1. Perform addition, substraction and multiplication operations on two matrics.

Code:
```
#include <stdio.h>
#define MAX_ROWS 10
#define MAX_COLS 10
void read_matrix (int matrix [MAX_ROWS][MAX_COLS], int rows, int cols) {
  printf ("Enter elements of the Matrix :\n");
  for (int i=0; i<rows; i++) {
    for (int j=0; j<cols; j++) {
      printf ("Enter element at [%d][%d] : ", i, j);
      scanf ("%d", &matrix [i][j]);
    }
  }
}
void print_matrix (int matrix [MAX_ROWS] [MAX_COLS], int rows, int cols) {
  for (int i=0; i<rows; i++) {
    for (int j=0; j<cols; j++) {
      printf ("%d \t", matrix [i][j]);
    }
    printf ("\n");
  }
}
void matrix_addition (int A [MAX_ROWS] [MAX_COLS], int B [MAX_ROWS]
  [MAX COLS], int C [MAX_ROWS] [MAX COLS], int rows, int colse) {
  for (int i=0; i<rows; i++) {
    for (int j=0; j<cols; j++) {
      C [i][j] = A [i][j] + B [i][j];
    }
  }
}
void matrix_substraction (int A [MAX_ROWS] [MAX_COLS], int B [MAX_ROWS]
  [MAX_COLS], int C [MAX_ROWS] [MAX_COLS], int rows, int cols) {
  for (int i=0; i<rows; i++) {
    for (int j=0; j<cols; j++) {
      C [i][j] = A [i][j] - B [i][j];
    }
  }
}
void matrix_multiplication (int A [MAX_ROWS][MAX_COLS], int B [MAX_ROWS]
  [MAX_COLS], int r1, int c1, int r2, int c2) {
  if (c1 != r2) {
    printf ("Error: Matrix multiplication not possible. Columns of first
    matrix must equal rows of second matrix. \n");
    return;
  }
  for (int i=0; i<r1; i++) {
    for (int j=0; j<c2; j++) {
      C [i][j] = 0;
      for (int k=0; k<c1; k++) {
        C [i][j] += A [i][k] * B [k][j];
      }
    }
  }
}
```

```
int main() {
    int A[MAX_ROWS][MAX_COLS], B[MAX_ROWS][MAX_COLS], C[MAX_ROWS][MAX_COLS];
    int r1, c1, r2, c2;
    printf("Enter number of rows and columns for matrix A: ");
    scanf("%d %d", &r1, &c1);
    read_matrix(A, r1, c1);
    printf("Enter number of rows and colums for matrix B: ");
    scanf("%d %d", &r2, c2);
    if (r1==r2 && c1==c2) {
        printf("\n Matrix Addition:\n");
        matrix_addition(A, B, C, r1, c1);
        printf_matrix(C, r1, c1);
        printf("\n Matrix Substraction:\n");
        matrix_substraction(A, B, C, r1, c1);
        printf_matrix(C, r1, c1);
    } else {
        printf("\n Addition and Substraction not possible. Matrix must have the
        same dimensions.\n");
    }
    printf("\n Matrix Multiplication:\n");
    matrix_multiplication(A, B, C, r1, c1, r2, c2);
    if (c1==c2) {
        print_matrix(C, r1, c2);
    }
    return 0;
}
```

Output:

```
Enter number of rows and columns for matrix A: 1 1
Enter elements of the matrix:
Enter element at [0][0]: 0
Enter number of rows and columns for matrix B: 1 1
Enter elements of the matrix:
Enter element at [0][0]: 9
Matrix addition: 9
Matrix Substraction: -9
Matrix Multiplication: 0.
```

2. Sort all the elements of a 4×4 matrix and store the result in a single dimension array.

Code :

```
#include <stdio.h>
void sortArray (int arr[] ,int n) {
   int i,j, temp ;
   for (i=0;i<n-1; i++) {
      for (j=0 ;j<n-1 ;j++) {
         if (arr[j] > arr[j+1]) {
            temp = arr[j] ;
            arr[j] =arr[j+1] ;
            arr[j+1] = temp ;
         }
      }
   }
}
int main() {
   int matrix[4][4] = {
      {16,12,1,5},
      {10,8,2,6},
      {14,9,3,7},
      {13,11,4,15}
   };
   int singleDimensionArray [16];
   int k=0;
   int i,j;
   for (i=0;i<4;i++) {
      for (j=0;j<4;j++) {
         singleDimensionArray[k] = matrix[i][j];
         k++;
      }
   }
   sortArray (singleDimensionArray , 16);
   printf ("sorted elements in a single-dimension array :\n");
   for (i=0; i<16; i++) {
      printf ("%d", singleDimensionArray[i]);
   }
   printf ("\n");
   return 0;
}
```

Output : Sorted elements in a single-dimension array :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

3. Print the largest and smallest numbers from a 3x3 matrix using pointer.

Code:
```
#include <stdio.h>
int main() {
    int matrix[3][3] = {
     {10,2,30} , {4,55,6} , {7,8,9}
    };
    int *ptr = &matrix[0][0];
    int largest = *ptr;
    int smallest = *ptr;
    for (int i=0; i<9; i++){
        if (*(ptr+i) > largest) {
            largest = *(ptr+i)
        }
        if (*(ptr+i) < smallest) {
            smallest = *(ptr+i);
        }
    }
    printf("The largest number in the matrix is: %d\n", largest);
    printf("The smallest number in the matrix is: %d\n", smallest);
    return 0;
}
```

Output: The largest number in the matrix is: 55

The smallest number in the matrix is: 2

4. Accept and print later on three book names ~~of individuals~~ using array of pointers.

Code :
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char * books[3];
    int i;
    for (i=0; i<3; i++) {
        books[i] = (char *) malloc (50 * sizeof (char));
        if (books[i] == NULL) {
            printf ("Memory allocation failed !\n");
            return 1;
        }
        printf ("Enter book name %d : ", i+1);
        scanf ("%s", books[i]);
    }
    printf ("\n--- Book names entered ---\n");
    for (i=0; i<3; i++) {
        printf ("%s\n", books[i]);
    }
    for (i=0; i<3; i++) {
        free (books[i]);
    }
    return 0;
}
```

Output : Enter book name 1 : red
Enter book name 2 : blue
Enter book name 3 : yellow
--- Book names entered ---
red
blue
yellow

5. Write a program that takes a set of names of individuals and abbreviates the first, middle and other names except the last name by their first later.

Code :
```
#include <stdio.h>
#include <string.h>
void abbreviateName (char * name) {
    int len = strlen (name);
    if (len > 0 && name [len - 1] == '\n') {
        name [len - 1] = '\0';
        len --;
    }
    if (len == 0) {
        return;
    }
    char * last_name_ptr = ~~NULL;~~ strrchr (name, ' ');
    if (last_name_ptr == NULL) {
        printf ("%s\n", name);
        return;
    }
```

```
    printf("%c.", name[0]);
    char* current_ptr = name + 1;
    while (current_ptr < last_name_ptr) {
        if (*current_ptr == ' ') {
            printf("%c.", *(current_ptr + 1));
        }
        current_ptr++;
    }
    printf("%s\n", last_name_ptr + 1);
}
int main() {
    char name[100];
    printf("Enter a full name: ");
    fgets(name, 100, stdin);
    abbreviateName(name);
    return 0;
}
```

Output: Enter a full name: daiya darshan dwarkadas

d.d. dwarkadas

1. Write a function power(a,b) to calculate the value of a raised to b.

Code:
```
# include <stdio.h>
int power (int a, int b) {
   int result = 1;
   for (int i = 0; i<b; i++) {
      result = result * a;
   }
   return result;
}
int main() {
   int base, exponent, result;
   printf ("Enter the base number:");
   scanf ("%d", &base);
   printf ("Enter the exponent:");
   scanf ("%d", &exponent);
   result = power (base, exponent);
   printf ("%d raised to the power of %d is %d\n", base, exponent,
           result);
   return 0;
}
```

Output: Enter the base number: 7 18
Enter the exponent: 7 raised to the power of 18 is -1777531471.

2. Any year is entered through the keyboard. Write a function to determine whether the year is a leap year or not.

Code:
```
# include <stdio.h>
int isLeapYear (int year) {
   if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
      return 1;
   } else {
      return 0;
   }
}
int main() {
   int year;
   printf ("Enter a year: ");
   scanf ("%d", &year);
   if (isLeapYear(year)) {
      printf ("%d is a leap year.\n", year);
   } else {
      printf ("%d is not a leap year.\n", year);
   }
   return 0;
}
```

Output: Enter a year: 2025
2025 is not a leap year.

3. Write a recursive function to calculate factorial of a number.

Code:
```
#include <stdio.h>
    long long int factorial (int n) {
      if (n==0 || n==1) {
        return 1;
      }
      else {
        return n * factorial (n-1);
      }
    }
    int main() {
      int number;
      printf ("Enter a non-negative integer: ");
      scanf ("%d", &number);
      if (number < 0) {
        printf ("Factorial is not defind for negative numbers.\n");
      } else {
        printf ("Factorial of %d is %lld.\n", number, factorial
                (number));
      }
      return 0;
    }
```

Output: Enter a non-negative integer: 7
Factorial of 7 is 5040.

4. Write a function to swap two integers using call by value. Show that the original values are not changed.

Code:
```
#include <stdio.h>
    void swap (int a, int b) {
        int temp;
        temp = a;
        a = b;
        b = temp;
        printf ("Inside swap function: a = %d, b = %d \n", a, b);
    }
    int main () {
        int x = 10, y = 20;
        printf ("Before swap function call : x = %d, y = %d \n", x, y);
        swap (x, y);
        printf ("After swap function call : x = %d, y = %d \n", x, y);
        return 0;
    }
```

output: Before swap function call : x = 10, y = 20

Inside swap function : a = 20, b = 10

After swap function call : x = 10, y = 20

5. Write a program that uses a function to update both the maximum and minimum values in an array using call by reference.

Code:
```
#include <stdio.h>
void findMinMax(int arr[], int size, int* min, int* max);
int main() {
    int numbers[] = {10, 5, 26, 8, 30, 15};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int min_val, max_val;
    findMinMax(numbers, size, &min_val, &max_val);
    printf("Minimum value : %d\n", min_val);
    printf("Maximum value : %d\n", max_val);
    return 0;
}
void findMinMax(int arr[], int size, int* min, int* max) {
    if (size <= 0) {
        return;
    }
    *min = arr[0];
    *max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < *min) {
            *min = arr[i];
        }
        if (arr[i] > *max) {
            *max = arr[i];
        }
    }
}
```

Output:
```
Minimum value: 5
Maximum value: 30
```

6. Write a program to implement a calculator using separate functions for add, subtract, multiply, and divide.

Code :
```
#include <stdio.h>
double add (double num1, double num2) {
    return num1 + num2;
}
double subtract (double num1, double num2) {
    return num1 - num2;
}
double multiply (double num1, double num2) {
    return num1 * num2;
}
double divide (double num1, double num2) {
  if (num2 != 0) {
     return num1 / num2;
  } else {
     printf ("Error: Division by zero is not allowed. \n");
     return 0;
  }
}
int main() {
   char operator;
   double num1, num2, result;
   printf ("Enter an operator (+, -, *, /): ");
   scanf ("%c", &operator);
   printf ("Enter two numbers: ");
   scanf ("%lf %lf", &num1, &num2);
   switch (operator) {
      case '+':
         result = add (num1, num2);
         printf ("%.2lf + %.2lf = %.2lf \n", num1, num2, result);
         break;
      case '-':
         result = subtract (num1, num2);
         printf ("%.2lf - %.2lf = %.2lf \n", num1, num2, result);
         break;
      case '*':
         result = multiply (num1, num2);
         printf ("%.2lf * %.2lf = %.2lf \n", num1, num2, result);
         break;
      case '/':
         result = divide (num1, num2);
         if (num2 != 0) {
            printf ("%.2lf / %.2lf = %.2lf \n", num1, num2, result);
         }
         break;
      default:
         printf ("Error: Invalid operator. \n");
         break;
   }
   return 0;
}
```

Output :
```
Enter an operator (+, -, *, /): +
Enter two numbers: 7 8
7.00 + 8.00 = 15.00
```

7. All the above mentioned programs.

→ The solutions for the programs mentioned in problems 1-6 are provided above.

8. All the programs of loop using recursion.

Ans The recursive solution for the power function and the max/min array search are provided below. The other problems (leap year, swap, calculator) are not suited for recursive solutions.

- Power function:

```
int power_recursive (int a, int b) {
        if (b == 0) {
            return 1;
        }
        return a * power_recursive (a, b - 1);
}
```

- max/min function:

```
void findMinMax (int arr[], int n, int index, int * min, int * max) {
    if (index == n) {
        return;
    }
    if (arr [index] < * min) {
        * min = arr [index]
    }
    if (arr [index] > * max) {
        * max = arr [index];
    }
    findMinMax (arr, n, index + 1, min, max);
}
```