ENGG1003 - PASS Session 1

Mitchell Deltoer

Welcome to PASS (Peer Assisted Study Sessions) for ENGG1003. PASS is essentially a regularly scheduled group study session led by a student who has previously achieved excellent results in the course. Sessions are free, voluntary, and do not require you to enrol beforehand. For ENGG1003, there are 3 repeat sessions per week; you are invited to show up to as many or as little as you like. The times are listed below:

Monday	**14-15**	**ES238**
Wednesday	**12-13**	**ES238**
Thursday	**10-11**	**MCLG42**

Each session, I will be bringing my own worksheets that cover topics from the previous week's lectures. You are also invited to bring your own questions and assignments to the sessions. It is **highly recommended that you bring your own laptops to the sessions** to enhance your own learning experience.

Leaders are not here to act like your tutors, we are here to facilitate the group discussion and provide help when necessary. Group work and asking other students is highly encouraged before directly asking me questions. I will of course do my best to give you different explanations and new perspectives on programming concepts, and I will happily give any tips/insights on anything uni related.

I will be uploading the weekly worksheets on the *Online PASS* blackboard site. If you haven't joined yet, just ask me or another student how to sign up and access them.

I hope to continue seeing you throughout the semester. Don't be afraid to ask for help and most importantly, have fun programming!

Fundamentals of C

1. What is the absolute bare minimum structure of a C program?

2. *Header files* are files that contain *function* and *macro definitions* so they can be used within your program. How and where are *header files* included in a C program?

3. Which *header file* is required for the `printf` and `scanf` *functions* to work?

4. Data, i.e. numbers, are stored in something called a *variable*. Before a C program can use a *variable*, it must be *declared* so that it is allocated its own memory location and unique name.
   Use the following descriptions to declare your own variables:

   (a) An integer called `i`

   (b) A float called `mass` *initialised* to the value 6.21

   There are many other different types of variables and it is up to the programmer (spoiler:that's you), to decide which is most appropriate for the situation.

5. The `scanf` and `printf` *functions* are the most simple ways of interfacing with a basic C program through *standard input* through the keyboard and *standard output* through the console respectively.
   Using the `scanf` and/or the `printf` *functions*, complete the following tasks:

   (a) Read an integer and store it in `i`.

   (b) Read a float and store it in `mass`.

   (c) Display the text `"The current mass is:  "` followed by the value of `mass`.

(d) Display the text `"Please enter the current mass:  "` followed by reading in a float and storing it in `mass`.

Arithmetic in C

6. Match the following mathematical operations with their C counterparts.

Operation	**C Symbol**
Addition	`-`
Subtraction	`--`
Multiplication	`+`
Division	`++`
Modulus	`/`
Increment	`%`
Decrement	`*`

7. Write the following mathematical expressions as C statements.
   Notes:
   - a, b and c are separate variables
   - Be sure to use enough parenthesis (brackets)

   (a) $\frac{ab}{23} + 12$

   (b) $\frac{10 + ab}{abc} + 3b$

   (c) $9a - \left(\frac{32}{a-3}b\right) - 4c$

   (d) $\left(\frac{a^2 + b^2}{46a}\right) c$

8. order of operations questions

Practice Programming

Some things to consider for the following questions:

- How can I take an *input* from the user? How can I display information to the user?
- How many *variables* will I need? What *data type* should each of them be?
- How could I use *intrinsic documentation* (Lab 1) to make my code easier to read?
- Where should I be using *comments* to further explain my code?
- What mathematical formulas (if any) will I need?

9. Write code for a C program that will calculate the perimeter and area of a rectangle, with two side lengths provided by user, and print the results to the console.

10. Write code for a C program that prompts the user to enter 3 different integers as input, then print those numbers back to the user in reverse order.