

```
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
        VALUES ... HENCE, THEY GO INTO THE FUNCTION AS ADDRESS PARAMETERS AND
        COME OUT OF THE FUNCTION FILLED WITH VALID VALUES
28     // THUS, (&answer_sum, &answer_difference, &answer_product,
        &answer_quotient, &answer_remainder) ARE CALLED "OUT PARAMETERS" OR
        "PARAMETERIZED RETURN VALUES" ... RETURN VALUES OF FUNCTIONS COMING VIA
        PARAMETERS
29     // HENCE, ALTHOUGH EACH FUNCTION HAS ONLY ONE RETURN VALUE, USING THE
        CONCEPT OF "PARAMETERIZED RETURN VALUES", OUR FUNCTION
        "MathematicalOperations()" HAS GIVEN US 5 RETURN VALUES !!!
30
31     answer_sum = (int *)malloc(1 * sizeof(int));
32     if (answer_sum == NULL)
33     {
34         printf("Could Not Allocate Memory For 'answer_sum'. Exiting Now...\n\n");
35         exit(0);
36     }
37
38     answer_difference = (int *)malloc(1 * sizeof(int));
39     if (answer_difference == NULL)
40     {
41         printf("Could Not Allocate Memory For 'answer_difference'. Exiting
        Now...\n\n");
42         exit(0);
43     }
44
45     answer_product = (int *)malloc(1 * sizeof(int));
46     if (answer_product == NULL)
47     {
```

```
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
```

```
197     if (answer_difference)
198     {
199         free(answer_difference);
200         answer_difference = NULL;
201         printf("Memory Allocated For 'answer_difference' Successfully
                Freed !!!\n\n");
202     }
203
204     if (answer_sum)
205     {
206         free(answer_sum);
207         answer_sum = NULL;
208         printf("Memory Allocated For 'answer_sum' Successfully Freed !!!\n
                \n");
209     }
210
211     return(0);
212 }
213
214 void MathematicalOperations(int x, int y, int *sum, int *difference, int
    *product, int *quotient, int *remainder)
215 {
216     //code
217     *sum = x + y;           // Value at address 'sum' = (x + y)
218     *difference = x - y;    // Value at address 'difference' = (x - y)
219     *product = x * y;       // Value at address 'product' = (x * y)
220     *quotient = x / y;      // Value at address 'quotient' = (x / y)
221     *remainder = x % y;     // Value at address 'remainder' = (x % y)
222 }
223
```