % Only the header comment exists identifying the student.	0 % Non existent
Efficiency Weight 15.00%	100 % The code is extremely efficient without sacrificing readability and understanding.	85 % The code is fairly efficient without sacrificing readability and understanding.	75 % The code is brute force but concise.	65 % The code is brute force and unnecessarily long.	35 % The code is huge and appears to be patched together.	20 % The code has created very poor runtimes for much simpler faster algorithms.	0 % Code is incomprehensible

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_firstname1.zip. (All lowercase, do not put anything else in the name like "hw1".)

The compressed file MUST contain the following:

Hw01q1.pdf	(screenshots from questions 1.1, 1.2, and 1.3)
hw01q2_1.c	(program)
hw01q2_2.c	(program)
hw01q3.1.txt	(explain the differences)
hw01q3.2.c	(program)
hw01q4.c	(program)

No other files should be in the compressed folder. A .txt file is a plain text file, which is produced by tools like vi or Notepad. Microsoft Word does not save plain text by default.

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

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;