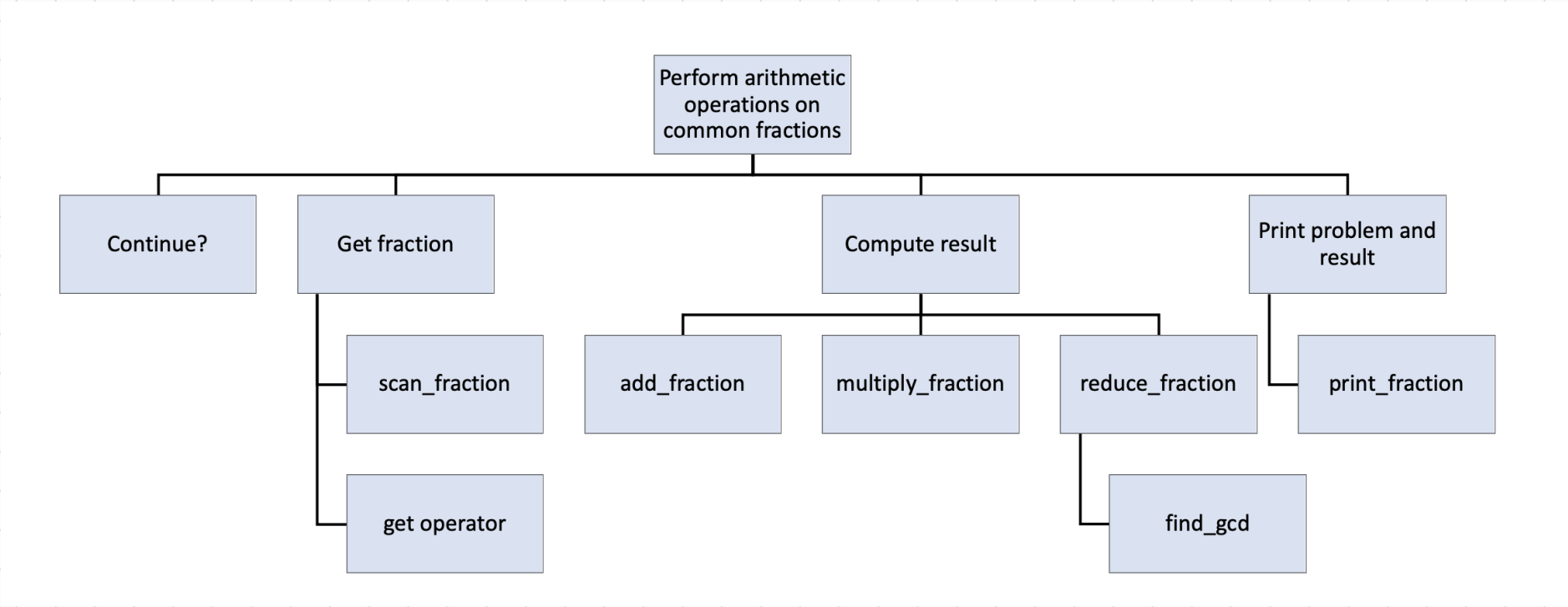
Testing

Wei-Mei Chen

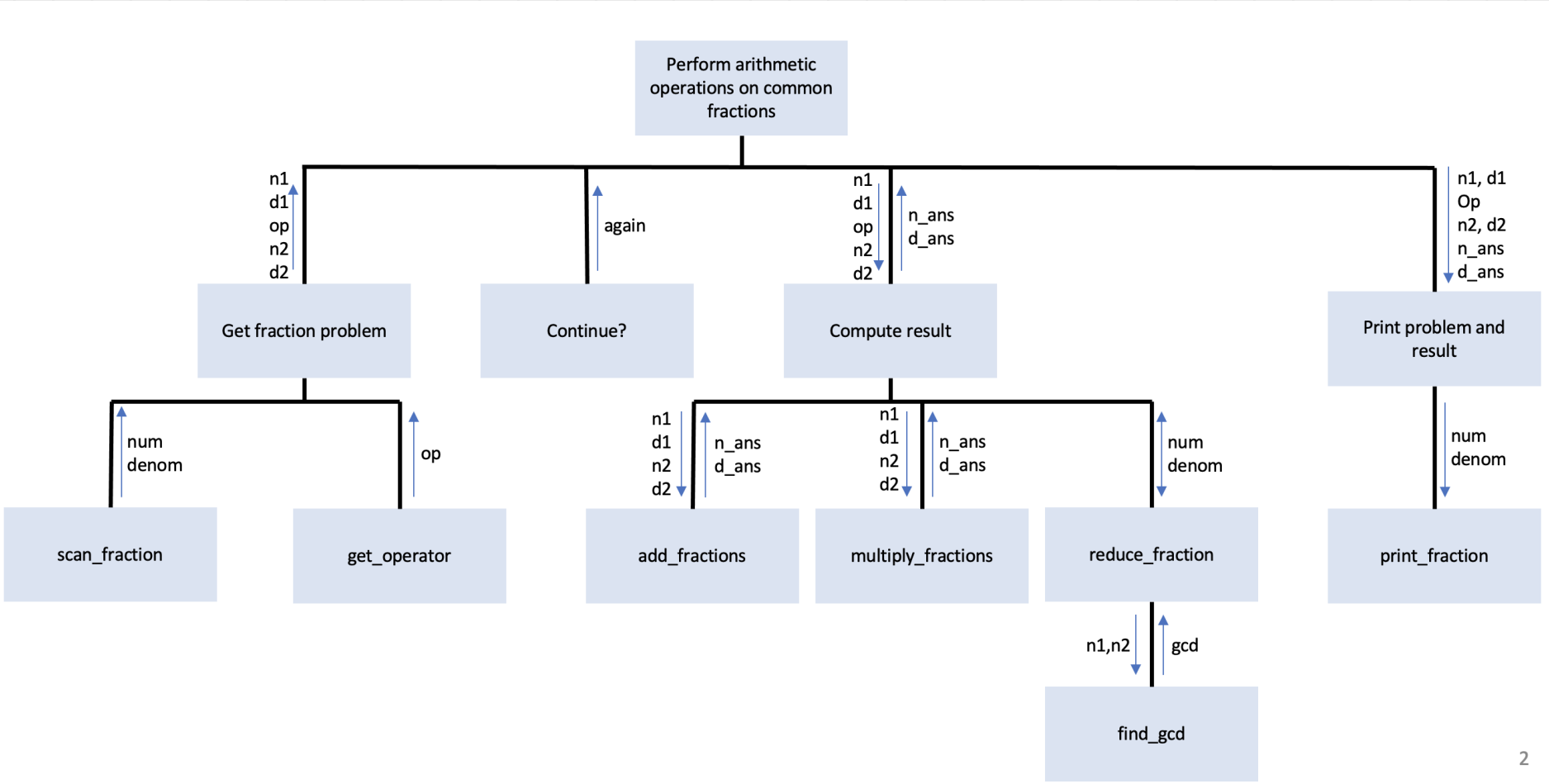
Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology

Structure Chart for a Computing Problem (1/2)

Data type: **fraction** p/q



Structure Chart for a Computing Problem (2/2)



Debugging and Testing a Program System (1/3)

- **Unit Testing**
  - Test the smallest testable piece of the software, a single function.
  - Write a short driver function to call the function tested.
  - The **driver** should give values to all input and input/output parameters.
  - After calling the function, the driver should display the function results.

```c
int
main(void)
{
        int num, denom;
        printf("To quit, enter a fraction with a zero numerator\n");
        scan_fraction(&num, &denom);
        while (num != 0) {
                printf("Fraction is %d/%d\n", num, denom);
                scan_fraction(&num, &denom);
        }

        return (0);
}
```

Debugging and Testing a Program System (2/3)

- **Integration Testing**
  - Test the interactions among functions.
  - Test functions that are dependent on other functions whose unit tests may not be complete requires a temporary function called a **stub**.
  - A stub has the same header as the function it replaces but its body displays only a message indicating that the stub was called.

```
void
multiply_fractions(int      n1, int d1, /* input - first fraction
                   int      n2, int d2, /* input - second fraction
                   int *n_ansp,         /* output -
                   int *d_ansp)         /* product of 2 fractions
{
    /* Displays trace message */
    printf("\nEntering multiply_fractions with\n");
    printf("n1 = %d, d1 = %d, n2 = %d, d2 = %d\n", n1, d1, n2, d2);

    /* Defines output arguments
    *n_ansp = 1;
    *d_ansp = 1;
}
```

Debugging and Testing a Program System (3/3)

- **System Testing**
  - Test the whole program in the context in which it will be used.
  - A program may need to be tested with other programs and hardware.
  - Show that the program meets its **functional requirements**
- **Acceptance Testing**
  - typically involves use of the system in the **real** environment or in a close approximation to the real environment