CS1010 Programming Methodology
Week 4: Functions and Selection Statements

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

To students:

The success of discussion sessions hinges very much on your (1) **PREPARATION** beforehand, (2) **ACTIVE PARTICIPATION** in class, and (3) after-class **REVISION**. Unless otherwise instructed, you do not need to submit your work for grading.

Please cooperate with your DL to work towards a fruitful and enriching learning experience.

Due to time constraint, not all the questions in this Discussion Sheet are discussed in class. Your DL has the discretion to choose the questions to discuss (or you may request your DL to discuss certain questions). Please do go through those left-over questions after class.

Most programs can be downloaded from cs1010 account in the way shown to you last week.

Please be reminded that **the deadline for Lab #1 is this Saturday 6pm!**

I. C Basics and Exploration

1.

(a) Download the following program from cs1010 account:

```
cp ~cs1010/discussion/Week4_Q1a.c .
```

Compile this program without and with the **–Wall** option. What do you observe? What is the output of the program?

```
#include <stdio.h>

int main(void)
{
    int i;
    float f;
    double d;

    printf("i = %d; f = %f; d = %lf\n", i, f, d);
    return 0;
}
```

Initialize variables.

(b) What are the values assigned to the variables?

```
int a, b, c;
a = b = c = 12;
```

(c) What are the values assigned to the variables?

```
float a;
int b;
double c;
a = b = c = 12.98;
```

(d) What is the output of the following program **Week4_Q1d**?

```
#include <stdio.h>

int main(void)
{
    int a, b, c, d, e, f;

    a = 3;
    b = a++ + 10;
    printf("a = %d; b = %d\n", a, b);

    c = 3;
    d = ++c + 10;
    printf("c = %d; d = %d\n", c, d);

    e = f = 3;
    f *= 2 + e;
    printf("e = %d; f = %d\n", e, f);

    return 0;
}
```

II. Functions

You have learnt functions which do not return any value (void functions) and functions which return one value (through the return statement).

2. Spot the errors in the program below and correct them. Keep the function `func` below the `main` function.

```
#include <stdio.h>

int main(void)
{
    void func(5);
    void func(3-7);

    return 0;
}

void func(y)
{
    if (y<0)
        printf("Nothing\n");
    else
        printf("Something\n");
}
```

We would require students to provide function prototypes at the top, and write the function definitions after the main function.

3. Write a program **triangle.c** to request for data of 3 points on a 2-dimensional plane forming 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.

   You program is to include a user-defined function called **distance** that returns the distance (a value 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
```

4. 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;
}
```

III. Selection Statements

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

   (a) **Week4_Q5a.c**: The gcc compiler issues a warning for the following program. Why?

```
// 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;
}
```

Download source code from cs1010 account

   If you ignore the warning and go ahead to run it, and enter 0.5 as the input, what result do you get? Why?

   Conclusion: 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.

   (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?

Tip: What can you do if you get a "core dump"?

Students often ask us, why does my program get a "core dump"? The answer is, there can be 101 reasons your program gets a "core dump", e.g., division-by-zero error or missing & in scanf, etc. Hence, it is IMPOSSIBLE for us to tell you right away what happened, unless we take a look at your code.

A "core dump" happens when a program terminates abnormally (crashes). The system then dumps the content of the memory into a file called "core". This name comes from the old days when magnetic core memory was used.

The file "core" is usually very huge, so it is advisable that you delete it (use UNIX command "rm core"). What follows, of course, is that you have to study your code to find out what went wrong.

(c) **Week4_Q5c.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;
}
```

6. **Conditional operator ? :**

(a) There is a *conditional operator* **? :** which is commonly used in place of the 'if' statement wherever appropriate. Study the program **Week4_Q6a.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 **Week4_Q6b.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;
}
```

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

$$\text{BMI} = \text{Weight} / \text{Height}^2$$

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 the parameters are 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. User's inputs are in bold.

```
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!
```