> It's hard enough to find an error in your code when you're looking for it; it's even harder when you've assumed your code is error-free.
> ~ *Steve McConnell*

CS1010 Programming Methodology

Week 7: Pointers and Functions with address parameters

To students:

Some programs for this session are available on the module website:

http://www.comp.nus.edu.sg/~cs1010/3_ca/discussion.html

Please be reminded that the **deadline for Lab #3 is this Saturday 9am!**

I. Review of PE1

Your DL will review PE1 with you. If you have doubts on PE1 exercises, please discuss.

II. Hi-Lo game

Your DL will discuss the Hi-Lo program given in Week 6.

III. Pointers

You learned pointers (addresses) in Week 6. Do you really understand how they work? For each of the following, trace the code. Line numbers are added for ease of reference.

1. **Understanding pointers and addresses.**
   You learned pointers (addresses) in Week 6. Do you really understand how they are used? For each of the following, trace the code. Line numbers are added for ease of reference.

   (a) What is the final value of **i**?

```
int i = 1;      // Line 1
int *p;         // Line 2

i += 2;         // Line 3
p = &i;         // Line 4
(*p)++;         // Line 5
*p = *p + i;    // Line 6
```

(b) What is the output?

```
int i = 1, j = 1;       // Line 1
int *p, *q,             // Line 2

i += 2;                 // Line 3
p = &i;                 // Line 4
*p = *p + 3;            // Line 5
q = &i;                 // Line 6
i = *p + *q;            // Line 7
printf("i = %d\n", i);  // Line 8
q = &j;                 // Line 9
i = *p + *q;            // Line 10
printf("i = %d\n", i);  // Line 11
p = &j;                 // Line 12
i = *p + *q;            // Line 13
printf("i = %d\n", i);  // Line 14
```

(c) [AY2012/2013 Semester 1 Exam Paper]
What is the output of the following program?

```
#include <stdio.h>

int f(int *, int);
int g(int *);

int main(void) {
    int x = 5, y = 5;
    int sum = f(&x, y);
    printf("%d %d %d\n", sum, x, y);
    return 0;
}

int f(int *x, int y) {
    int sum = 0;

    while (*x > 0) {
        sum += g(&y);
        *x -= 1;
    }
    return sum;
}

int g(int *y) {
    *y *= 2;
    return *y;
}
```

2. **Lab1 Ex2: Surface area and Longest Diagonal of a Box**
   In Lab 1 Exercise 2, you wrote two functions **compute_surface_area(int, int, int)** and **compute_diagonal(int, int, int)** to compute the surface area and longest diagonal of a box respectively. The program is available as **box.c** which you may copy from the cs1010 account.

   ```
   cp ~cs1010/discussion/prog/week7/box.c .
   ```

   Can you combine the two functions into one, called **compute_surface_area_and diagonal()**, which passes back both the surface area and length of the longest diagonal?

IV. Design Issues: Programming Methodology and Cohesion

3. After learning about function with address parameters, Brusco is so excited that he replaced this GCD function (see Week 6 discussion question 3):

```
int brusco_gcd(int a, int b) {
    . . .

    return a;
}
```

with the following function:

```
void brusco_gcd(int a, int b, int *answer) {
    . . .

    *answer = a;
}
```

He did not make a wise move? Why?

4. After learning in question 3 above that he should stick to his old function instead of using address parameter in his GCD function, Brusco, being a very inquisitive and adventurous student (and we all love such student!), tried another new version:

```
void brusco_gcd(int a, int b) {
    . . .

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

His reason being: since the answer (variable a) is to be returned to the caller and get printed anyway, why can't he just save the returning part (and hence make the function a void function) and print the answer inside the function instead?

Comment on his move.

5. **Lab 1 Ex2: Surface area and Longest Diagonal of a Box – Revisit**
In question 2 you attempt to combine the two functions **compute_surface_area()** and **compute_diagonal()** into a single function **compute_surface_area_and_diagonal()**.

Compare the two approaches. Which one do you think is more desirable in terms of good programming methodology?

V. Programming Exercises

6. **Sorting three numbers.**
Sorting is an important topic in Computer Science. You will learn some sort algorithms in week 10. Normally we perform sorting on an array of elements. However, the topic of arrays is covered this week so we won't use array here. Here, we attempt to sort 3 integers stored in 3 variables.

Write a program **sort3.c** to read 3 integers into the variables num1, num2, and num3, and sort them in non-descending order. Examples of 3 values in non-descending order are: {12, 32, 57}, {48, 92, 92}, {-11, -11, 2} and {20, 20, 20}.

Your program must contain the function **sort()** to do the sorting. You must decide the parameters for this function. There should be no printf() statement inside the function. The main() function should print out the values of num1, num2, num3 after the sorting.

The following is a sample run with input values shown in bold:

```
Enter 3 integers: 57 12 32
After sorting: 12 32 57
```

There are many ways to do sorting. Here, we adopt the following:

- Move the largest value into num3 by comparing num1 and num2 and swap their values if necessary, and then by comparing num2 and num3 and swap their values if necessary.
- Move the second largest value into num2 by comparing num1 and num2 and swap their values if necessary.
- After the above steps, num1, num2 and num3 would be sorted. We have made 3 comparisons and at most 3 swaps.

Do not write any additional function. We will improve on this program in the next question. A skeleton program **sort3.c** is given which you may copy:

```
cp ~cs1010/discussion/prog/week7/sort3.c .
```

7. **Sorting three numbers – version 2**

You should have noticed that the swap operation is needed more than once in the program for question 6. The proper way to do this is to define a function **swap()** and use this function. The function **swap()** is discussed in Week 6 lecture slide 19, or you may download the demo program **Week6_SwapCorrect.c** from the CS1010 "Lectures" page:

http://www.comp.nus.edu.sg/~cs1010/2_resources/lectures.html

Rewrite your program **sort3.c** into **sort3v2.c** to include the **swap()** function, and call the **swap()** function appropriately in your **sort()** function.

8. **Triangle incenter**

In week 6 lecture exercise #3, you are to compute the centroid of a triangle. Besides the centroid, there are other "centers" of a triangle: circumcenter, orthocenter and incenter. You may refer to

http://jwilson.coe.uga.edu/emat6680/dunbar/assignment4/assignment4_kd.htm

Here, you are to write a program **incenter.c** to compute the incenter of a triangle given its three vertices. Google to search for the formula to compute the coordinates of the incenter. Your program should contain a function **incenter()**. You may use **float** type for all values.

Two sample runs are shown below. The coordinates of the incenter are printed in 2 decimal places.

```
Coordinates of 1st vertex: -1 0
Coordinates of 2nd vertex: 3 0
Coordinates of 3rd vertex: 1 5
Coordinates of incenter = (1.00, 1.35)
```

```
Coordinates of 1st vertex: 63.2 21.8
Coordinates of 2nd vertex: -15 -6
Coordinates of 3rd vertex: -19.2 5.7
Coordinates of incenter = (-11.52, 1.34)
```

VI. Lab #3 Exercise 1: Hourglass

Read up Lab #3 assignment on the module website:

http://www.comp.nus.edu.sg/~cs1010/3_ca/labs.html

A worksheet on Lab #3 Exercise 1 will be given out during the discussion session. (Some groups might have done this in week 6.)