PDS-Lab (Section-17, Autumn 2020-21)
Class Test – 2 (1st Feb 2021, 2.30 – 5.00 PM) Marks = 100

Instructions:

a) *Create a directory named as <rollno>_ct2, where <rollno> is your roll number.*
b) *Give the name of the program as <p>.c where <p> implies the problems number, like 1.c 2.c 3.c etc. Store all the programs under this <rollno>_ct2 directory. Zip the entire directory <rollno>_ct2.*
c) *You should upload your zipped file <rollno>_ct2.zip to the Moodle course web page latest by 5**.00 PM** (without penalty). The **cutoff time** will be till **5.30 PM** with a penalty of **25%** on your secured marks (i.e., if you secured 80 marks, after penalty you will get 60 marks). Beyond 5.30 PM, the moodle system will not allow you to submit, as a result you will get zero.*
d) Do not use library functions
e) *Penalty for plagiarism/copying: You will be awarded **0 (zero)** in the Test if you are involved in plagiarism/copying and an **additional 10 marks** will be deducted from overall PDS Lab marks.*
f) *Keep your Camera **ON** (with **No virtual background**) throughout the Test. You should be always in front of the camera.*

---

1. Write a C program to compute the value of *cos(x)* using the expression shown below: **(20 Marks)**

$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots\ldots\ldots\ldots$$

In this problem you are required to use recursive calls of *cos* function, *factorial* function and *power* function. Through *main* () function provide the value of *x* in 'degrees' and number of terms to be considered for computation. Demonstrate the C function by calling through main program.

Note: In the above expression for computing the value of *cos(x),* using above expression *x* will take the value in radians only. Therefore, you need to convert degrees into radians.
180 degrees = ∏ = 22/7 = 3.141592 radians
cos(∏ + x) = - cos(x)
cos(2∏ + x) = cos(x)
cos(-x) = cos(x)
As cos function is the standard name of the trigonometric function, you may use my_cos() in your program in case if you get any warning during compilation.

2. Write a C program using structures to implement the Set operations: add (C = A + B), subtract (C = A – B), union (C = A U B) and intersection (C = A ∩ B) using the following functions : **(40 Marks)**

struct set read_s(int);
void print_s(struct set, int);
struct set add_s(struct set, struct set, int, int, int);*
struct set sub_s(struct set, struct set, int, int, int);*

struct set union_s(struct set, struct set, int, int, int);*
struct set intersection_s(struct set, struct set, int, int, int);*

Here, consider *struct set* consists of finite number of points in 2D plane. Define point in 2D plane as one more structure with 2 members represent x and y coordinates. Assume the maximum number of 2D points in sets A and B are 10. User will specify the number of points in A and B in the program.

Example:
set A = {(1,2), (4,3), (4,7), (12, 11), (3,5)}
set B = {(2,2), (3,3), (4,3), (1,2), (4,5)}
set C = A + B = {(1,2), (4,3), (4,7), (12, 11), (3,5), (2,2), (3,3), (4,3), (1,2), (4,5)}
set C = A - B = {(4,7), (12, 11), (3,5)}
set Union C = A U B = {(1,2), (4,3), (4,7), (12, 11), (3,5), (2,2), (3,3), (4,5)}
set Intersection C = A ∩ B = {(1,2), (4,3)}

3. Write a C program using pointers for performing find-and-replace function using the following function prototype *char* find_replace (char *str, char *find, char *replace)*. Input, find and replace strings need to be captured separately from the user using the following prototype function *void getstring(char *s).* Through main(), enter the strings by calling the function and demonstrate find-and-replace functionality. The *find_replace* function should return the output string to main(). Display input, find, replace and output strings in main(). **(40 Marks)**

Example:
Input : str[] = "xxfor xxxx for xx",
find[] = "xx",
replace[] = "yyy",
Output : yyyfor yyyyyy for yyy

Input : str[] = "xxfor xxx for xx",
find[] = "xx",
replace[] = "yyy",
Output : yyyfor yyyx for yyy