1.Write a program in C to concatenate two given arrays of integers.

```
#include <stdio.h>

int main()
{
    int arr1[50], arr2[50], result[100];
    int i, j, k, n1, n2;

    printf("Enter the size of the first array: ");
    scanf("%d", &n1);
    printf("Enter elements of the first array:\n");
    for(i=0; i<n1; i++)
    {
        scanf("%d", &arr1[i]);
    }

    printf("Enter the size of the second array: ");
    scanf("%d", &n2);
    printf("Enter elements of the second array:\n");
    for(j=0; j<n2; j++)
    {
        scanf("%d", &arr2[j]);
    }

    k=0;
    for(i=0; i<n1; i++)
    {
        result[k] = arr1[i];
        k++;
    }
```

```
    for(j=0; j<n2; j++)
    {
        result[k] = arr2[j];
        k++;
    }

    printf("Concatenated array:\n");
    for(i=0; i<n1+n2; i++)
    {
        printf("%d ", result[i]);
    }

    return 0;
}
```

2.FIND TRANSPOSE OF A MATRIX

```
#include <stdio.h>

int main() {
    int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int transpose[3][3];

    // Finding transpose of matrix
    for(int i=0; i<3; i++) {
        for(int j=0; j<3; j++) {
            transpose[j][i] = matrix[i][j];
        }
    }

    // Printing original matrix
    printf("Original matrix:\n");
```

```
    for(int i=0; i<3; i++) {
        for(int j=0; j<3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }

    // Printing transpose matrix
    printf("\nTranspose of matrix:\n");
    for(int i=0; i<3; i++) {
        for(int j=0; j<3; j++) {
            printf("%d ", transpose[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

3.FIND INVERSE OF A MATRIX

```
#include <stdio.h>

#define N 3 // size of the matrix

// Function to print a matrix
void printMatrix(float matrix[N][N+1]) {
    for(int i=0; i<N; i++) {
        for(int j=0; j<=N; j++) {
            printf("%.2f ", matrix[i][j]);
        }
        printf("\n");
    }
```

```
}

// Function to swap two rows of a matrix
void swapRows(float matrix[N][N+1], int i, int j) {
    for(int k=0; k<=N; k++) {
        float temp = matrix[i][k];
        matrix[i][k] = matrix[j][k];
        matrix[j][k] = temp;
    }
}

int main() {
    float matrix[N][N+1] = {{3, 2, -4, 3},
                            {2, 3, 3, 15},
                            {5, -3, 1, 14}};

    // Augment the matrix with identity matrix
    for(int i=0; i<N; i++) {
        for(int j=N; j<2*N; j++) {
            if(i == j-N) {
                matrix[i][j] = 1;
            }
            else {
                matrix[i][j] = 0;
            }
        }
    }

    // Perform row operations to get the reduced row-echelon form of the matrix
    for(int i=0; i<N; i++) {
        // Check if the diagonal element is zero, if so swap rows to get a non-zero element
        if(matrix[i][i] == 0) {
```

```
            int k = i+1;
            while(k<N && matrix[k][i] == 0) {
                k++;
            }
            if(k == N) {
                printf("Matrix is not invertible.\n");
                return 0;
            }
            swapRows(matrix, i, k);
        }
        // Divide the row by the diagonal element to make it 1
        float divisor = matrix[i][i];
        for(int j=i; j<2*N; j++) {
            matrix[i][j] /= divisor;
        }
        // Perform row operations to make all other elements in the column zero
        for(int k=0; k<N; k++) {
            if(k != i) {
                float factor = matrix[k][i];
                for(int j=i; j<2*N; j++) {
                    matrix[k][j] -= factor * matrix[i][j];
                }
            }
        }
    }

    // Print the inverse matrix
    printf("Inverse of the matrix:\n");
    for(int i=0; i<N; i++) {
        for(int j=N; j<2*N; j++) {
            printf("%.2f ", matrix[i][j]);
        }
```

```
        printf("\n");
    }

    return 0;
}
```

4.TO FIND THE MULTIPLICATION OF MATRIX

```
#include <stdio.h>

#define ROWS1 3
#define COLS1 2
#define ROWS2 2
#define COLS2 3

// Function to print a matrix
void printMatrix(int matrix[][COLS2], int rows, int cols) {
    for(int i=0; i<rows; i++) {
        for(int j=0; j<cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int matrix1[ROWS1][COLS1] = {{1, 2}, {3, 4}, {5, 6}};
    int matrix2[ROWS2][COLS2] = {{7, 8, 9}, {10, 11, 12}};
    int result[ROWS1][COLS2];

    // Multiplying matrix1 and matrix2 to get result
    for(int i=0; i<ROWS1; i++) {
        for(int j=0; j<COLS2; j++) {
```

```
            int sum = 0;
            for(int k=0; k<ROWS2; k++) {
                sum += matrix1[i][k] * matrix2[k][j];
            }
            result[i][j] = sum;
        }
    }

    // Printing the matrices and the result
    printf("Matrix1:\n");
    printMatrix(matrix1, ROWS1, COLS1);

    printf("\nMatrix2:\n");
    printMatrix(matrix2, ROWS2, COLS2);

    printf("\nResult:\n");
    printMatrix(result, ROWS1, COLS2);

    return 0;
}
```

*5.The factorial of an integer n, written n!, is the product of all the integers from 1 to n inclusive. The factorial quickly becomes very large; 13! is too large to store as an integer on most computers, and 35! is too large for a floating-point variable. Your task is to find the rightmost non-zero digit of n!. (1<<= n <= 100) For example, 5!= 1*2*3*4*5= 120, so the rightmost non-*

*zero digit of 5! is 2. Also, 7! = 1*2*3*4*5*6*7= 5040, so the rightmost non-zero digit of 7! is 4.*

```
#include <stdio.h>

int main() {
    int n, i, num = 1;
    printf("Enter a number: ");
    scanf("%d", &n);

    for (i = 2; i <= n; i++) {
        num *= i;
        while (num % 10 == 0) {
            num /= 10;
        }
        num = num % 100000;  // to avoid overflow
    }

    printf("The rightmost non-zero digit of %d! is %d.\n", n, num % 10);
    return 0;
```

6..Modify AI(1) to have a function CheckOddEven(num) that checks if the num is odd or even; sets a flag accordingly and return it. Use this function to find the sum of even and odd numbers in a given input of N numbers.

```
#include <stdio.h>

int CheckOddEven(int num) {
    int flag;
```

```
    if (num % 2 == 0) {
        flag = 0; // even
    } else {
        flag = 1; // odd
    }
    return flag;
}

int main() {
    int N, i, num, flag, sum_even = 0, sum_odd = 0;
    printf("Enter the number of elements: ");
    scanf("%d", &N);

    for (i = 0; i < N; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &num);
        flag = CheckOddEven(num);
        if (flag == 0) {
            sum_even += num;
        } else {
            sum_odd += num;
        }
    }

    printf("Sum of even numbers: %d\n", sum_even);
    printf("Sum of odd numbers: %d\n", sum_odd);
    return 0;
}
```

7.Write a C function ReverseNum(num) that takes integer num and reverses its digits. Let num be passed by reference.

Example:

Input: 453275

Output: 572354

```
#include <stdio.h>

void ReverseNum(int* num) {
    int rev = 0;
    while (*num != 0) {
        rev = rev * 10 + *num % 10;
        *num /= 10;
    }
    *num = rev;
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    ReverseNum(&num);
    printf("Reversed number: %d\n", num);
    return 0;
}
```

8.Write a function power(X,N) that will allow a floating-point number to be raised to an integer power. Y=XN

In other words, evaluate the formula where y and x are floating-point variables and n is an integer variable. Write a C program that will read in numerical values for x and n, evaluate the formula using power(X,N), and then display the calculated result. Test the program using the following data:

X

3

12

1.5

3

1222333

-5

1.5

10

-3

3

1.5

-5

-3

7

-5

0.2

3

0.2

5

-3

X

0.2

-5

```
#include <stdio.h>

float power(float X, int N) {
    float Y = 1.0;
    int i;
    if (N >= 0) {
        for (i = 0; i < N; i++) {
            Y *= X;
        }
```

```
    } else {
        for (i = 0; i > N; i--) {
            Y /= X;
        }
    }
    return Y;
}

int main() {
    float X;
    int N;
    printf("Enter the value of X: ");
    scanf("%f", &X);
    printf("Enter the value of N: ");
    scanf("%d", &N);
    float Y = power(X, N);
    printf("%f raised to the power of %d is %f\n", X, N, Y);
    return 0;
}
```

9.String reverse, string compare, string concatenate, finding substring using pointers

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

// function to reverse a string
void reverse(char* str) {
    int len = strlen(str);
    int i;
    for (i = 0; i < len / 2; i++) {
        char temp = str[i];
        str[i] = str[len - i - 1];
```

```
        str[len - i - 1] = temp;
    }
}

// function to compare two strings
int compare(char* str1, char* str2) {
    int len1 = strlen(str1);
    int len2 = strlen(str2);
    int len = len1 < len2 ? len1 : len2;
    int i;
    for (i = 0; i < len; i++) {
        if (str1[i] != str2[i]) {
            return str1[i] - str2[i];
        }
    }
    return len1 - len2;
}

// function to concatenate two strings
void concatenate(char* dest, char* src) {
    int len1 = strlen(dest);
    int len2 = strlen(src);
    int i;
    for (i = 0; i < len2; i++) {
        dest[len1 + i] = src[i];
    }
    dest[len1 + len2] = '\0';
}

// function to find a substring in a string
char* find_substring(char* str, char* sub) {
    int len1 = strlen(str);
```

```
    int len2 = strlen(sub);
    int i, j;
    for (i = 0; i <= len1 - len2; i++) {
        for (j = 0; j < len2; j++) {
            if (str[i + j] != sub[j]) {
                break;
            }
        }
        if (j == len2) {
            return &str[i];
        }
    }
    return NULL;
}

int main() {
    char str1[100], str2[100];
    printf("Enter a string: ");
    scanf("%s", str1);
    printf("Enter another string: ");
    scanf("%s", str2);

    // reverse str1
    reverse(str1);
    printf("Reversed string 1: %s\n", str1);

    // compare str1 and str2
    int cmp = compare(str1, str2);
    if (cmp == 0) {
        printf("String 1 is equal to string 2\n");
    } else if (cmp < 0) {
        printf("String 1 is less than string 2\n");
```

```
    } else {
        printf("String 1 is greater than string 2\n");
    }

    // concatenate str1 and str2
    concatenate(str1, str2);
    printf("Concatenated string: %s\n", str1);

    // find substring in str1
    char sub[100];
    printf("Enter a substring to find in string 1: ");
    scanf("%s", sub);
    char* ptr = find_substring(str1, sub);
    if (ptr) {
        printf("Substring found at position %ld\n", ptr - str1);
    } else {
        printf("Substring not found\n");
    }

    return 0;
}
```

10.Read and print text. Also count the number of characters, words and lines in the text.

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

int main() {
    char ch, prev;
    int char_count = 0, word_count = 0, line_count = 0;

    printf("Enter some text: \n");
```

```
    while ((ch = getchar()) != EOF) {
        if (ch == '\n') {
            line_count++;
        }
        if (isspace(ch) && !isspace(prev)) {
            word_count++;
        }
        if (!isspace(ch)) {
            char_count++;
        }
        prev = ch;
        putchar(ch);
    }

    if (char_count > 0) {
        word_count++;
        line_count++;
    }

    printf("\n\nNumber of characters: %d\n", char_count);
    printf("Number of words: %d\n", word_count);
    printf("Number of lines: %d\n", line_count);

    return 0;
}
```

11.Finding the mean of n numbers using arrays

```
#include <stdio.h>

int main() {
    int n, i;
```

```
    float sum = 0, mean;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    float nums[n];
    printf("Enter %d numbers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%f", &nums[i]);
        sum += nums[i];
    }
    mean = sum / n;
    printf("Mean = %.2f\n", mean);
    return 0;
}
```

12.Finding whether a number is prime or composite till -1 is entered

```
#include <stdio.h>

int main() {
    int num, i, flag;

    while (1) {
        printf("Enter a number (-1 to exit): ");
        scanf("%d", &num);
        if (num == -1) {
            break;
        }
        flag = 0;
        for (i = 2; i <= num / 2; ++i) {
            if (num % i == 0) {
                flag = 1;
                break;
```

```
            }
        }
        if (num == 1) {
            printf("%d is neither prime nor composite.\n", num);
        }
        else {
            if (flag == 0) {
                printf("%d is a prime number.\n", num);
            }
            else {
                printf("%d is a composite number.\n", num);
            }
        }
    }
    return 0;
}
```

13.Display the sum and average of numbers from m to n

```
#include <stdio.h>

int main() {
    int m, n, i, sum = 0, count = 0;
    float avg;
    printf("Enter the value of m: ");
    scanf("%d", &m);
    printf("Enter the value of n: ");
    scanf("%d", &n);
    for (i = m; i <= n; i++) {
        sum += i;
        count++;
    }
```

```
    avg = (float)sum / count;
    printf("The sum of numbers from %d to %d is %d\n", m, n, sum);
    printf("The average of numbers from %d to %d is %.2f\n", m, n, avg);
    return 0;
}
```

14.To convert a floating point n umber to integer

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

int main() {
    float x = 3.14159;
    int y;

    // Using typecasting
    y = (int)x;
    printf("Using typecasting: x=%f, y=%d\n", x, y);

    // Using floor function
    y = floor(x);
    printf("Using floor function: x=%f, y=%d\n", x, y);

    // Using ceil function
    y = ceil(x);
    printf("Using ceil function: x=%f, y=%d\n", x, y);

    return 0;
}
```

15.Find the size of various data types and different types of pointers

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

```
int main() {
    // Size of basic data types
    printf("Size of char: %ld bytes\n", sizeof(char));
    printf("Size of int: %ld bytes\n", sizeof(int));
    printf("Size of float: %ld bytes\n", sizeof(float));
    printf("Size of double: %ld bytes\n", sizeof(double));
    printf("Size of long: %ld bytes\n", sizeof(long));
    printf("Size of long long: %ld bytes\n", sizeof(long long));

    // Size of pointers
    printf("Size of int pointer: %ld bytes\n", sizeof(int *));
    printf("Size of char pointer: %ld bytes\n", sizeof(char *));
    printf("Size of float pointer: %ld bytes\n", sizeof(float *));
    printf("Size of void pointer: %ld bytes\n", sizeof(void *));

    // Size of function pointers
    printf("Size of function pointer: %ld bytes\n", sizeof(void (*)(void)));

    // Size of struct
    struct example {
        char c;
        int i;
        float f;
        double d;
    };
    printf("Size of struct example: %ld bytes\n", sizeof(struct example));

    return 0;
}
```

16.Finding GPA and Rank of a class having 5 subjects

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

```
#define MAX_STUDENTS 100

struct student {
    char name[50];
    float marks[5];
    float gpa;
    int rank;
};

int main() {
    struct student students[MAX_STUDENTS];
    int n;

    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Input data for each student
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for student %d:\n", i+1);
        printf("Name: ");
        scanf("%s", students[i].name);

        float total_marks = 0.0;

        printf("Enter marks for 5 subjects:\n");
        for (int j = 0; j < 5; j++) {
            printf("Subject %d: ", j+1);
            scanf("%f", &students[i].marks[j]);
            total_marks += students[i].marks[j];
        }
```

```
        students[i].gpa = total_marks / 5.0;
    }

    // Calculate rank for each student
    for (int i = 0; i < n; i++) {
        int rank = 1;

        for (int j = 0; j < n; j++) {
            if (students[j].gpa > students[i].gpa) {
                rank++;
            }
        }

        students[i].rank = rank;
    }

    // Display GPA and rank for each student
    printf("\nGPA and rank of students:\n");
    printf("%-20s %-10s %-10s\n", "Name", "GPA", "Rank");
    for (int i = 0; i < n; i++) {
        printf("%-20s %-10.2f %-10d\n", students[i].name, students[i].gpa, students[i].rank);
    }

    return 0;
}
```

17. Sorting list of numbers using arrays

```
#include <stdio.h>

#define MAX_SIZE 100

void sort(int arr[], int size);
```

```
int main() {
    int arr[MAX_SIZE], size, i;

    printf("Enter the number of elements: ");
    scanf("%d", &size);

    printf("Enter the elements: ");
    for (i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    sort(arr, size);

    printf("Sorted elements: ");
    for (i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}

void sort(int arr[], int size) {
    int i, j, temp;

    for (i = 0; i < size - 1; i++) {
        for (j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
```

```
        }
    }
}
```

18. Find whether a given matrix is lower triangular or upper triangular matrix

```
#include <stdio.h>

#define MAX_SIZE 10

int main() {
    int matrix[MAX_SIZE][MAX_SIZE], rows, cols, i, j;
    int is_upper_triangular = 1, is_lower_triangular = 1;

    printf("Enter the number of rows and columns of the matrix: ");
    scanf("%d %d", &rows, &cols);

    printf("Enter the elements of the matrix: \n");
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            if (i > j && matrix[i][j] != 0) {
                is_upper_triangular = 0;
            }
            if (i < j && matrix[i][j] != 0) {
                is_lower_triangular = 0;
            }
```

```
        }
    }

    if (is_upper_triangular == 1) {
        printf("The matrix is upper triangular.\n");
    } else if (is_lower_triangular == 1) {
        printf("The matrix is lower triangular.\n");
    } else {
        printf("The matrix is neither upper triangular nor lower triangular.\n");
    }

    return 0;
}
```

19. Read and print an array of numbers, then find out the smallest number and print its postion. Use read_array(), print_array(), find_small() functions.

```
#include <stdio.h>

#define MAX_SIZE 100

void read_array(int arr[], int size);
void print_array(int arr[], int size);
int find_small(int arr[], int size, int *pos);

int main() {
    int arr[MAX_SIZE], size, pos, small;

    printf("Enter the size of array: ");
    scanf("%d", &size);

    printf("Enter the elements of array:\n");
```

```
    read_array(arr, size);

    printf("The elements of array are:\n");
    print_array(arr, size);

    small = find_small(arr, size, &pos);

    printf("The smallest number is %d at position %d.\n", small, pos);

    return 0;
}

void read_array(int arr[], int size) {
    int i;

    for (i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
}

void print_array(int arr[], int size) {
    int i;

    for (i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    printf("\n");
}

int find_small(int arr[], int size, int *pos) {
    int i, small;
```

```
    small = arr[0];
    *pos = 0;

    for (i = 1; i < size; i++) {
        if (arr[i] < small) {
            small = arr[i];
            *pos = i;
        }
    }

    return small;
}
```

20. Read a text till the you enter END. It can be multiple lines

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

int main() {
    char text[1000] = "", line[100];
    printf("Enter text (type END to stop):\n");

    while (strcmp(line, "END") != 0) {
        fgets(line, 100, stdin);
        strcat(text, line);
    }

    printf("The text you entered is:\n%s", text);
    return 0;
}
```

21.Create a user defined type enum days of week anddisplayall.

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

int main() {
    enum days {sun = 1, mon, tue, wed, thu, fri, sat};
    int no = 1;
    enum days d1;

    printf("Enter the number of the day you want to see\n");
    printf("To exit, enter 0\n");

    while(no) {
        scanf("%d", &no);
        d1 = no;
        if(d1 == sun)
            printf("The day is: Sunday\n");
        else if(d1 == mon)
            printf("The day is: Monday\n");
        else if(d1 == tue)
            printf("The day is: Tuesday\n");
        else if(d1 == wed)
            printf("The day is: Wednesday\n");
        else if(d1 == thu)
            printf("The day is: Thursday\n");
        else if(d1 == fri)
            printf("The day is: Friday\n");
        else if(d1 == sat)
            printf("The day is: Saturday\n");
        else if(d1 != 0) {
```

```
            printf("Invalid input\n");
            break;
        }
        else {
            printf("Exiting...\n");
            break;
        }
    }
    return 0;
}
```