

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      //function declarations
7      void MathematicalOperations(int, int, int *, int *, int *, int *, int *);
8
9      //variable declaration
10     int a;
11     int b;
12     int *answer_sum = NULL;
13     int *answer_difference = NULL;
14     int *answer_product = NULL;
15     int *answer_quotient = NULL;
16     int *answer_remainder = NULL;
17
18     //code
19     printf("\n\n");
20     printf("Enter Value Of 'A' : ");
21     scanf("%d", &a);
22
23     printf("\n\n");
24     printf("Enter Value Of 'B' : ");
25     scanf("%d", &b);
26
27     // PASSING ADDRESSES TO FUNCTION ... FUNCTION WILL FILL THEM UP WITH
28     VALUES ... HENCE, THEY GO INTO THE FUNCTION AS ADDRESS PARAMETERS AND
29     COME OUT OF THE FUNCTION FILLED WITH VALID VALUES
30
31     // THUS, (&answer_sum, &answer_difference, &answer_product,
32     &answer_quotient, &answer_remainder) ARE CALLED "OUT PARAMETERS" OR
33     "PARAMETERIZED RETURN VALUES" ... RETURN VALUES OF FUNCTIONS COMING VIA
34     PARAMETERS
35
36     // HENCE, ALTHOUGH EACH FUNCTION HAS ONLY ONE RETURN VALUE, USING THE
37     CONCEPT OF "PARAMETERIZED RETURN VALUES", OUR FUNCTION
38     "MathematicalOperations()" HAS GIVEN US 5 RETURN VALUES !!!
39
40     answer_sum = (int *)malloc(1 * sizeof(int));
41     if (answer_sum == NULL)
42     {
43         printf("Could Not Allocate Memory For 'answer_sum'. Exiting Now...\n\n");
44         exit(0);
45     }
46
47     answer_difference = (int *)malloc(1 * sizeof(int));
48     if (answer_difference == NULL)
49     {
50         printf("Could Not Allocate Memory For 'answer_difference'. Exiting
51         Now...\n\n");
52         exit(0);
53     }
54
55     answer_product = (int *)malloc(1 * sizeof(int));
56     if (answer_product == NULL)
57     {
```

```
48     printf("Could Not Allocate Memory For 'answer_product'. Exiting
      Now...\n\n");
49     exit(0);
50 }
51
52 answer_quotient = (int *)malloc(1 * sizeof(int));
53 if (answer_quotient == NULL)
54 {
55     printf("Could Not Allocate Memory For 'answer_quotient'. Exiting
      Now...\n\n");
56     exit(0);
57 }
58
59 answer_remainder = (int *)malloc(1 * sizeof(int));
60 if (answer_remainder == NULL)
61 {
62     printf("Could Not Allocate Memory For 'answer_remainder'. Exiting
      Now...\n\n");
63     exit(0);
64 }
65
66 MathematicalOperations(a, b, answer_sum, answer_difference,
      answer_product, answer_quotient, answer_remainder);
67
68 printf("\n\n");
69 printf("***** RESULTS ***** \n\n");
70 printf("Sum = %d\n\n", *answer_sum);
71 printf("Difference = %d\n\n", *answer_difference);
72 printf("Product = %d\n\n", *answer_product);
73 printf("Quotient = %d\n\n", *answer_quotient);
74 printf("Remainder = %d\n\n", *answer_remainder);
75
76 if (answer_remainder)
77 {
78     free(answer_remainder);
79     answer_remainder = NULL;
80     printf("Memory Allocated For 'answer_remainder' Successfully Freed !!!
      \n\n");
81 }
82
83 if (answer_quotient)
84 {
85     free(answer_quotient);
86     answer_quotient = NULL;
87     printf("Memory Allocated For 'answer_quotient' Successfully Freed !!!
      \n\n");
88 }
89
90 if (answer_product)
91 {
92     free(answer_product);
93     answer_product = NULL;
94     printf("Memory Allocated For 'answer_product' Successfully Freed !!!\n
      \n");
95 }
96
```

```
97     if (answer_difference)
98     {
99         free(answer_difference);
100         answer_difference = NULL;
101         printf("Memory Allocated For 'answer_difference' Successfully
102             Freed !!!\n\n");
103     }
104     if (answer_sum)
105     {
106         free(answer_sum);
107         answer_sum = NULL;
108         printf("Memory Allocated For 'answer_sum' Successfully Freed !!!\n
109             \n");
110     }
111     return(0);
112 }
113
114 void MathematicalOperations(int x, int y, int *sum, int *difference, int
115     *product, int *quotient, int *remainder)
116 {
117     //code
118     *sum = x + y;           // Value at address 'sum' = (x + y)
119     *difference = x - y;    // Value at address 'difference' = (x - y)
120     *product = x * y;       // Value at address 'product' = (x * y)
121     *quotient = x / y;      // Value at address 'quotient' = (x / y)
122     *remainder = x % y;     // Value at address 'remainder' = (x % y)
123 }
```