CSE 240 Homework 1, Spring 2021 (50 points)

Due Saturday, January 23, 2021 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 Unix operating system, 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 Unix and GNU GCC the programming environment.

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 assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their Zoom 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

You are required to do these exercises, in order for you to do the following assignments that require submission.

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 ASU **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 PuTTY or SSH to access the general server. You will use ASU General account and PuTTY or SSH for C/C++ and Prolog programming in this course. You can also create your personal web page in the folder called www in ASU General server.

In this course, you will use GNU gcc compiler to run C programs. 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 assignments using GNU GCC, you need to have basic Unix knowledge. If you are not familiar with basic Unix commands, you need to read textbook Appendix B, sections B.1 and B.2.1 and the Unix Tutorial given in the homework folder.

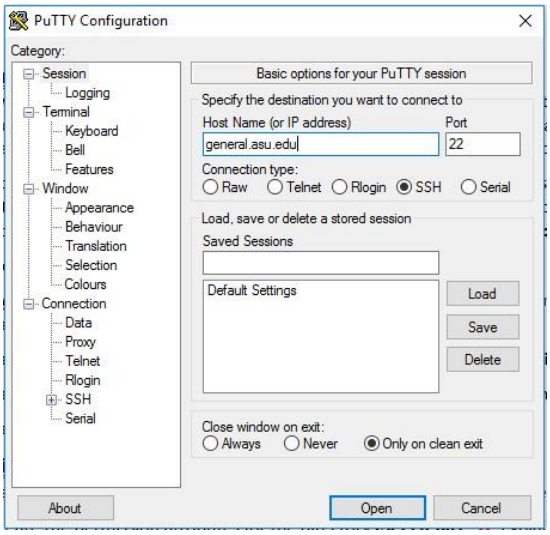
In order to connect to ASU general server, you need to install and use ASU VPN:

Enter MyASU --> My Apps --> Search VPN

You will find: Cisco Anyconnect SSLVPN. Install and enter the connection address: sslvpn.asu.edu. Then, you can use your ASUAD account to connect.

Once you have connected your sslvpn, you can connect to ASU General. Enter the "general.asu.edu" server by using PuTTY or SHH to connect to *general.asu.edu*.

1) *If you already have an account on* General*, then skip ahead to the next exercise*. Otherwise, in your browser, navigate to http://www.asu.edu/selfsub and subscribe to **general.asu.edu** to obtain an account on that system. It takes anywhere from a few minutes to perhaps an hour for your account to be activated. Please read Unix Tutorial for more detail.
2) Once your account is activated, run a **secure shell terminal** program such as **PuTTY** or **SSH** on your PC or Mac. Configure the terminal program to connect to **general.asu.edu** (in the **host name** text field) on **port 22**. Note: if you are running on Linux or a Mac, then you can start an SSH connection to **General** by opening a terminal window and typing **ssh *yourasuruteid*@general.asu.edu** at the prompt.



3) Log in to **General** using your ASUAD credentials.
4) Do a **pwd** command to determine your home directory. What is the **absolute** path name of your home directory?
   For example, when I log in to **General** and perform a **pwd** command, I see,
   /afs/asu.edu/users/y/c/h/ychen1
   Of course, it should be the path from the root to your directory.
5) Do an **ls** command to see the files in your home directory.
6) Do an **ls -a** command to see **all** of the files in your home directory, including hidden files.
7) Do an **ls -l** command to produce a **long** listing of the non-hidden files in your home directory.
8) Do an **ls -al** command to produce a **long** listing of **all** of the files in your home directory.

Make a new directory (e.g. MyDir or CSE240) 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 (using PuTTY or 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., *vim* or *pico* or upload (using the same PuTTY software that includes SSH/telnet) 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 and the Unix Tutorial PPT given in the homework folder to complete the following tasks:

   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 Create three new subdirectories inside CSE240. Name the directories d1, d2, and d3 respectively.

   1.3 Enter the directory d1 and use vi or vim 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". Save the file.

```
1
2  #include <stdio.h>
3  void main()
4  {
5          printf("Hello, my name is John Doe");
6          // replace John Doe with your name
7  }
8
```

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

   1.4 Use ls, ls -l, and ls -al respectively to list files in d1 directory. Use command ./hello to execute the compiled code. . [4 points]

   Take screenshots of each command in this question. You may take one screenshot with all commands in it.

   1.5 Enter d2 directory and then use one command to copy (not move) the files hello.c and hello from d1 directory into d2 directory. Use ls -al to view all the files in d2.

   Take screenshots of each command in this question. You may take one screenshot with all commands in it. [3 points]

   1.6 In d2 directory, use chmod 660 hello to change the permission. Use ls -l to view the files. Use command ./hello to execute the compiled code. Take a screenshot of this output. Use chmod command again to make hello an executable, but not readable and not writeable for all users. Use command ./hello to execute the compiled code. Take a screenshot of this output. [4 points]

1.7 Enter CSE240 directory. Use ls -l to view all the files. Then, use one command to delete d2 directory and all the files in d2 directory. Use ls -l to view all the files.

Take screenshots of each command in this question. You may take one screenshot with all commands in it. [3 points]

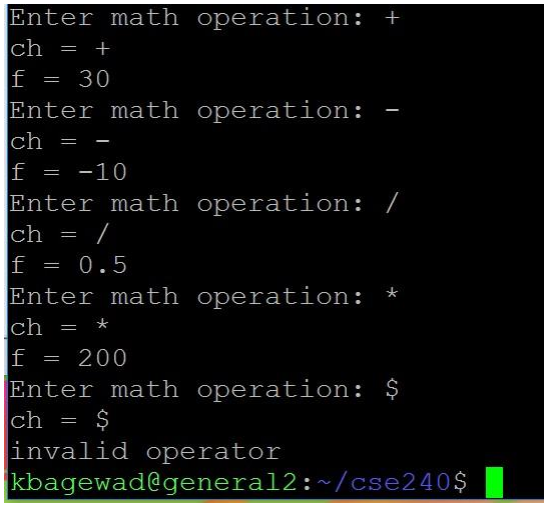
Screenshots needed for questions 1.4 through 1.7. Put all the screenshots along with their question numbers in a word file and convert to pdf. Submit the file as hw01q1.pdf

2. In this question, you will use Unix tools to edit, debug, and execute a simple C program. The purpose of the homework is to learn the Unix programming environment. Read textbook Section 2.2.3, the tutorials, and the Unix tutorial in text Appendix B.2.

2.1 Use a text editor to enter the following program. Variations and more discussions of this program can be found in textbook Section 2.1.3. Use GNU GCC 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 math operation in each switch-case. The 'ch' variable is assigned a new math operator before each switch-case. Submit the corrected program as hw01q2.c [8 points]

```c
/* This C program demonstrates the switch statement without using breaks. */
#include <stdio.h>
 main() {
        char ch = '+';
        int a = 10, b = 20;
       double f;
        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"); }
}
```

2.2 Edit the code above or write a new code to use a for-loop (that runs 5 times) to ask the user for a math operation as input (+, - , * , /). Use an input statement such as *ch = getchar();* or *scanf("%c\n", &ch);* to replace the assignment statement *ch = '+'* in the code above. Expected result shown in the figure below. Submit the revised program as hw01q2_2.c. [8 points]



Note, when you add the loop, you may need to use a *fflush(stdin)* or a *getchar()* to flush the newline '\n' character left behind by a previous operation; You can use one ch = getchar() at the beginning of the loop and one before the end of the loop:

```
for ( … )
{
  printf("please enter a char");
  ch = getchar(); // read a char
  other code;
  ch = getchar(); // This line will flush '\n' character left behind by the previous getchar().
  ch = getchar(); // You may need this one to keep the console window open
}
```

Please read text Section 2.1.3 for more details.

3. You are given a file named "hw01q3.c". All instructions are given in the form of comments in the file. You should correct the errors and identify which error type they are. Please read all instructions carefully, then complete and submit the updated file. [20 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 answers in a compressed format (.zip). The compressed (zip) file MUST contain the followings:

hw01q1.pdf	(screenshots from questions 1.4 through 1.7)
hw01q2_1.c	(program)
hw01q2_2.c	(program)
hw01q3answer.c	(program)

No other files should be in the compressed folder. Do not submit the project solution or other project files.
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.