Learn as much as you can while you are young, since life becomes too busy later. ~ *Dana Stewart Scott*

CS1010 Programming Methodology

Week 5: Selection Statements and Repetition Statements

To students:

- Some of the programs shown here are available on the CS1010 website, "CA" → "Discussion": http://www.comp.nus.edu.sg/~cs1010/3_ca/discussion.html
- Please be reminded that the **deadline for Lab #2 is 13 September, Wednesday, 5pm!**

I. Syntax and Good Programming Habits

1. The following programs either do not work or work but are very badly written. Explain why and how would you correct/improve them?

   (a) **bad_prog1.c**: The gcc compiler issues a warning for this program. Why?

   If you ignore the warning and go ahead to run it, and enter **0.5** as the input, what result do you get? Why? (Hint: if you add this **printf**() statement just before the **if** statement, `printf("%d\n", 0 < value);` what output does it give?)

   (Always compile your programs with the **–Wall** option, and do not ignore warnings. The compiler must have some good reason to alert you through its warnings.)

```
// This program checks whether a user-input value
// is between 0 and 1, non-inclusive.
#include <stdio.h>

int main(void) {
   float value;

   printf("Enter value: ");
   scanf("%f", &value);
   if (0 < value < 1) {
      printf("%f is between 0 and 1\n", value);
   }
   else {
      printf("%f is not between 0 and 1\n", value);
   }
   return 0;
}
```

   (b) Suppose you correct the program in (a), but remove the ampersand (&) in the `scanf` statement. You ignore the compiler's warnings and go ahead to run the program. What will happen?

(c) **bad_prog2.c**

A list is in *non-decreasing order* if any value in the list is never smaller than its preceding value. For example: (-5, 6, 9, 12), and (2, 8, 8, 15, 15). (If any value must be larger than its preceding value, the list is said to be *strictly increasing* or *monotonically increasing*.)

Comment on the program below and improve it.

```
// This program checks whether 3 input values
// are in non-decreasing order.
#include <stdio.h>

int main(void) {
  int a, b, c;

  printf("Enter 3 integers: ");
  scanf("%d %d %d", &a, &b, &c);
  if (a <= b) {
    if (b <= c)
      printf("The values are in non-decreasing order.\n");
    else
      printf("The values are not in non-decreasing order.\n");
  }
  else if (a > b) {
    if (b <= c)
      printf("The values are not in non-decreasing order.\n");
    else
      printf("The values are not in non-decreasing order.\n");
  }
  else
    printf("The values are not in non-decreasing order.\n");

  return 0;
}
```

(d) **bad_prog3.c**

Improve the following program. To avoid re-computation and to aid checking, what variable do you think you may introduce and how are you going to make use of it?

```
#include <stdio.h>

int main(void) {
   // declare the first input and second input
   double the_first_input, the_second_input;

   // ask user to enter two values
   printf("Enter two values: ");
   scanf("%lf %lf", &the_first_input, &the_second_input);

   if (the_first_input/the_second_input < 90.2) {
      if (the_first_input/the_second_input < 32.5)
         printf("Paper\n");
      else if (the_first_input/the_second_input >= 45.8)
         printf("Ruler\n");
      else
         printf("Pencil\n");
   }
   else {
      if (the_first_input/the_second_input >= 100.0)
         printf("Unknown\n");
      else if (the_first_input/the_second_input < 100.0)
         printf("Eraser\n");
      else
         printf("Clip\n");
   }

   return 0;
}
```

(e) **bad_prog4.c**

In Unit #3 slides 5-7, we discussed the GCD algorithm. The program below includes the GCD function. Trace the code to verify that it is correct. You should discover that the assumption of *A* > *B* in the Euclidean algorithm (slide 5) is not necessary.

```
// This program computes the GCD of two non-negative integers,
// not both zeroes.
#include <stdio.h>

void GCD(int, int); // prototype for GCD function

int main(void) {
   int num1, num2;

   printf("Enter two non-negative integers, not both zeroes: ");
   scanf("%d %d", &num1, &num2);
   GCD(num1, num2);

   return 0;
}

// This function computes the GCD of a and b
// Pre-cond: a and b are both >= 0, and not both = 0
void GCD(int a, int b) {
   int r;  // r is the remainder of a/b

   while (b > 0) {
      r = a%b;
      a = b;
      b = r;
   }

   printf("The GCD is %d\n", a);
}
```

However, the function is considered <u>not</u> well written, although it is correct, as it violates some principle of programming. Do you know why?

II. Exploration

2. **Conditional operator ? :**

(a) There is a *conditional operator* **? :** which is commonly used in place of the **if** statement wherever appropriate. Study the program **cond_op1.c** below, run it and test it with several inputs.

```
// Illustrating the conditional operator ? :
#include <stdio.h>

int main(void) {
   int n, p;

   printf("Enter an integer: ");
   scanf("%d", &n);

   p = ((n > 5) && (n < 20)) ? 33 : -77;
   printf("p = %d\n", p);

   return 0;
}
```

(b) Do you know how to use the conditional operator "**? :**" now? Try to replace the **if** statement in the following program **cond_op2.c** with the conditional operator.

```
#include <stdio.h>

int main(void) {
   int a, b, max;

   printf("Enter 2 integers: ");
   scanf("%d %d", &a, &b);

   if (a > b)
      max = a;
   else
      max = b;

   printf("max = %d\n", max);

   return 0;
}
```

Replace the **if** statement in the box with the conditional operator.

(c) Exploration: You have used format specifiers in a **printf**() statement, such as **%d** for integers, and **%f** for float and double values. There are others, such as **%c** for characters, and **%s** for strings. We will explore **%s** here.

The following two **printf**() statements are equivalent:

```
printf("Good morning, David!\n");
printf("Good %s, %s!\n", "morning", "David");
```

Here we use string literals such as "morning" and "David" in the print list (see lecture notes: Unit #4 slide 19) of the second statement, but in general you can use string variables here. We will discuss string variables another time.

Can you modify the program **cond_op3.c** below by using the **%s** format specifier and the conditional operator?

```
#include <stdio.h>

int main(void) {
   int n;

   printf("Enter a non-negative integer: ");
   scanf("%d", &n);

   if (n < 2)
      printf("There is %d person.\n", n);
   else
      printf("There are %d persons.\n", n);

   return 0;
}
```

3. Study the following program **switch_to_if.c**.

   You see a new data type here: the **char** type, which stands for character. Character constants are enclosed in single quotes, for example, 'A', 'w', '+'. The format specifier for character in a **printf**() statement is **%c**.

   What is the output if the user enters 12? Replace the **switch** statement with an equivalent `if` statement.

```
// To convert switch statement into if statement.
#include <stdio.h>

int main(void) {
   int score;
   char grade;

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

   switch (score) {
      case 10:
      case 9:
      case 8:  grade = 'A'; break;
      case 7:
      case 6:  grade = 'B'; break;
      case 5:  grade = 'C'; break;
      default: grade = 'F';
   }
   printf("Grade is %c.\n", grade);

   return 0;
}
```

III. Problem Solving with Selection Statements

4. Study the following program **if_to_switch.c**.

   This program differs from the one in question 3 above in that the input is a real number from 0.0 to 100.0. You may assume that the user always enters a non-negative score that is at most 100.0.

```
// To convert if statement into switch statement.
#include <stdio.h>

int main(void) {
   float score;
   char grade;

   // You may assume score entered is >= 0.0 and <= 100.0
   printf("Enter score: ");
   scanf("%f", &score);

   if (score >= 80.0)
      grade = 'A';
   else if (score >= 60.0)
      grade = 'B';
   else if (score >= 50.0)
      grade = 'C';
   else
      grade = 'F';

   printf("Grade is %c.\n", grade);

   return 0;
}
```

   Convert the **if** statement into a **switch** statement. The challenge here is that the input **score** is a real number. Think about it. If you need help, post on IVLE, in the "Discussion sessions" forum.

5. Modularise the **switch_to_if.c** program given in question 3, that is, write a function **compute_grade(int mark)** to take in the mark (an integer) and return the corresponding grade (a character). This function is to be called in the main() function.

6. Given a person's weight in kilograms and height in metres, his/her BMI (Body Mass Index) is calculated based on this formula:

$$BMI = Weight / Height^2$$

The following table shows the body types according to a person's gender and BMI:

	Female	Male
Underweight	BMI $\leq$ 19	BMI $\leq$ 20
Acceptable	BMI > 19 and $\leq$ 24	BMI > 20 and $\leq$ 25
Overweight	BMI > 24	BMI > 25

Write a program **bmi.c** to do the following:

1. Read the user's gender (type int), weight (type double) and height (type double).
2. Call a function **body_type()** that takes in the above values, and returns the body type which is an integer. You need to find out what are the parameters for this function.
3. Upon obtaining the body type, display a suitable advice for the user.

The gender is encoded using the following integers:

- 0 to represent female
- 1 to represent male

The body type is encoded using the following integers:

- -1 to represent underweight
- 0 to represent acceptable
- 1 to represent overweight

You are to define constants for the above integers, and use the **switch** statement wherever possible.

Sample runs are shown below:

```
Enter your gender (0 for female, 1 for male): 0
Enter your weight (kg) and height (m): 62 1.6
Time to join the gym!
```

```
Enter your gender (0 for female, 1 for male): 1
Enter your weight (kg) and height (m): 62 1.6
Great! Maintain it!
```

```
Enter your gender (0 for female, 1 for male): 1
Enter your weight (kg) and height (m): 61.5 1.8
Stuff yourself with more food!
```

IV. Repetition Statements I

7. **Conversion of loop construct.**

   You have learned 3 loop constructs: **for**, **while** and **do-while**. For each of the following parts, a particular loop construct is used. Rewrite the loop construct using the other 2 loop constructs (without adding any **if** statement), and indicate if there is any limitation on the new codes. You may assume that all variables are of type **int**.

   (a)

```
sum = 0;
i = -5;
do {
    sum += i;
    i += 5;
} while (i < 100);
printf("sum = %d\n", sum);
```

   (b) This **for** loop may look strange, but after converting it into the other 2 loop constructs you should understand what it does. Can you describe what the code does in words?

```
printf("Enter n: ");
for (scanf("%d", &n); n < 0; scanf("%d", &n)) {
    printf("Enter n: ");
}
printf("n = %d\n", n);
```

   Suppose the valid range of values for **score** in question 3 should be 0 to 10 inclusive. Would you now be able to write a code to check that the user's input is valid (and keep asking if it is invalid) before going on to the **switch** statement?

   (c)

```
printf("Enter n: ");
scanf("%d", &n);
i = 0;
while (i < n) {
    printf("*");
    i += 3;
}
printf("\n");
```

8. **Manual tracing.**

(a) Spot an error in this program and correct it. Then manually trace the program and write out its output.

```
int i, sum;

for (i = 1; i < 1000; i*=2) {
    sum += i;
}
printf("sum = %d\n", sum);
```

(b) Manually trace the program and write out its output.

```
int a = 10, b = 200;

while ((a*a) < (a+b)) {
    printf("a = %d, b = %d\n", a, b);
    a++;
    b+=10;
}
```

(c) Manually trace the program and write out its output.

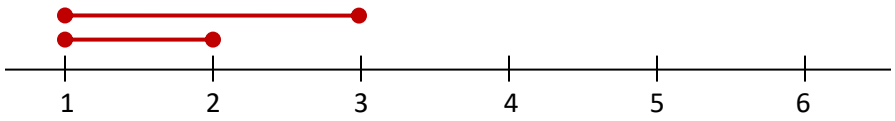
```
int x, y, count = 0;

for (x = 1; x <= 6; x++)
    for (y = x + 1; y <= 6; y++)
        count++;

printf("count = %d\n", count);
```

(d) Suppose we have a function **draw_line(int a, int b)** which draws a horizontal line segment from a to b. If you replace the statement **count++;** in part (c) above with the statement **draw_line(x, y);**, how would the line segments be drawn? The diagram below shows the first two line segments drawn (a new line segment is drawn above an old one). Complete the diagram.

[You may look up the CS1010 website ("CA" → "Discussion") after 13th September for the answer, but only after you attempt the question.]



(e) Manually trace the program and write out its output.

```
int x, y, z, count = 0;

for (x = 1; x <= 5; x++)
    for (y = x; y <= 5; y++)
        for (z = x; z <= y; z++)
            count++;

printf("count = %d\n", count);
```