CS1010 Programming Methodology

Week 3: Algorithms and C Basics

> No matter how one may think himself accomplished, when he sets out to learn a new language, science, or the bicycle, he has entered a new realm as truly as if he were a child newly born into the world. ~ Frances Willard, *How I Learned to Ride the Bicycle*

To students:

Discussion sessions are small group forums where **you** present and discuss answers to the questions, and raise your doubts for clarification. Answers come from you, not from your discussion leader (DL). Your DL and classmates can then build on your contribution to bring up any intricate points that have been overlooked, or to correct any misconceptions.

The success of discussion sessions hinges very much on (1) your **PREPARATION** beforehand, (2) your **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, sometimes not all the questions in the Discussion Sheet are discussed in class. Your DL has the discretion to choose the questions (or you may request your DL to discuss certain questions) or set his/her own questions for you. You may continue to discuss the questions on IVLE forums after class.

Also, why limit yourself only to the exercises here? You may find exercises from other sources for your own practice.

I. Algorithms

You may assume that for the problems below, the input data have been read into the list. That is, you need not concern yourself about how to read the data into the list.

1. Given a list of N integers, find out how many of them are negative.
   An algorithm is shown below, which includes all 3 **control structures**: sequence, selection and repetition. For a list, we may use the subscript (such as a_k) to represent individual items in the list.

   Study the algorithm and trace it with some test data.

   Let the list of integers be a_1, a_2, … a_N.

```
for $k$ from 1 to $N$
    if ( $a_k < 0$ ) then countNeg ← countNeg + 1
print countNeg
```

   To implement this algorithm in C, what type should the variable **countNeg** be?
   Is **c**, instead of **countNeg**, a good variable name?
   Can you spot an error in the algorithm and correct it?

2. N white and black balls are arranged in a row. The example below shows a case of N=7.

   ●●○●○●○

   Of course, we shall choose an appropriate notation to represent our objects (balls): **B** for black ball and **W** for white ball. So the above may be presented as BBWBWBW.

   The task is to determine the least number of 'swaps' you need to shift all the white balls to the left of all the black balls, subject to the condition that you may only swap two neighbouring balls. For our example above, the least number of swaps you need is 9. (Work it out yourself! Try out other examples as well.)

   Write an **algorithm** (not a C program!) to compute the answer. For this problem, a good algorithm needs not even carry out the swapping.

   To help you start, here's a suggestion:

   In solving algorithmic problem, we must name our data. Let's name the balls, from left to right, B_1, B_2, B_3, …, B_N for N balls.

3. Go to CS1010 module website → CA → Labs and click on "Lab #1: Three Simple Exercises". Read the three exercises.

   You will receive a worksheet for Exercise 3 in class to be filled in.

II. Programming Environment

In this section, we will go through the following which you should already know by now, just to make sure that you clarify your doubts:

- Logging into UNIX system
- Using basic UNIX commands
- Using editor (vim) to write/modify programs
- Using compiler (gcc) to compile programs

<u>You should try questions 4 and 5 at home before you come for the discussion session.</u>

4. The source codes in the reference book are available for download. You may do it in one of these two ways:

    a) Go to CS1010 website → Resources → Books → Source Codes and download the required program into your computer's hard-disk. Use **SSH Secure File Transfer Client** or any other appropriate software to transfer the program into your UNIX account; OR

    b) Log into your UNIX account and copy the program from the CS1010 account.

    We will try out method (b) here. The directory

    ```
    ~cs1010/public_html/C_Programming/SourceCodes/Chapter2
    ```

    contains the following programs:

    L2_1.c, L2_2.c, L2_3.c, L2_4.c, L2_5.c

    To copy the program L2_1.c into your own directory, issue this command:

    ```
    cp  ~cs1010/public_html/C_Programming /SourceCodes/Chapter2/L2_1.c  .
    ```

    (Note the dot at the end of the command. It denotes the *current directory* which is your destination directory to hold the copy of L2_1.c. The dot <u>must be present</u>, or the cp command will not work. If you want to copy the file into a different directory, replace the dot with the directory name accordingly.)

5. After you have copied **L2_1.c** into your own directory, compile and run it.

    We suggest that you use the –Wall option in gcc to turn on "all warnings":

    ```
    gcc  -Wall  L2_1.c
    ```

    However, if you type the above you will get a compiler warning. This is because the book uses "void main(void)" in the source codes, whereas gcc prefers to have type int for main() function of. Hence in our lectures, we use the header "int main(void)" instead.

6. Refer to question 5 above. If you compile L2_1.c using the following command:

   gcc -Wall L2_1.c

   the executable file that the compiler generates has the default filename of '**a.out**'. (This will overwrite any existing **a.out** file in that directory.)

   You may rename **a.out** to another name (say, L2_1) with the **mv** command:

   mv a.out L2_1

   This is tedious if you have to do it for every program. An alternative is to use a compiler option to compile a C source code into an executable file with a specific filename of your choice. Do you know how to do this?

7. **<u>A very common mistake</u>** made by students.

   Copy the program L3_1.c from Chapter 3 as follows:

   cp ~cs1010/public_html/C_Programming/SourceCodes/Chapter3/L3_1.c .

   Compile the program as follows:

   gcc -Wall L3_1.c

   You get a compilation error. What happens and why? How do you resolve it? (Hint: a particular compiler option is missing in this case.)

   Even if the above compilation error is resolved, you still get compilation warnings when you compile the program. Do you know why?

III. Programming

8. Spot the errors, bad programming practice and logic flaws in the following program. Compile the program and see if you understand the error messages.

```
#include <stdio.h>

int Main(void) {
float j, k, l;

scanf("%.2f", k);
k = l + 1.2;
printf("k = %.2f\n", k);
return 0;
}
```

9. Study **L2_5.c**. It includes certain operators which are not covered in class, for example:

```
    k = i++;
    h = ++j;

    k1 += 2;
    k2 -= i;
    k3 *= (8/4);
    k4 /= 2.0;
    k5 %= 2;
```

Do you know how to use them? Explain them in class.

IV. Using CodeCrunch

Refer to the "Introduction to CodeCrunch" on the module website ("CA" → "Labs").

You are to familiarize yourself with CodeCrunch. Though Trial Lab (lab #0) will not be graded, you are to submit Trial Lab exercises in the presence of your DL (even though you might have submitted it before on your own). Clear your doubts on CodeCrunch with your DL.

There are other practice exercises on CodeCrunch which you should attempt on your own too.