```
1 #include <stdio.h>
 2
 3 int main(void)
 4
 5
        //function declarations
        void MathematicalOperations(int, int, int *, int *, int *, int *, int *);
 6
 7
 8
        //variable declaration
 9
        int a:
10
        int b;
11
        int answer_sum;
12
        int answer difference;
13
        int answer_product;
14
        int answer_quotient;
15
        int answer_remainder;
16
17
        //code
        printf("\n\n");
18
        printf("Enter Value Of 'A' : ");
19
20
        scanf("%d", &a);
21
        printf("\n\n");
22
23
        printf("Enter Value Of 'B' : ");
24
        scanf("%d", &b);
25
26
        // PASSING ADDRESSES TO FUNCTION ... FUNCTION WILL FILL THEM UP WITH
          VALUES ... HENCE, THEY GO INTO THE FUNCTION AS ADDRESS PARAMETERS AND COME
          OUT OF THE FUNCTION FILLED WITH VALID VALUES
27
        // THUS, (&answer_sum, &answer_difference, &answer_product, &answer_quotient,
                                                                                        P
          &answer remainder) ARE CALLED "OUT PARAMETERS" OR "PARAMETERIZED RETURN
          VALUES" ... RETURN VALUES OF FUNCTIONS COMING VIA PARAMETERS
        // HENCE, ALTHOUGH EACH FUNCTION HAS ONLY ONE RETURN VALUE, USING THE CONCEPT 🤝
28
          OF "PARAMETERIZED RETURN VALUES", OUR FUNCTION "MathematicalOperations()"
          HAS GIVEN US 5 RETURN VALUES !!!
29
30
        MathematicalOperations(a, b, &answer_sum, &answer_difference, &answer_product, >>
           &answer_quotient, &answer_remainder);
31
        printf("\n\n");
32
        printf("***** RESULTS ***** : \n\n");
33
        printf("Sum = %d\n\n", answer_sum);
34
35
        printf("Difference = %d\n\n", answer_difference);
36
        printf("Product = %d\n\n", answer_product);
37
        printf("Quotient = %d\n\n", answer quotient);
        printf("Remainder = %d\n\n", answer_remainder);
38
39
        return(0);
40 }
41
   void MathematicalOperations(int x, int y, int *sum, int *difference, int *product, →
       int *quotient, int *remainder)
43 {
        //code
44
```

```
\dots ointer As Out Parameter \verb|\| 01-Method One \verb|\| Pointer As Out Parameter.c
```

```
2
                            // Value at address 'sum' = (x + y)
45
       *sum = x + y;
46
       *difference = x - y; // Value at address 'difference' = (x - y)
47
       *product = x * y;
                           // Value at address 'product' = (x * y)
       *quotient = x / y; // Value at address 'quotient' = (x / y)
48
       *remainder = x % y; // Value at address 'remainder' = (x % y)
49
50 }
51
```