Lab Practice Week 6
To be submitted as C Exercise 2 by Week 9

Internal Students: You need to show a working version of your solutions to program 13 and 14. Your tutor will expect to see your submission on or before your lab class in week 9 as part of your C Exercise 2 assessment.

External Students: Please email your Program 13 and 14 to your tutor. Your tutor will expect to receive them by the end of week 9.

Program 13

Write a program in C that reads in two strings, STR1 and STR2 of exactly the same size. The program is to compare the two strings and output “Yes” if the two strings are identical, or else, output “No”. Note that Student should develop you own string comparison routine. The function “strcmp” should only be used for work validation purpose.

Assuming STR1 is longer than STR2, determine whether STR2 is “embedded” in STR1. If yes, consider the position where STR2 is found. Also, determine whether multiple copies of STR2 are embedded in STR1 and print their position as well.

Hint: This can be done via a For loop and/or nested loops

Applications: This is an illustration of “string processing”. It is used extensively in practice such as comparing password. Other applications of this program are to search for the occurrence of a particular string pattern. STR1 can be a document where STR2 can be embedded as a form of “signature”.

Program 14

There are three ways that a binary operator can be expressed with the operands that it works upon. For example, the addition operator (+), operates on two operands. The operation can be expressed as - prefix (+ 123 45), infix (123 + 45), and postfix (123 45 +).
Write and test a C program that accepts a string from the user in any of these three forms, and then calculate the appropriate result.

For instance:
- (125 45 +) should output 170
- (125 34 34) should output “Invalid Input”
- (12 + +) should output “Invalid input”
- (+ 1 1) should output 2

Hint: You should treat the statement as a string and then search for the operator. The function `strtok` from `string.h` can be used to tokenise the string input.

Applications: This is in fact how the compiler works! It “parses” the statements or expression and then determines how to do execute it with a sequence of low level instructions. The Assembly codes are actually similar to the Prefix arrangement:

e.g. `ADD AX, BX`

Question

If you compile and develop a C program on different compilers for different target systems with different OS and CPU, why are the sizes of executable codes different? Demonstrate with an example C program compiled with different compilers and OS.

Note: While you may develop your program in any platform or compiler, all programs are expected to be able to run in the Cygwin environment. Even you may not be able to produce the complete program in the lab session, you MUST record and show your work to your tutor, otherwise no marks will be given. You should be able to give the following information:

1. An overall design of the program (How does the program work?)
2. Algorithm of the program (How do you process the data?)
3. Code and comments (What have you developed?)
4. Results (What are the outputs from the program?)
5. Testing (How did you test it?)
6. Discussion (Does it work? If no, what will you do next? If yes, how can it be improved?)