CPSC 102

- syllabus, grade dist., acad. integrity, etc.

- course objective:
  - advanced C prog.
  - → ray tracer

Modular programming:

- header files:
  (.h)

API of what you wrote

Application Program Interface

like an ad for users

- implementation files (.c) : contents of functions you advertised in header .h + .c } form a module

- example: how to use C math lib, e.g., use sqrt() function

① Look up documentation

% man sqrt

↑

Unix prompt

(2) put #include <math.h>

header

<> say to look elsewhere beyond ./

(3) Compile & link (-lm)

(-lm for cwd

- -lm tells linker to find libm.a file (usually in /usr/lib)

- math.h comes from /usr/include

- Compiler "knows" to look in /usr dir

- When you create your own lib (Lab 1) you have to tell compiler where to find:

a) header file (.h)

-I flag in gcc line

b) library object file (.a)

-L flag

eg.:

-I/usr/include -I./

-L/usr/lib -L./

- what's in a .a file?

a bunch of .o files (object)

- Lab 1: build your own lib called libvec.a

gcc -L./ ... -lvec (main.o)

- purpose of vector lib?
  math ops on vectors
  & vector defn. (type)

① – you'll need to:

specify vector.h

+ vector.c

- vector.h :

typedef double vec_t[3];

(just an array)

when A type

vec_t p; same as

double[3] p

& prototypes:

```
Vec_sum(Vec_t, Vec_t, Vec_t);
Vec_mult(...);
Vec_diff(...)
.
.
.
```

(2) #include "vector.h"

(or #include <vector.h>)

in your main.c &

use it

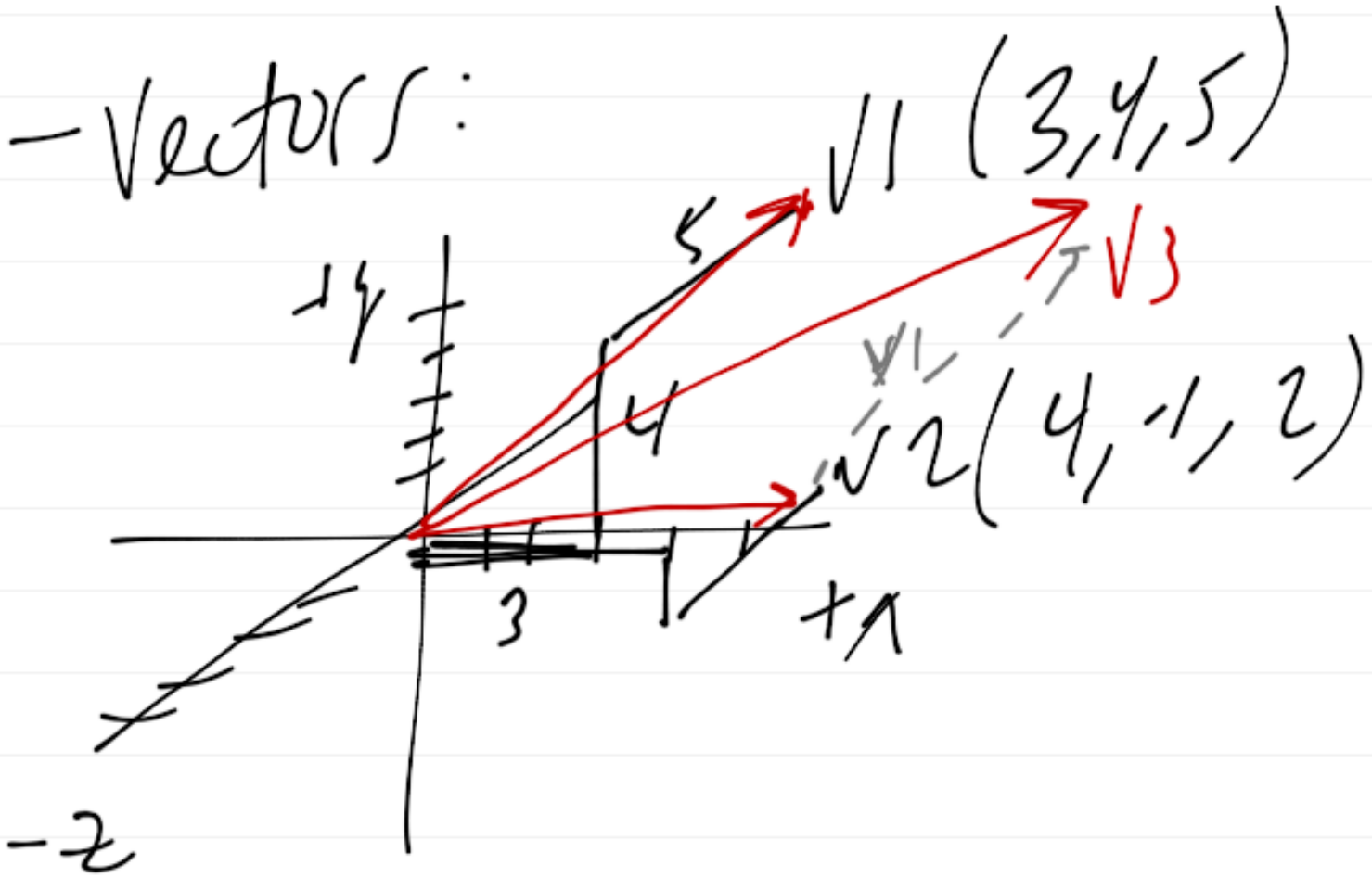
Example

in main.c:

```
vec_t v1 = { 3.0,
             4.0,
             5.0 };
vec_t v2 = { 4.0, -1.0, 2.0 };
vec_t v3;

vec_sum(v1, v2, v3)
vec_fprint(stderr, v3);
```

- Vectors:



V1+V2 = (7,3,7)