ignou
THE PEOPLE'S
UNIVERSITY

MASTER OF COMPUTER APPLICATIONS (MCAOL) JANUARY- 2025

PROGRAMME TITLE: Master of Computer Applications (MCAOL)

NAME: PAWAR VIKAS ANANT

ADDRESS: Matoshri niwas, near Vithal mandir, Sonwadi, 413801

ENROLMENT No.: 2500104692

COURSE CODE: MCS-201

COURSE TITLE: Programming in C and PYTHON

ASSIGNMENT CODE: PGDCA_NEW (I)/201/Assign /2025

CONTACT NUMBER: 8797879696

21/03/2025

ONLINE ASSIGNMENT SUBMISSION DATE

Pawar 21/03/2025

SIGNATURE

Course Code	:	**MCS-201**
Course Title	:	**Programming in C and PYTHON**
Assignment Number	:	**PGDCA_NEW(I)/201/Assign/2025**
Maximum Marks	:	**100**
Weightage	:	**30%**
Last Date of Submission	:	**30th April 2025 (for January Session)**

There are ten questions in this assignment which carries 80 marks. Each question carries 8 marks. Rest 20 marks are for viva-voce. Answer all the questions from both the sections i.e. Section A and Section B. You may use illustrations and diagrams to enhance the explanations. Include the screen layouts also along with your assignment responses. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

SECTION-A (C-Programming)

Q1: Write an algorithm, draw a flow chart and write its corresponding C program to convert a Binary decimal number to its equivalent Decimal number. **(8 Marks)**

Q2: Write an algorithm and use the concept of Structures to write the program in C, to generate Progress-Report of students of a class X of the school for all its 4 terms (the class is of 20 students). Assumptions can be made wherever necessary. **(8 Marks)**

Q3: Write a C program to generate the following pattern: **(8 Marks)**

```
*
* *
* * *
* * * *
* * * * *
```

Q4: Write a C program to perform the following operation on matrices D = A * (B + C), where A, B and C are matrices of (3 X 3) size and D is the resultant matrix. **(8 Marks)**

Q5: Use the concept of File Handling, to Write a program in C, to collect a list of N numbers in a file, and separate the even and odd numbers from the given list of N numbers, and put them in two separate files namely even_file and odd_file, respectively. **(8 Marks)**

SECTION-B (PYTHON-Programming)

Q6: Write Python code to perform the following: **(8 Marks)**

(i) Copy content of file first.txt to second.txt
(ii) Reading a file
(iii) Writing into a file
(iv) Appending into a file

Q7: Write an algorithm to find the slope of a line segment whose endpoint coordinates are $(x1, y1)$ and $(x2, y2)$. The algorithm gives output whether the slope is positive, negative or zero. Transform your algorithm into Python program. **(8 Marks)**

Note: Slope of line segment = $(y2 - y1)/(x2-x1)$.

Q8: Write a programme in Python to create a package named Volume and create 3 module in it named – Cube, Cuboid and Sphere each having a function to calculate Volume of Cube, Cuboid and Sphere respectively. Import the module in separate location and use the functions. Assumptions can be made wherever necessary. Support your programme with suitable comments to improve readability. **(8 Marks)**

Q9: Write a program in Python to perform following: **(8 Marks)**

- To find square root of numbers in a list using lambda function.
- To display first n lines from a file, where n is given by user.
- To display size of a file in bytes
- To display frequency of each word in a file.

Q10: What are Co-routines? How Co-routines differ from threads? How Co-routines support cooperative multi-tasking in python? Compare Subroutines and Co-routines. **(8 Marks)**

MASTER OF COMPUTER APPLICATIONS (MCAOL) JANUARY- 2025
SCHOOL OF COMPUTER AND INFORMATION SCIENCES
INDIRA GANDHI NATIONAL OPEN UNIVERSITY

Course Code: MCS-201

Course Title: Programming in C and PYTHON

Assignment Number: PGDCA_NEW(I)/201/Assign/2025

Section-A [C-programming]

Q1]

→ Defination:-

Binary to decimal conversion involves converting a number from base-2 (binary) to base-10 (decimal).

The decimal equivalent is obtained by multiplying each binary digit with 2 raised to corresponding position index.

Algorithm:-

1] Start

2] Take binary number as input

3] Initialize decimal value = 0, base = 1

4] Extract the last digit of the binary number

5] Multiply the digit with the base and add decimal value

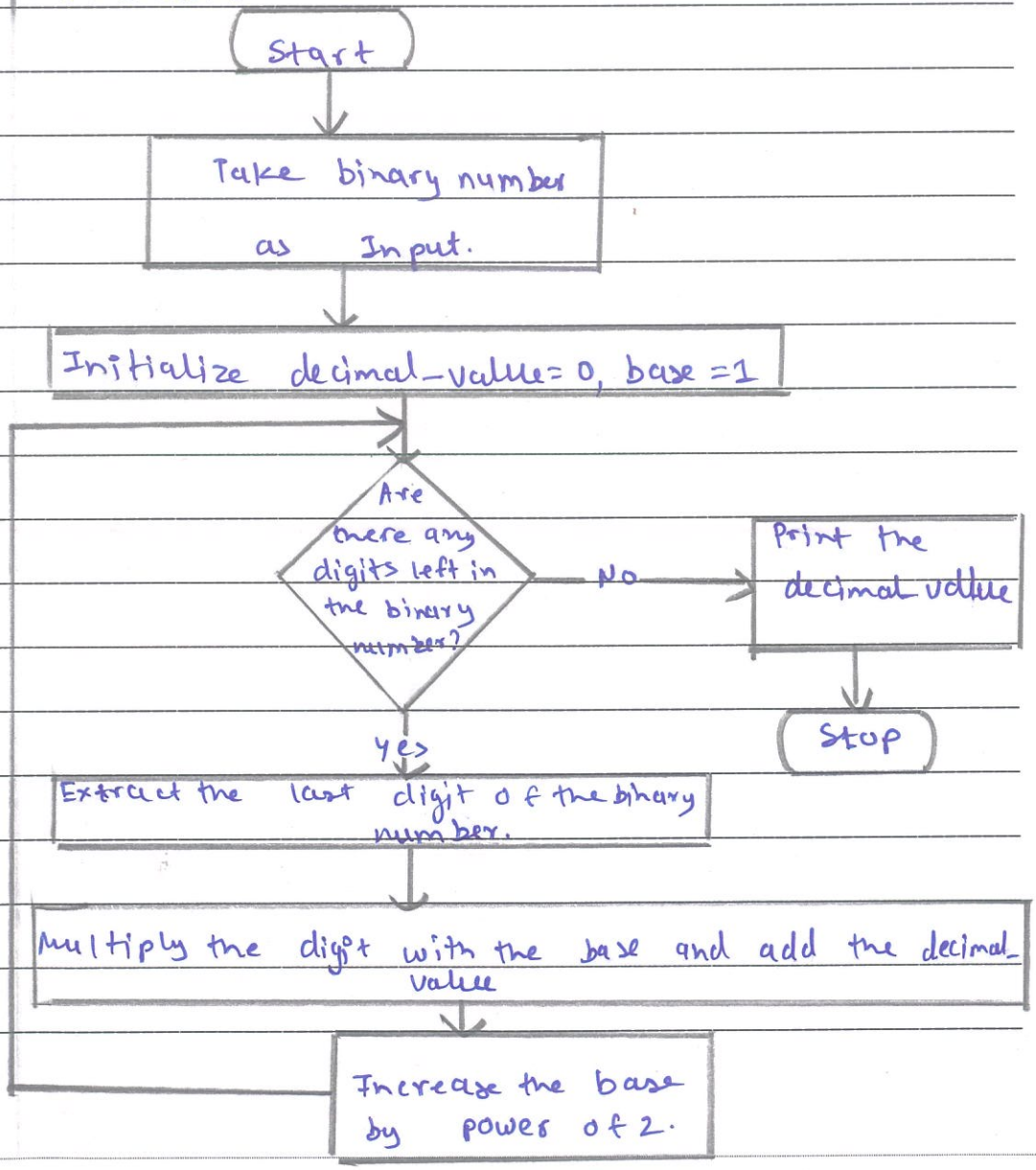
6] Increase the base by power of 2.

7) Repeat steps 4 to 6 until all digits are processed.

8) Print the decimal value

9) Stop.

Flow chart :-



Program:-

```
#include <stdio.h>
#include <math.h>

//function to convert binary to decimal
int binaryToDecimal(long long n){
    int decimal=0, base=1, remainder;

while(n!=0){
    remainder = n%10;
    decimal += remainder * base;
    base *=2;
    n /= 10;
}
return decimal
}
    int main(){
        long long binary;
// taking input
    printf("Enter a binary number:");
    scanf("%lld", &binary);
```

```
// converting and displaying result

printf("Decimal equivalent: %d\n",
        binaryToDecimal(binary));

return 0;

}
```

(Q 2)

→ Algorithm:

1) Define a structure Student to store students details:
   a) Roll number
   b) Name
   c) Marks for four terms
   d) Total and average marks
2) Create an array of students structures for 20 students
3) Input details of each student.
   ⇒ Accept Roll number, name and marks for four terms
4) Calculate the total and average marks for each student.
5) Display the progress report in tabular format.

Program:

```
#include <stdio.h>
#include <string.h>
```

```
#define STUDENTS 20
#define TERMS 4
// structure to store student details.
    struct student {
        int roll_no;
        char name[50];
        float marks[TERMS];
        float total;
        float average;
    };

// function to input students details
    void inputStudents (struct Student
     students[]) {
        For (int i=0; i<STUDENTS; i++) {
printf ("\nEnter details of student %d:\n",
        i+1);
printf ("Roll no: ");
Scanf ("%d", &students[i].roll_no);
printf ("Name:");
Scanf ("%[^/n]", students[i].name);
Student [i].total = 0;
```

```
for (int j=0, j<TERMS, j++) {
 scanf("%f", &students[i].marks[j]);
 students[i].total += students[i].marks[j];
}
students[i].average = students[i].total / TERMS;
}
}

//function to display progress report
    void displayReport(struct student students
    [])
printf("\n ..... ");
printf("\n %-5s %-20s %-10s %-10s %-10s
 %-10s %-10s %-10s", "Roll", "name", "Term 1",
 "Term 2", "Term 3" "Term 4" "Total" "Average");

For (int i=0 ; i< STUDENTS; i++) {
printf("%-5d %-20s, students[i].roll_no,
 students[i].name);
for (int j=0; j< TERMS; j++){
printf("%-10.2f", students[i].marks[j];
}
```

```
printf("%-10.2f %-10.2f\n", students[i].
total, students[i].average);
}
}
int main() {
    struct student students[STUDENTS];
inputStudents(students);

displayReport(students);

return 0;
}
```

Explanation:-

a] Structure defination: Stores roll no, name, marks for four terms, total and average marks.

b] Function Input:- Takes input for 20 students, including marks, calculates total and average marks.

c] Function display:- Prints progress Report in tabular format.

d] main function:- calls inputs students to take input. calls display report to display the report.

Q3]

⇒ The right-half pyramid is nothing but a right-angle triangle whose hypotenuse is in the right direction. We can print the half pyramid pattern using numbers, alphabets or any other characters like a star (*).

Program for * Pattern:-

```
#include <stdio.h>
int main() {
int rows=5;
//first loop for printing rows.
for (int i=0 ; i<rows; i++) {
//second loop for printing character in each row
for (int j=0; j<=i, j++) {
printf ("*");
}
printf ("\n");
}
return 0;
}
```

Q4]

→ Program :-

```
#include <stdio.h>
#define SIZE 3
//function to input a 3x3 matrix
void inputMatrix (int matrix [SIZE] [SIZE],
    char name) {
printf ("Enter elements of matrix %c (3x3) : \n",
name);
for (int i=0 ; i< SIZE ; i++) {
        for (int j = 0; j < SIZE ; j++) {
printf ("%c [%d] [%d] : ", name, i+1, j+1);
    Scanf ("%d", &matrix [i] [j]);
}
}
}
//function to display a 3x3 matrix
void displayMatrix (int matrix [SIZE] [SIZE],
 char name) {
printf ("\n matrix %c : \n", name );
    for (int i= 0, i< SIZE ; i++ ) {
    for (int j=0 ; j < SIZE ; j++) {
```

```
        printf("%5d", matrix[i][j]);
    }
    printf("\n");
}
}

// Function to add two 3x3 matrices (Temp = B+C)
void addMatrices(int B[SIZE][SIZE], int
C[SIZE][SIZE], Temp[SIZE][SIZE]) {
    for (int i=0; i<SIZE; i++) {
        for (int j=0; j<SIZE; j++) {
            Temp[i][j] = B[i][j] + C[i][j];
        }
    }
}

// function to multiply two 3x3 matrices (D = A * Temp)
void multiplyMatrices(int A[SIZE][SIZE],
int Temp[SIZE][SIZE], int D[SIZE][SIZE]) {
    for(int i=0; i<SIZE; i++) {
        for (int j=0; j<SIZE; j++) {
            D[i][j] = 0;

        for (int k=0; k<SIZE; k++) {
```

```
D[i][j] += A[i][k] * Temp[k][j];
}
}
}
}

int main(){
int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE],
Temp[SIZE][SIZE], D[SIZE][SIZE];

  inputMatrix(A, 'A');
  inputMatrix(B, 'B');
  inputMatrix(C, 'C');

  addMatrices(B, C, Temp);
  multiplyMatrices(A, Temp, D);

  displayMatrix(A, 'A');
  displayMatrix(B, 'B');
  displayMatrix(C, 'C');
  displayMatrix(Temp, 'Temp');
  displayMatrix(D, 'D');
  return 0;
}
```

Explanation:-

a] Input Matrices (A, B, C) using inputMatrix().

b) Compute Matrix sum (Temp = B+C) using addMatrices().

c) Perform Matrix Multiplication (D = A × Temp) using multiplyMatrices().

d) Display all matrices using displayMatrix().

I/P & O/P:-

Enter:-	Enter:-	Enter:-
matrix A (3×3)	matrix B (3×3)	matrix C (3×3)
1 2 3	9 8 7	1 2 3
4 5 6	6 5 4	4 5 6
7 8 9	3 2 1	7 8 9

Computed Matrices :

Matrix A			Matrix B		
1	2	3	9	8	7
4	5	6	6	5	4
7	8	9	3	2	1

Matrix C			Matrix T (B+C)		
1	2	3	10	10	10
4	5	6	10	10	10
7	8	9	10	10	10

Matrix D (A * (B+C)):		
60	60	60
150	150	150
240	240	240

Q5)

→ Program:

```
#include <stdio.h>
void separate_even_odd (const char * input_file,
const char * even_file , const char * odd_file)
{
  FILE * input , *even , * odd;
  int num;

  input = fopen(input_file , "r");
  even = fopen (even_file , "w");
  odd = fopen (odd_file , "w");

if (input == NULL || even == NULL || odd == NULL)
{
  printf ("Error opening file \n");
  return ;
}
while (fScanf %2 == 0)
    fprintf (even , "%d \n", num );
    else
    fprintf (odd , "%d \n", num );
}
```

```
    fclose(input);
    fclose(even);
    fclose(odd);
}
int main() {
    FILE * file;
    int n, num;
file = fopen("numbers.txt", "w");
    if (file == NULL) {
printf("Error creating file\n");
    return 1;
}
printf("Enter the number of Elements");
scanf("%d", &n);

printf("Enter %d numbers:\n", n);
    for (int i=0, i<n, i++) {
    scanf("%d", &num);
    fprintf(file, "%d\n", num);
}

fclose(file);
```

```
Separate_even_odd ("numbers.txt",
"even_file.txt", "odd_file.txt");

printf ("Even and odd has been
        separated\n");
  return 0;
  }
```

Explanation:-

a) It first asks user to enter N numbers and store them in numbers.txt.

b) The function separate_even_odd() reads from numbers.txt and writes even numbers to ~~odd_file~~ in even_file.txt and odd numbers to odd_file.txt.

c) file handling function such has fopen(), fclose(), fprintf() and fscanf() are used to manage file operations.

Section-B (PYTHON Programming)

Q6)

→ # (i) Copy content from first.txt to second.txt

```
def copy_file():
try:
with open ('first.txt', 'r') as source_file:
with open ('second.txt', 'w') as destination_file:
destination_file.write (source_file.read())
print ("file copied successfully")
except FileNotFoundError:
print ("file not found")
except Exception as e:
print (f"An error occurred: {e}")
```

(ii) Reading a file

```
def read_file():
try:
with open ('first.txt', 'r') as file:
content = file.read()
print ("file content:")
print (content)
```

```
except FileNotFoundError:
    print("file not found")
except Exception as e:
    print(f"An error occured: {e}")
```

#(iii) Writing into a file

```
def write_file():
try:
with open('second.txt', 'w') as file:
    file.write("This is new content\n")
    file.write("writing into a file")
    print("content written successfully")

except Exception as e:
print(f"An error occurred: {e}")
```

#(iv) Appending into a file

```
def append_file():
try:
with open('second.txt', 'a') as file:
    file.write("\nThis is appened
    content").
```

```
file.write("\n Adding more text")
print("content appended successfully")

except Exception as e:
print(f"An error occured: {e}")

copy_file()
read_file()
write_file()
append_file()
```

Explanation:

a) Copying file content: opens 'first.txt' in read mode ('r') and 'second.txt' im writ mode ('w'), then copies the content.

b) Reacling a file: opens 'first.txt' in read mode and prints the content.

c) Writing a file: Opens 'second.txt' in write mode('w'), which overwrites existing content with new text.

d] Appending a file: Opens 'second.txt' in append mode ('a'), which adds new content to the end of existing content.

Q7)

→ Algorithm:

1) Start
2) Input coordinates (x_1, y_1) and (x_2, y_2)
3) Calculate slope using formula: slope = $(y_2 - y_1) / (x_2 - x_1)$
4) If x_2 equal x_1, print "slope is undefined (vertical line)" and stop.
5) Else if slope > 0, print "slope is positive"
6) Else if slope < 0, print "Slope is negative"
7) Else print "slope is zero"
8) stop.

Program:

```
def find_slope(x1, y1, x2, y2):

    if x2 - x1 == 0:
    return "slope is undefined (vertical line)"
    else:
```

```
slope = (y2 - y1) /(x2 - x1)
if slop > 0:
return f"Slope is positive: {slope}"
elif slope < 0:
return f"Slope is negative: {slope}"
else :
return f" slope is zero: {slope}"

# Test the functions
x1, y1 = map (float, input ("Enter coordinates
of first point (x, y1):") . split())

x2, y2 = map (float, input ("Enter coordinates
of second point (x2 y2)"). split ())

result = find_slope (x1, y1, x2, y2)

print (result)
```

Explanation:.

a) The program takes input for two points and calculate the slope.

b) It handles the edges case of a vertical line (division by zero) and classfies the slope as positive, negative or zero.

(Q8)

→ Directory Structure:

```
Volume/
|- __init__.py (empty file to mark as package)
|- cube.py
|- cuboid.py
|- sphere.py
|- main.py (to test package)
```

Program :- a) cube.py

```
#cube
    def cube_volume(side):
        """Calculate volume of a cube"""
        return side**3
```

b) cuboid.py

```
#cuboid
    def cuboid_volume(length, width, height):
        """calculate volume of cuboid"""
        return length * width * height.
```

c) sphere.py

```
# sphere
import math
def sphere_volume(radius):
"""calculate volume of sphere"""
return (4/3) * math.pi * (radius ** 3)
```

d) main.py

```
# main
from volume.cube import cube_volume
from volume.cuboid import cuboid_volume
from volume.sphere import sphere_volume

# Test functions
print("Volume of cube (side = 3):", cube_volume
(3))
print("Volume of cuboid (l=2, w=3, h=4):",
cuboid_volume(2, 3, 4))
print("Volume of sphere (radius=5):",
sphere_volume(5))
```

Explanation:-

a) Each module defines a function to compute the volume of the respective shape.

b) The 'main.py' file imports and uses these functions, demonstrating modularity and package usage.

Q9]

→ Program:-

```
import math
import os
from collections import Counter

# Square root using Lambda
numbers = [4, 9, 16, 25]
sqrt_list = list(map(lambda x: math.sqrt(x),
numbers))
print("Square roots of", numbers, ":", sqrt_list)

# Display first n lines from a file
def display_n_lines(filename, n):
try:
with open(filename, 'r') as file:
lines = file.readlines()
for i in range(min(n, len(lines))):
print(lines[i], end = "")
except FileNotFoundError
print("File not found")
```

```
#Display file size in bytes
    def file_size(filename):
try:
    size = os.path.getsize(filename)
    return size
    except FileNotFoundError:
        return "file not found"

# word frequency in a file
    def word_frequency(filename):
try:
    with open(filename, 'r') as file:
        text = file.read().lower().split()
        freq = counter(text)
        return freq
        except FileNotFoundError
            return "file not found"

# Test the functions.
    filename = "sample.txt"
    n = int(input("Enter number of lines
        to display:"))
```

```
print(f"\n First {n} lines from {filename}:")
    display_n_lines(filename, n)
print(f"\n Size of {filename}: {file_size
    (filename)} bytes")

print(f"\n Word frequency in {filename}:")
    freq = word_frequency(filename)
    for word, count in freq.items():
        print(f"{word}: {count}")
```

Explanation:-

a] Uses 'lambda' with 'map()' to compute square roots

b] Reads and displays the first 'n' lines from a file.

c] Uses 'os.path.getsize()' to get the file size

d] Uses 'Counter' to count word frequencies in a file.

Q10]

→ What are Co-routines?

→ Co-routines are a generalization of sub-routines that allow multiple entry points and can suspend and resume execution at specific points.
In Python, they are implemented using 'asyn def' and 'await' keywords.
They are used for asynchronous programming, enabling non-blocking execution.

How Co-routines Differ from threads:-

→ Execution Model: Threads use preemptive multitasking (they yield control explicitly).

Resource Usage: Threads are heavier (each has its own stack), while co-routines are lightweight (single-threaded, managed by the event loop).

Concurrency: Threads run concurrency via

OS scheduling; Co-routines run in a single thread and switch manually.

How Co-routines support Cooperative Multitasking in Python:

→ Co-routines use an event loop (e.g, 'asyncio') to manage tasks. A co-routine yields control back to the event loop using 'await', allowing other tasks to run.
This cooperative nature avoids race conditions and reduces the need for locks, as tasks explicitly decide when to pause.

Comparison of subroutines and Co-routines

→ Execution flow:- Subroutines follow a strict call-return model (single entry/exit), while co-routines can pause and resume at multiple points.

State: Subroutines loses state after returning; co-routines retain state between yields.

Use cases: Subroutines are for sequential tasks; co-routines are for concurrent, asynchronous tasks. e.g I/O operations.

The End