Pointers

7.1 Objectives

1. To familiarize the computation using Pointers, pass by address and returning address.

Time-span: 1 lab day (2 hrs.)

7.2 Problems

1. Declare two dynamic matrices and assign values from user input. Write your functions to perform the following operations on them.
   a. Addition
   b. Multiplication
   c. Addition of transpose matrices

   [*Hint:* Read size (row and column) of matrix and use malloc functions to allocate the dynamic memory and pass pointer to function for arithmetic operations as shown below]

   ```
   int** matrixAddition(int **m1, int**m2, int r,int c);
   ```

2. In this section, you need to implement the following string utilities functions using pointers. While writing program, be careful on exceeding the length of the string while iterating it.

   a. *stringCat*: this function concatenates two strings and returns the result string.

   ```
   char* stringCat (char *s1, char *s2);
   ```

   b. *stringToUpper*: this function converts a given string to its equivalent upper case string.

   ```
   char* strinToUpper (char *src);
   ```

   c. *stringToLower*: this function converts a given string to its equivalent lower case string.

   ```
   char* strinToLower (char *src);
   ```


   d. *subString*: this function returns the sub-string of the given string from the specified position.

   ```
   char* subString (char *src, int from, int len);
   ```

   For example: subString of *"Nepal"* from 0 of length 2 is *"Ne"*.

   e. *stringCompare*: this compares two strings, let str_1 and str_2. It returns -1 if $str_1 < str_2$, it returns 0 if both are equal and it return 1 if $str_1 > str_2$.

   ```
   int stringCompare (char *str1, char *str2);
   ```

   f. *stringReverse*: this functions returns the string in reverse order of the original string passed as an argument.

   ```
   char* stringReverse (char *src);
   ```

3. Suppose your class have **N** number of students. Write the following functions:

   a. *readNames*: this function allocates and read **N** students' names and returns the first address (pointer).

   ```
   char** readNames (int n);
   ```

   b. *sortNames*: this function sorts **N** names alphabetically and returns the first address (pointer) of sorted names.

   ```
   char** sortNames (char** srcNames);
   ```

***