COMPUTER PROGRAMMING

LAB ASSIGNMENT – 9

NOTES:

1. Please carefully read all questions and there is no choice.
2. Submit **6 individual .c files for the 6 problems.**
    a. Name the file as follows: S2020xxxxx_A9_P1.c for the solution for problem 1.
3. Do NOT zip. Just attach the .c files directly to your submission in google classroom.

PROBLEM INSTRUCTIONS:

1. For all problems write the solution as a function. You shall call the function from main. The function(s) and main for each problem should be in a separate file.
2. The functions **should not have a return statement** (hence all return-type should be void).
    a. If you are convinced that you can't implement a solution without a return statement. Write your justification as comments.
3. **All** the **arguments** to the functions **must** be **pointers**
4. Do **not use global or static** variables as a workaround to pass values around

Questions 1 to 5 have 4 points each. Question 6 has 4 bonus points. Your marks are guaranteed only when you complete the entire solution to the question. Partial marks are completely at the discretion of the faculty. No claims can be made for partial answers.

PROBLEMS:

1. Write a simple function to cycle('int') values stored in three variables.Example: If a =1, b =2, c=3 then after one cycle you get a=3, b=1, c=2 and so on. Example:

```
int main(){
  int a = 1, b =2, c =3;
  cycle(&a, &b, &c);
//now the value of a=3, b=1, c=2
  cycle (&a, &b, &c);
//now the value of a =2, b=3, c=1
….
}
```

   Improve the function and implement two other versions 'acycle' (anti-clockwise cycle, which is same as above) and 'ccycle' (clockwise-cycle).

2. Write a recursive function to compute the factorial of a given number.

3. Write a function which takes in an empty array of size 30 and fills it with the first 30 values of the function T(n) = T(n-1) + T(n-3) for n > 2. Where T(0)=0, T(1)=1,T(2)=1.
    a. The function should only populate, you should print the result in main() after the function call.

4. In main, accept a string of 'n' characters (taken as input from the user) and write a function which takes as input
    a. a pointer to the string
    b. Reverses the list of characters in place (i.e same memory location)
    c. The function should only reverse, you should only print the result in main()

5. In main, use an array to store the integers (taken as input from user) and write a function which takes as input
    a. the pointer to the first memory location
    b. an integer pointer to store the sum (result-pointer)
    c. an integer pointer which contains the length of the array.

` The function should access the array elements using the pointer, compute sum and populate the appropriate memory location with the result.

The below exercise is a bit more challenging than the rest. First, try to implement a version that works. While doing so, don't write the solution as a monolithic monster in main, instead, separate the solution into a nice collection of functions. Later, you can try to modify the functions to make use of pointers and become more memory efficient (i.e. supporting in-place operations).

6. For this assignment, you are to implement a simple transposition cipher. This cipher encrypts and decrypts a sequence of characters by dividing the sequence into blocks of size n, where n is specified by the encryption key. If the input text has a length that is not a multiple of n, the last block is padded with null characters ('\0'). In addition to n, the key also specifies two parameters a and b. For each block, the i-th output character, starting from 0 as usual, is set to the j-th input character, where

$$j = (ai + b) \bmod n.$$

For appropriate choices of a and b, this will reorder the characters in the block in a way that can be reversed by choosing a corresponding decryption key (n, a, b).

For example, if n = 5, a = 3, and b = 2, the string Hello, world! would be encrypted like this:

in:	H	e	l	l	o	,		w	o	r	l	d	!	\0	\0
i:	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
j:	2	0	3	1	4	2	0	3	1	4	2	0	3	1	4
out:	l	H	l	e	o	w	,	o		r	!	l	\0	d	\0