```
#include <stdio.h>
   #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;
        int b;
11
12
        int *answer sum = NULL;
        int *answer difference = NULL;
13
14
        int *answer_product = NULL;
        int *answer_quotient = NULL;
15
16
        int *answer remainder = NULL;
17
18
       //code
19
        printf("\n\n");
20
        printf("Enter Value Of 'A' : ");
21
        scanf("%d", &a);
22
        printf("\n\n");
23
        printf("Enter Value Of 'B' : ");
24
25
        scanf("%d", &b);
26
        // PASSING ADDRESSES TO FUNCTION ... FUNCTION WILL FILL THEM UP WITH
27
          VALUES ... HENCE, THEY GO INTO THE FUNCTION AS ADDRESS PARAMETERS AND
                                                                                    P
          COME OUT OF THE FUNCTION FILLED WITH VALID VALUES
        // THUS, (&answer_sum, &answer_difference, &answer_product,
28
          &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
                                                                                    P
          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
        {
            printf("Could Not Allocate Memory For 'answer sum'. Exitting Now...\n →
34
              \n");
35
            exit(0);
36
        }
37
        answer_difference = (int *)malloc(1 * sizeof(int));
38
39
        if (answer_difference == NULL)
40
        {
41
            printf("Could Not Allocate Memory For 'answer difference'. Exitting
              Now...\n\n");
42
            exit(0);
        }
43
44
45
        answer_product = (int *)malloc(1 * sizeof(int));
        if (answer product == NULL)
46
47
```

```
...nterAsOutParameter\02-MethodTwo\PointerAsOutParameter.c
```

```
2
```

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

```
... nter As Out Parameter \verb|\| 02-Method Two \verb|\| Pointer As Out Parameter.c|
```

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