

# **COMP1511/1911 Programming Fundamentals**

**Week 1 Lecture 1**

**The Beginning**

# Today's Lecture

- Welcomes and Introductions
- How COMP1511/COMP1911 works
- How to get help
- What is programming?
- Working in Linux
- A first look into C



# Who am I?



**Dr Angela Finlayson**

Lecturer in Charge

Oscar's mum

Tina Arena Fan

Civ 6 Enthusiast

# Our Wonderful Course Admins!!!!



**Sofia De Bellis**



**Grace Murray**

# Amazing Lecture Streamers/Moderators



**Liam  
Phillips**



**Rosemary  
Ai**



**Conrad  
Vernon**

And also **Brianna Kim** as our special guest moderator for the first 2 weeks



# Fabulous Forum Team



**Amber  
Lucas**

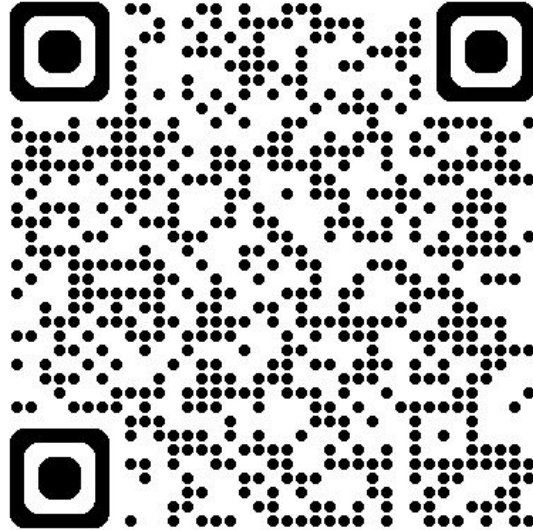


**Max  
Davidson**



**Timothy  
Zhou**

# And our Incredible Team of Tutors!



[Meet the COMP1511/1911 25T3 Team!](#)

# In COMP1511/COMP1911

**We teach** you the fundamentals of programming

- how to approach and solve problems
- how to express our solutions in step by step instructions that a computer can execute

**We assume**

- no prior knowledge
- no previous programming experience

**Problem solving is a skill that can only be built with practice!**



# Code of Conduct

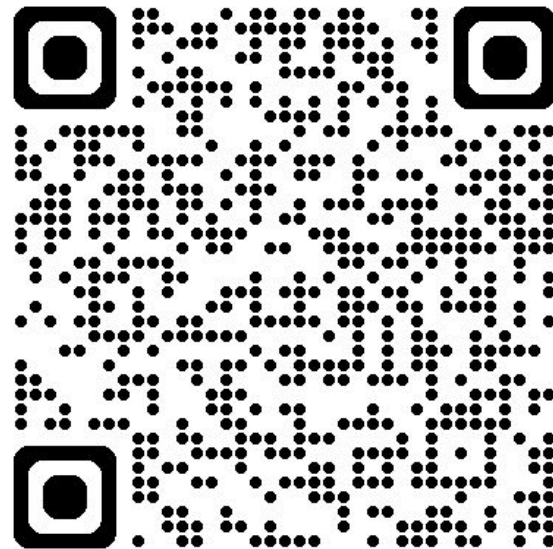
- UNSW is committed to providing an inclusive learning environment
- All activity connected to COMP1511 and COMP1911, including social media, must be respectful.
  - No discrimination of any kind
  - No inappropriate behaviour
  - No harassment, bullying, aggression or sexual harassment
  - Full respect for the privacy of others

# Getting to Know You Game!

# Course Website

<https://cgi.cse.unsw.edu.au/~cs1511/current>

- All course information is on our course website
  - Please bookmark it
- **We don't use Moodle**
- Please read the course outline thoroughly



# Course Format

- Weekly Lectures 2 x 2 hours
- Weekly tut/labs 3 hour blocks
- 2 Major Assignments
- 1 Final Exam

# Lectures 2x2 Hours a Week

- All lectures:
  - Live streamed online via YouTube and recorded
- Times and Locations:
  - **Monday 11am-1pm** also in person in **Ainsworth G03**
  - **Tuesday 4pm-6pm** (online only)

Week 6 is Flex Week, so no formal lectures!

# Lectures 2x2 Hours a Week

- Lecture slides and code are available on course website
- Recordings will be available on course website and YouTube playlist
- If you have a question during the lecture:
  - Put your hand up and ask
  - Ask in live chat
- Please be respectful of others - everyone is here to learn
  - Don't be noisy
  - Be kind to one another in the chat and of course in person too :)



# Tutorial: 1 hour

- Classroom environment - interactive learning
- Tutorial Questions will be available in advance
- Online and face-to-face:
  - please check your timetable for your enrolment details
- For online classes:
  - use Teams
  - Please turn on your cameras if you can
  - We love seeing pets make an appearance

# Lab: 2 hour (directly after tutorial)

- Hands on Practical coding
  - Mostly individual, sometimes working in pairs
- Time to have one-on-one conversations with your tutors
- Problem set released each Sunday night 6pm
- Problem sets submitted by Monday 6pm the next week
  - marked automatically
  - Worth 15% total over the term (no marks for week 1)
    - Worth 2 marks each (capped at 15 overall)

# Lab: 2 hour (directly after tutorial)

- Problems have difficulty levels, 1 dot, 2 dot and 3 dot
  - 3 dot problems are optional, bonus marks, some are very hard and time consuming
- Week 8 will include practice array hurdle exam questions!
- Tutorials and Labs do NOT run in Week 6 (Flex Week)

# Assignments

Assignment 1 - 20% (Released Week 4 Due Week 7)

Assignment 2 - 25% (Released Week 8 Due Week 10)

- Individual work
- Take you a few weeks
- Apply the theory and practice your skills on larger problems

# Assignment Tips

- Start as early as possible
  - Time to get help as needed
  - Time to take breaks
  - Too stressful to do at the last minute
- Don't plagiarise or use ChatGpt to generate your code
  - We have sophisticated plagiarism detection software.
- Get help from appropriate sources - help sessions, forum, tutors in your lab

# Late Penalties

- Assignments and Labs can be submitted late with a penalty
- Late penalties of 5% per day late apply off the ceiling
  - maximum lateness is five days, after which time it is zero marks



# Plagiarism

- Plagiarism: The presentation of someone else's work or ideas as if they were your own.
- Any kind of cheating on your work for this course will incur penalties (see the course outline for more details)
- At best, you'll lose the marks for the assignment
- At worst, you'll be asked to leave UNSW
- You will also cheat yourself out of learning!!!
  - Doing labs and assignments on your own is the best way to study for the final exam

# Plagiarism vs Collaboration

- Collaboration on individual assessments like Assignments is considered plagiarism
  - Your submissions must be entirely your own work
  - Don't use other people's code
  - Don't ask someone else to solve problems for you (even verbally)
  - Don't provide your code to other people
- 2 lines of code from stack overflow is ok
  - Just make sure to reference your source in a comment
- Downloading complete code found on github is NOT ok.

# Use of Generative AI Tools

- You are not permitted to submit code generated by automatic tools such as ChatGPT, GitHub Copilot, Google Gemini (Bard) etc
- This is treated the same as plagiarism

# Generative AI

- Will you be able to detect the weird issues and bugs in AI generated C code ?
- Will AI generate the same code for other students?
- What will you do in the final exam without AI?



# Final Exam: 3 hours

- **In Person**

- 3 hours
- C programming questions

- **Exam Hurdles**

- Parts of the exam are competency hurdles
- Hurdles must be completed satisfactorily to pass the course
- Array Hurdle for COMP1511 and COMP1911
- Linked List Hurdle for COMP1511 but NOT COMP1911  
(COMP1911 students still do linked list questions for marks).

# Total Assessment

Labs = 15%

Assignment 1 = 20%

Assignment 2 = 25%

Final Exam = 40%

To pass the course you must:

- Score at least 50/100 overall in the course
- Pass the exam hurdle/s



# COMP1911 vs COMP1511 Assessment

- COMP1911 is for non-computing majors
  - is a prerequisite for COMP1521
  - but not a prerequisite for other COMP courses
- COMP1911 and COMP1511 have the same classes, assessments and exam except:
  - COMP1911 does not need to do some of the final stages of assn2
  - Linked List questions in exam are not a hurdle for COMP1911

# Special Consideration

- Support for any issues that impact study and assessments
  - Feeling unwell or an emergency situation
  - You will need documentation
  - Extension for assignment
  - Supplementary exam

<https://student.unsw.edu.au/special-consideration>

# Supplementary Exam

- May be offered to students granted Special Consideration
- Identical in format to the main exam
- Held in January/February 2026 ( exact date pending)
  - you must make yourself available if you are granted a supplementary exam
- Fit-to-Sit rule

# Equitable Learning Plans

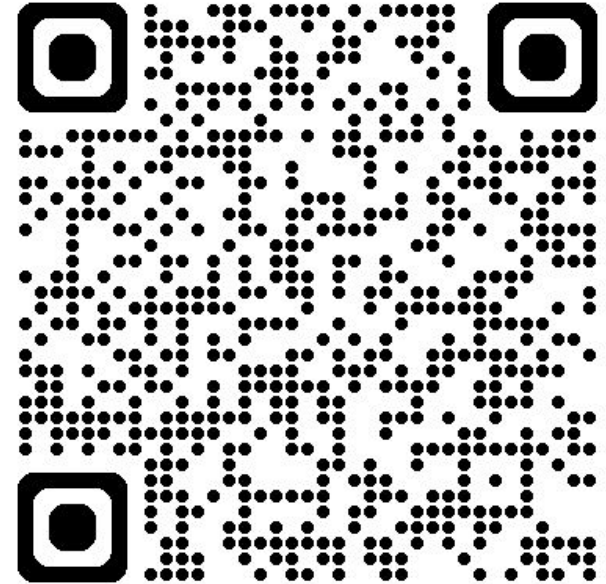
- Please make sure to have your ELP plans support you in the best possible way this
- Email your ELP plan pdf to me at:  
**angela.f@unsw.edu.au**
- For more info, please visit:



<https://www.student.unsw.edu.au/equitable-learning>

# Course Content Related Help

- Ask questions in lectures and in lecture chat
- Ask Questions in tuts and labs!
- **Forum:**
  - Post all your questions here
  - Feel free to answer other's questions
  - Don't post your code publicly in the forum



<https://discourse01.cse.unsw.edu.au/25T3/COMP1511/>

# Course Content Related Help

- **Help Sessions: Sessions start in week 3**
  - Good place to get one-on-one help outside of normal lab/tutorial times
  - There are optional drop in sessions
  - Schedule will be announced shortly
- **Revision Sessions: Sessions start in week 4**
  - Optional group sessions to revise relevant topics
  - Schedule announced shortly



# Admin Related Help

- **Course Administration Issues:**

- Email:

[cs1511@unsw.edu.au](mailto:cs1511@unsw.edu.au)

- **Enrollment Issues:**

- UNSW Nucleus Student Hub

<https://nucleus.unsw.edu.au/en/contact-us>

- **cse course account issues:**

- CSE Help Desk:

<http://www.cse.unsw.edu.au/~helpdesk/>

# Student Support - I Need Help With...

## My Feelings and Mental Health

Feeling depressed, overwhelmed  
or not your usual self?



**Mental Health Connect**  
[student.unsw.edu.au/mhc](http://student.unsw.edu.au/mhc)



**TalkCampus**  
[student.unsw.edu.au/Talkcampus](http://student.unsw.edu.au/Talkcampus)



**UNSW (24/7) Mental Health Support:** [+61\(2\) 9385 5418](tel:+61293855418)

**Text support After Hours:** [0485 826 595](tel:0485826595)



**Offshore**  
**Call 24-hour Medibank Hotline:** [+61 2 8905 0307](tel:+61289050307)

## Uni and Life in Australia

Stress, financial, visas,  
Accommodation & More



**Student Support**  
**Indigenous Student Support**

[student.unsw.edu.au/advisors](http://student.unsw.edu.au/advisors)  
[nura-gili-centre-indigenous-programs](http://nura-gili-centre-indigenous-programs)

## Reporting Sexual Assault/Harassment



**Equity Diversity & Inclusion (EDI)**

[edi.unsw.edu.au/sexual-misconduct](http://edi.unsw.edu.au/sexual-misconduct)

## Educational Adjustments

To manage my studies and  
disability / health condition



**Equitable Learning Services (ELS)**

[student.unsw.edu.au/els](http://student.unsw.edu.au/els)

## Academic and Study Skills



**Academic Skills**

[student.unsw.edu.au/skills](http://student.unsw.edu.au/skills)

## Special Consideration

Because life impacts our  
studies and exams



**Special Consideration**

[student.unsw.edu.au/special-consideration](http://student.unsw.edu.au/special-consideration)

## Physical Health

Doctors visits, Vaccinations



**Health Service**

[student.unsw.edu.au/hsu](http://student.unsw.edu.au/hsu)

# Quick Break. Back Soon...



**Computers, programs, operating systems, Linux,  
terminals, compilers, C programming language, oh  
my...**

# What is a Computer?

# What is a computer?

- A machine that can be reconfigured for different purposes
- The key elements:
  - A processor to execute commands
  - Memory to store information

# What is Programming?

# What is Programming?

- Providing a computer with specific instructions to solve a problem
  - Using specific languages to write those instructions (code)
- It involves problem solving skills!
  - You may need to start with pen and paper planning
  - Testing and Debugging skills are required
  - You may go through many iterations before you get it right



# Mistakes and Bugs are Normal

At times you may require

- patience
- perseverance

The key to success is

- practice



# What is a Programming Language?

# Computers vs Humans

- Computers execute precise instructions encoded in binary
- This language is not easy for humans to read or write

```
0100 0000 0000 0000 0000 0000 0000 0000
1011 0110 0000 0000 0000 0000 0000 0010
0000 0100 0110 0000 1001 0000 0000 0000
```

# Programming Languages

Humans speak in languages that are not precise enough for computers

Programming Languages take the middle ground:

- Precise enough to be translated to machine code
- Simple enough that a human can (hopefully) understand it

# How do these programs run?

- Computers often have hundreds of programs executing at the same time!
- Imagine if your kitchen was used to prepare hundreds of recipes all at once and no-one was in charge.



**We need a head chef  
(operating system) to manage  
resources (computer hardware)**

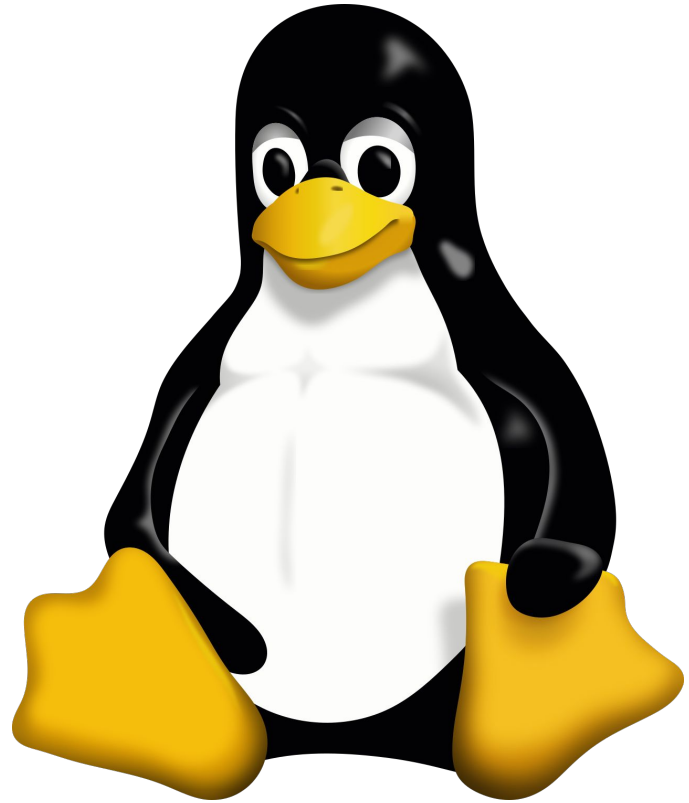
# Operating Systems

- An Operating System is the interface between the user and the computer hardware
  - Execute user programs
  - Make sure programs do what they're supposed to
  - Schedules access to limited resources (hardware)
  - Make the computer system convenient to use
- Operating systems are themselves programs!
  - Which one are you using?



# The Linux Operating System

- UNIX-based
- open-source
- reliable
- lightweight
- secure



# Why Linux?

- Consistency and fairness
  - everyone uses the same environment
- Industry relevance
  - widely used in servers, research, and development
- Transparency
  - open-source tools show how systems really work

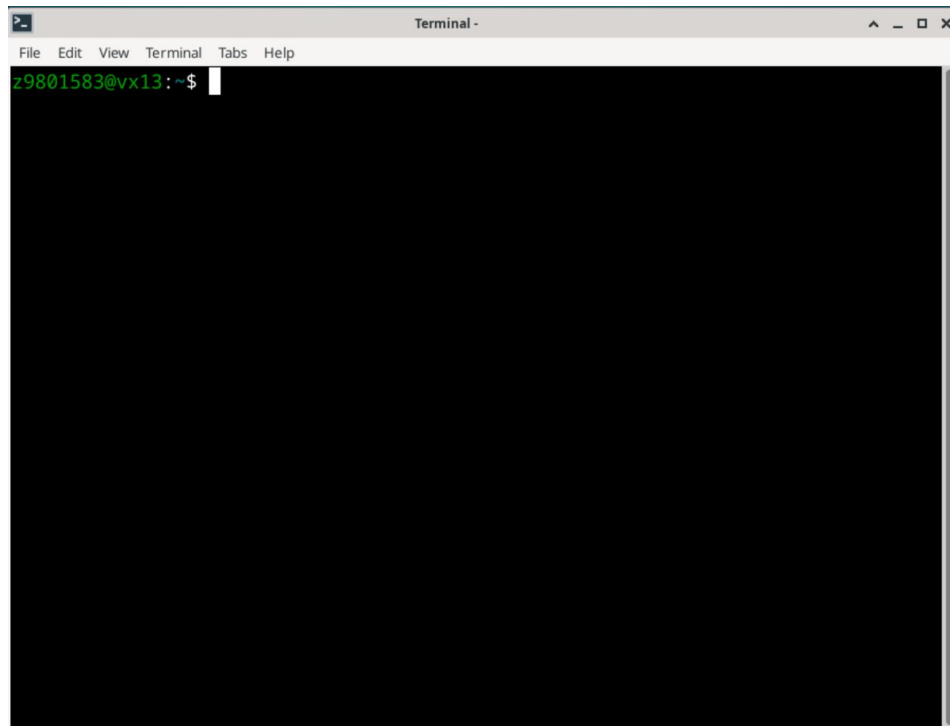


# How do REAL programmers interact with the Computer?



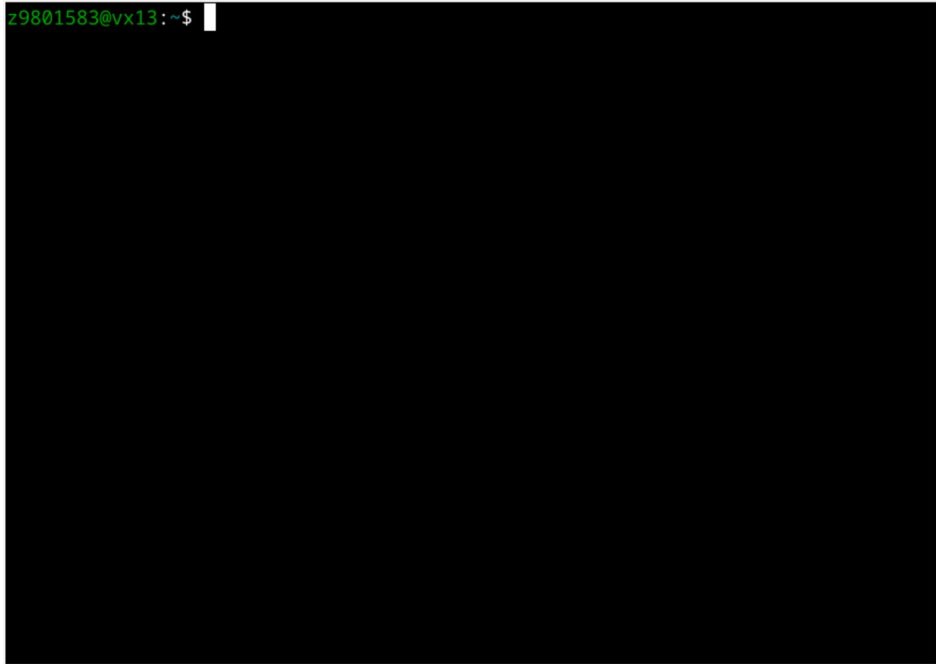
# The Terminal

- A graphical application
  - sends text commands to the shell
  - displays shell output



# The Shell

- The shell, (bash, zsh) is a program that runs in the terminal.
- It executes commands, and has its own syntax.
- It returns output which the terminal can display

A screenshot of a terminal window with a black background. The prompt 'z9801583@vx13:~\$' is displayed in green text at the top left, followed by a white cursor. The rest of the terminal area is empty.

```
z9801583@vx13:~$
```

# The Command Line Prompt

- The prompt is controlled by the shell
- It is the line of text which displays some information often ending in a \$
- Lets you know it is ready for you to type

A screenshot of a terminal window with a black background. The prompt 'z9801583@vx13: ~\$' is displayed in a bright green monospaced font. The text shows the username 'z9801583', the hostname 'vx13', the tilde '~' representing the home directory, and the dollar sign '\$' representing the shell prompt.

```
z9801583@vx13: ~$
```

**Help!!! I don't have a Linux machine!!!**

**Don't worry! We have one for you!!!**

# CSE's Computing Resources

- Our labs run Linux with everything you need to get started
- Running Linux on your own computer:
  - VLAB allows you to remotely use CSE's resources 24/7
  - instructions on setting this up available in the first laboratory

[https://cgi.cse.unsw.edu.au/~cs1511/25T3/resources/home\\_computing.html](https://cgi.cse.unsw.edu.au/~cs1511/25T3/resources/home_computing.html)



# Working in your CSE account

- Log into **VLAB** or CSE lab machine
  - with zid and zpass
- Open a terminal
- Type **1511 setup** in the terminal to get everything ready
  - you will do this in your first Lab.
  - you only need to do it once

# Important Linux Shell Commands

<code>ls</code>	List all the files in the current directory
<code>mkdir dir_name</code>	Make a new directory called dir_name
<code>cd dir_name</code>	Changes the current directory to dir_name
<code>cd ..</code>	Move up one directory (folder) level
<code>cd</code>	Move to your home directory
<code>pwd</code>	Tells you where you are in the directory structure

# Important Linux Shell Commands

<code>cp source destination</code>	Copy a file from the source to the destination
<code>mv source destination</code>	Move a file from source to destination directory OR Rename source to destination
<code>rm file_name</code>	Remove (delete) file_name Does not put in trash. Permanently deletes <b>Warning:</b> Use <code>rm</code> with caution

# Important Linux Shell Commands

The `-r` flag can be added to `cp` or `rm` commands to recursively go through a directory and perform the command on all the files

`cp -r source destination`

copy all files from source to destination directory

Find out more tips and tricks in the [Linux Cheatsheet](#)

# Programming in C

## Why C?

- Many modern programming languages are based on C
- Learning C helps in understanding the underlying architecture of computers
- Windows, Linux, MacOS operating system kernels written almost entirely in C
- We are learning and using C but really we want you to learn
  - Logical thinking and problem-solving skills
  - Skills that will transfer to other languages and areas of computing

# Finally... we get to C some C

```
// A program showing how to print output in C
// The first of many C programs you will C

#include <stdio.h>

int main(void) {
    printf("Hello COMP1511 and COMP1911\n");
    return 0;
}
```

# Let's Break it Down: Header Comment

- `//` in front of a line makes it a comment
- If we use `/*` and `*/` everything between them will be comments
- Comments are for humans
- The compiler ignores them
- They help future selves and colleagues understand our code
- Header comment is a special comment we put at the top of each program

```
// A program showing how to print output in C  
// The first of many C programs you will C
```

```
#include <stdio.h>
```

```
int main(void) {  
    printf("Hello COMP1511 and COMP1911\n");  
    return 0;  
}
```

# Let's Break it Down: #include

- `#include` lets the compiler know what part of the standard C library we want to use
- In this case, it's the Standard Input Output Library
  - we need it for `printf`
- Almost every C program you will write will have this line

```
// A program showing how to print output in C  
// The first of many C programs you will C
```

```
#include <stdio.h>
```

```
int main(void) {  
    printf("Hello COMP1511 and COMP1911\n");  
    return 0;  
}
```



# Let's Break it Down: main function

- Every C program needs exactly 1 main function
- That is where program execution starts
- Between the { and } are a set of program instructions
- Our computer will execute the instructions line by line
- A main that returns **0** signifies that the program executed successfully.

```
// A program showing how to print output in C  
// The first of many C programs you will C
```

```
#include <stdio.h>
```

```
int main(void) {  
    printf("Hello COMP1511 and COMP1911\n");  
    return 0;  
}
```

# Let's Break it Down: printf

- `printf()` writes text to standard output (screen).
- It is a function from `stdio.h` which we included
- Everything between the starting "`"` and ending "`"` gets printed
- `\n` is the newline character
- All executable statements end with a semicolon `;`

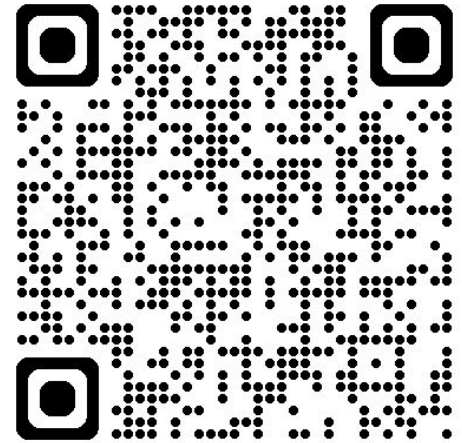
```
// A program showing how to print output in C
// The first of many C programs you will C

#include <stdio.h>

int main(void) {
    printf("Hello COMP1511 and COMP1911\n");
    return 0;
}
```

# Demo: Writing your first C program

- Follow along now or try it later once you have set up VLAB
- Open a file called `hello_world.c` in VSCode from the terminal by typing  
`code hello_world.c`
- Edit the file and save it
- You can find live lecture code at  
[https://cgi.cse.unsw.edu.au/~cs1511/25T3/code/week\\_1/](https://cgi.cse.unsw.edu.au/~cs1511/25T3/code/week_1/)



# But how can we actually run our program???

Reminder:

- Computers can't execute C code
- Compilers are programs that turn code into machine code

We use the **dcc compiler**:

- **dcc** was developed at CSE UNSW
  - Designed for beginners and gives helpful error messages

# Demo: Compiling and running code

- This compiles a C program `hello_world.c` into an executable file called `a.out`:

```
dcc hello_world.c
```

- We can then run our executable file:

```
./a.out
```

Note: `./` just means the current directory

- `a.out` is not the greatest name. Let's see a better way...

# Demo: Compiling and running code

What if we have lots of executable programs and we want to give them names?

This compiles a C program `hello_world.c` into an executable file called `hello_world`:

```
$ gcc hello_world.c -o hello_world
```

We can then run our executable:

```
$ ./hello_world
```

# Compile errors or warnings?

1. Try to fix the first error first
2. Save
3. Re-compile
4. Repeat till errors are gone

Then you can run your program to test that it does what you want!  
You may need to edit it, compile and repeat...

# Compile errors or warnings?

Error on line 42

```
41      });  
42  
43      if (includ
```





# dcc-help and dcc-sidekick

`dcc-help` and `dcc-sidekick` are tools to help beginners understand compiler error messages to help you fix your code.

Great to use while learning and doing labs and assignments

**Warning:** `dcc-help` and `dcc-sidekick` are not available in the final exam. Note that you WILL still have `dcc` in the final exam.

# Escaping

- \ is an escape character
  - \n is an escape sequence that means newline
  - \t is an escape sequence that means tab
  - \" is an escape sequence that allows us to print the " character
    - Why would we need this?

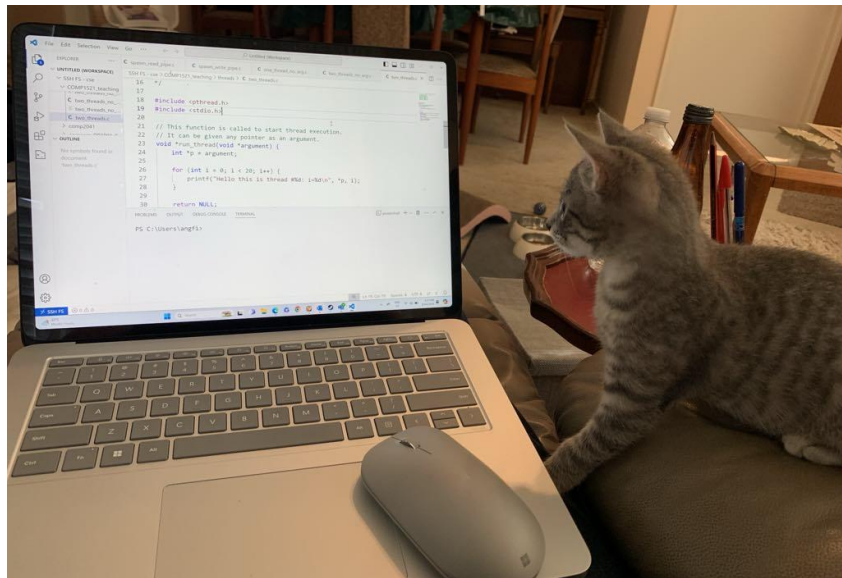
How can we actually print a \ character?

Why we use \\ of course!

```
printf("This is a Backslash \\");  
printf("This is a Backslash followed by a newline \\n");
```

# Learning Programming

- Requires
  - practice - labs and assignments are a great way to learn!
  - trying things and making many many many mistakes!
  - asking for help when needed
- It can be challenging but also very rewarding!
- We are here to guide you and support you.



# What did we learn today?

- **Admin:** How the course is run
- **Resources:** Where to find them
- **Help:** Where to find it
- **Concepts:** Computer, Program, Operating System
- **Linux:** Some basic commands
- **C:** Compiling and running your first C program
  - `hello_world.c`

# Reach Out

Content Related Questions:  
Forum

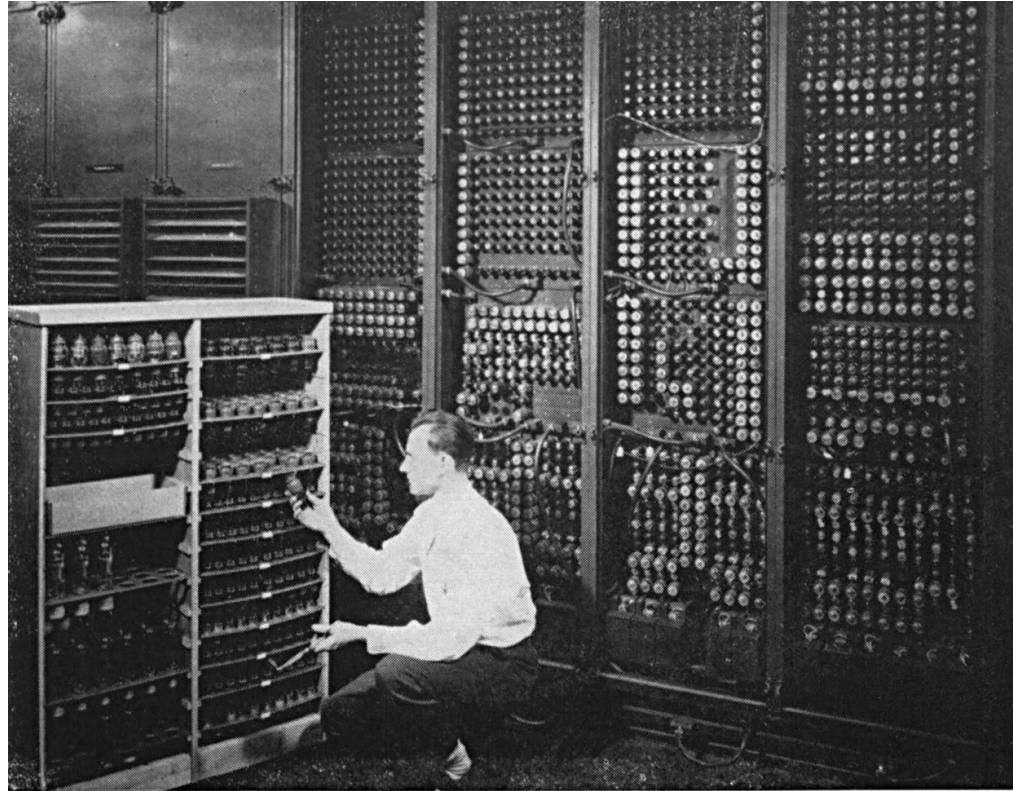
Admin related Questions email:  
[cs1511@unsw.edu.au](mailto:cs1511@unsw.edu.au)



# Computing Trivia: ENIAC

The first computer in the world, ENIAC, weighed more than 27 tonnes and covered an area of about 1800 square feet?

Designing the correct configuration for each new problem, and then connecting the wires and setting the switches, took many days



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.