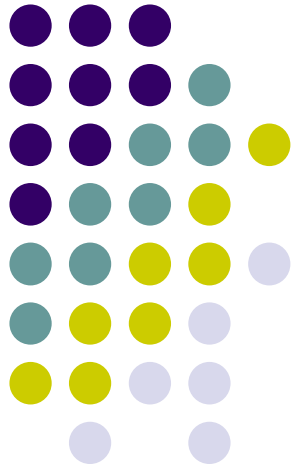
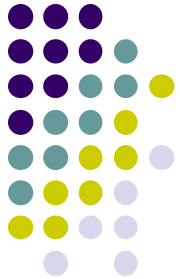
# PDS Lab Section 15

December 24, 2020

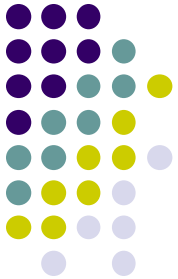


# Header to put in your file



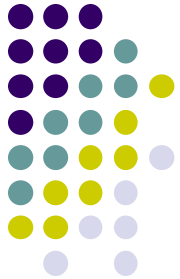
- Every program must start with a comment containing
  - Section No.
  - Roll No.
  - Name
  - Assignment No.
  - A one line description of the assignment
- Type the example header (replace with your name, roll no. assignment no. etc.) at the beginning of each of your C file, even before the #include <stdio.h>

# Example Header

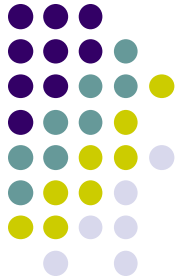


```
/*
*     Section 15
*     Roll No  :  20CS30010
*     Name     :  Your Name
*     Assignment No : 3
*     Description : Program to check points
*/
```

# Naming your program

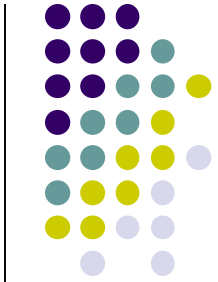


- Name your file with assgnX_Y.c, where X is the assignment number and Y is your roll no.
  - assgn5_20ME10010, assgn6_20CE30014,…..



# Assignments

# Assignment 5



Suppose we want to evaluate the series

$f(x) = 1 - x/1! + x^2/2! - x^3/3! + x^4/4! - \ldots$

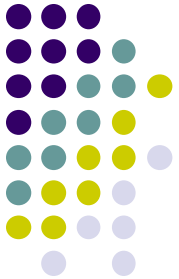
for $0 < x \leq 1$ (x is a floating point number)

upto a given number of terms.

Write a C program that will do the following:

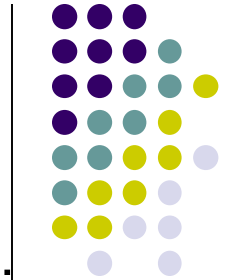
- Read in the values of x and an integer n
- If either the values entered for x does not satisfy $0 < x \leq 1$, or n is $\leq 0$, print a message asking the user to enter BOTH the values again, and read the values again (must use a while loop)
- Continue the above two steps until values entered satisfy both $0 < x \leq 1$ and n is positive
- Now compute the value of f(x) upto n terms and print it (must use a for loop)

You cannot use any mathematical function like pow etc. (50% penalty if you use it)

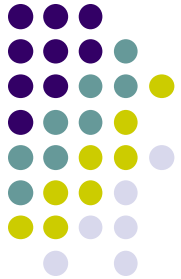


- Hint:
  - If you try to compute each term independently, the factorial will soon get very large even for small n
  - So think how you can generate the next term from the previous term (that you already computed)
    - Note that the first term is $1 = x^0/0!$
    - The second term is $(x/1)(x^0/0!) = x/1!$
    - The third term is $(x/2)(x/1!) = x^2/2!$
    - You should get a hint now
  - Decide on the sign of the term
    - Remember the term number ($1^{st}$, $2^{nd}$, $3^{rd}$,…) with a variable i (initialize it properly)
    - If i is even, multiply the term with -1 before adding

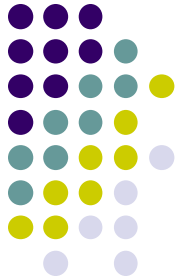
# Assignment 6



- In this assignment, you will read a line of alphanumeric characters (alphabets or digits), and replace them with the following code:
  - A lowercase alphabet should be replaced with the lowercase alphabet just before it (in cyclic order).
    - Example: ‘a’ should be replaced with ‘z’ (as ‘z’ is just before ‘a’ in the cyclic order), ‘b’ should be replaced with ‘a’, ‘c’ should be replaced with ‘b’, and so on
  - An uppercase alphabet should be replaced with the lowercase equivalent of the alphabet just before it (in cyclic order).
    - Example: ‘A’ should be replaced with ‘z’ (as ‘Z’ is just before ‘A’ in the cyclic order), ‘B’ should be replaced with ‘a’, ‘C’ should be replaced with ‘b’, and so on
    -

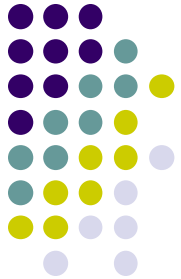


- A digit should be replaced with the digit just after it (in cyclic order)
  - So ‘0’ should be replaced by ‘1’, ‘1’ should be replaced by ‘2’, ‘2’ should be replaced by ‘3’, …, ‘9’ should be replaced by ‘0’
- Enter the line of characters from the keyboard with no space or anything in between, press Enter at the end once



- Approach:
  - Read in the characters one by one from the keyboard until you get the character ’\n’ (the special character corresponding to Enter keypress)
    - Use  a while loop
  - For each character read, take action as described and print the changed character
- Do NOT use an array even if you know it (will get 0 if array is sued)

# Teaser problems (not to be submitted)



- Read a positive integer. Find out if the digit sequence forms a palindrome, for example 34543 or 24455442 are such Palindrome numbers. Do NOT use any array.
- Consider two axis-parallel squares (sides parallel to x and y axis), with each square specified by its bottom left corner and length of side. Find out if the squares intersect, and if they do, the area of the overlap.
  - Do not do the brute force way of trying to find out all possible ways the squares can overlap and check them. There is a simple and elegant way to do this.
  - Will the same method work if the squares are not axis-parallel?