# **CIS 241**

# **Lab 4: *play with pointers***

**Due:** Friday, Feb 14, 2020

**Objective**

This lab is designed to help students better understand pointer operations through string processing. GNU debugger will also be introduced.

**Description**

A DNA can be represented as an array of **N** characters, where **N** is a very large integer (on the order of tens of thousands). A problem in genetic research is to discover whether any pattern of length **LEN** is ever repeated in the array, where **LEN** is a fixed and small constant integer, say 4.

In this lab, you are going to modify and complete given code to perform such pattern search in a string.

**Your Tasks**

Please do the following for this lab:

* Make a directory this this lab, for example, cis241/labs/lab4. And go to this directory.
* Use the command below to download the source code **dna.c** from the instructor’s web page to your current directory.

wget [http://cis.gvsu.edu/~wangx/teaching/cis241/lab4/dna.c](http://cis/gvsu.edu/~wangx/teaching/cis241/dna.c)

Program **dna.c** is intended to determine whether a substring of length 4 in the given string is repeated. For the string that is stored in array **line**, this program would show that substring “book” is repeated. Its first occurrence is pointed to by pointer **p1** and its second occurrence by **p2**. The out loop in the **main ()** function allows **p1** to point to a substring in the given string and the inner loop allows **p2** to point to a substring after **p1**. As such, these two nested while-loops make it possible to compare every possible pair of substrings in a given string. If the substring of LEN characters that **p1** points to is identical to the one that **p2** points to, the results are displayed on the screen and the program terminates. If no match is located, the program displays such a message and terminates.

For this lab, you will write C code to replace each line of pseudo-code in the incomplete program. Please note that the given pseudo-code is intended to show the main logic of this program only. You may add additional statements to address what you find necessary.

* When your code works correctly, take the following steps to familiarize yourself with the GNU debugger **gdb**, which would be helpful when you test and debug your program.
* Type ***gcc –g dna.c –o dna***to prepare an executable for **gdb**
* Type ***gdb dna*** to run **gdb** on the executable
* Use ***list*** to show the first ten lines of code in **dna**
* Use ***list*** again to show the next ten lines of code in **dna**
* Use ***break lineNo*** to set one or more breakpoints; for example, one at the first line of the out loop and one at the first line of the inner loop
* Use ***run*** to run **dna** until the first breakpoint is reached
* Use ***print variable*** to check the current value of variable
* Use ***step*** or ***step 2*** to run one or two lines of code from where this program stopped
* Use ***cont*** to make this program continue from where it is to the next breakpoint or the end of this program if no more breakpoints are on the way
* Use ***quit*** to leave **gdb** when the program terminates

At the **gdb** prompt, you may use ***help* *command*** to find a description about the named command. Refer the PDF document “Unix Programming Tools” (Section 3) that is available from the following link:

<http://cslibrary.stanford.edu/107/UnixProgrammingTools.pdf>

GNU GDB documentation can be found from the following link:

<https://www.gnu.org/software/gdb/documentation/>

* When your program runs correctly, change the value of LEN from 4 to 5 and then run your program again. It is a case in which no match can be found.

Show your program to the instructor after you have done all the above successfully. If you cannot demo in class, please either demo at a later time or turn in your file to Blackboard by due time.