

People learn something every day,
and a lot of times it's that what they
learned the day before was wrong.
~Bill Vaughan

7. (a) Why do we write modular programs? Compare the two programs on NRIC check code below. Why do you think the modular one is preferred?

```
// Non-modular
int main(void) {
    int number;
    char code;
    int dig1, dig2, dig3, dig4, dig5, ...

    printf("Enter NRIC number: ");
    scanf("%d", &number);

    // determine check code
    . . .
    printf("Check code is %c\n", code);

    return 0;
}
```

```
// Modular
int main(void) {
    int number;
    char code;

    printf("Enter NRIC number: ");
    scanf("%d", &number);
    code = generateCode(number);
    printf("Check code is %c\n", code);

    return 0;
}

// This function ...
char generateCode(int num) {
    char code;
    int dig1, dig2, dig3, dig4, dig5, ...
    . . .
    return code;
}
```

Answer:

Reuse of codes; top-down design; ease debugging; ease replacement of function without affecting other parts of program.

- (b) Why is the following function considered bad?

```
// This function ...
void generateCode(int num) {
    char code;
    int dig1, dig2, dig3, dig4, dig5, ...
    . . .

    printf("Check code is %c\n", code);
}
```

Answer:

Leave out the printf() statement. Should just return computed check code to caller. As much as possible do not mix input/output with computation in a function. This will be discussed in **Unit 5 slide 41 Function Cohesion**.

9. Trace the program below manually. Determine the values of the variables `a`, `b`, and `c` in the **main()** function, and the parameters `a`, `b`, and `c` in the **confuse()** function at every step.

What are the final values of `a`, `b`, and `c` in the **confuse()** function just before control returns to the **main()** function, and what are the final values of `a`, `b`, and `c` in the **main()** function?

```
#include <stdio.h>

int confuse(int, int, int);

int main(void) {
    int a = 6, b = 2, c = 5;
    a = confuse(c, b, a);

    return 0;
}

int confuse(int b, int a, int c) {
    a = b + c;
    c = a * b;

    return c - a + b;
}
```

Answer:

Program: **confusing.c**

Final values of `a`, `b`, `c` in **confuse()**:

a	11	b	5	c	55
---	----	---	---	---	----

Final values of `a`, `b`, `c` in **main()**:

a	49	b	2	c	5
---	----	---	---	---	---

Students should be aware of the scope of variables/parameters, and that the value of a local variable in the **main()** function is not changed by the called function (unless we use `&`, which is a topic to be covered later).

10. Write a program **triangle.c** to request for data of 3 points on a 2-dimensional plane forming the vertices of a triangle, and compute the perimeter of the triangle. Each point is represented by two non-negative integers in the x-coordinate and y-coordinate.

You may assume that the input data are always valid and they represent the vertices of a triangle.

The perimeter is to be displayed in 2 decimal places.

Your program is to include a user-defined function called **distance()** that returns the distance (of **double** type) between two points. Your function may call some appropriate functions in `math.h`.

A sample run of the program is given below. User's inputs are in bold.

```
Enter 1st point: 1 5
Enter 2nd point: 3 3
Enter 3rd point: 7 4
Perimeter of triangle = 13.03
```

Answer:

See **triangle.c**