```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    //function declarations
    void MathematicalOperations(int, int, int *, int *, int *, int *, int *);

    //variable declaration
    int a;
    int b;
    int *answer_sum = NULL;
    int *answer_difference = NULL;
    int *answer_product = NULL;
    int *answer_quotient = NULL;
    int *answer_remainder = NULL;

    //code
    printf("\n\n");
    printf("Enter Value Of 'A' : ");
    scanf("%d", &a);

    printf("\n\n");
    printf("Enter Value Of 'B' : ");
    scanf("%d", &b);

    // 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
    // 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
    // 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 !!!

    answer_sum = (int *)malloc(1 * sizeof(int));
    if (answer_sum == NULL)
    {
        printf("Could Not Allocate Memory For 'answer_sum'. Exitting Now...\n\n");
        exit(0);
    }

    answer_difference = (int *)malloc(1 * sizeof(int));
    if (answer_difference == NULL)
    {
        printf("Could Not Allocate Memory For 'answer_difference'. Exitting Now...\n\n");
        exit(0);
    }

    answer_product = (int *)malloc(1 * sizeof(int));
    if (answer_product == NULL)
    {
        printf("Could Not Allocate Memory For 'answer_product'. Exitting Now...\n\n");
        exit(0);
    }

    answer_quotient = (int *)malloc(1 * sizeof(int));
    if (answer_quotient == NULL)
    {
        printf("Could Not Allocate Memory For 'answer_quotient'. Exitting Now...\n\n");
        exit(0);
    }

    answer_remainder = (int *)malloc(1 * sizeof(int));
    if (answer_remainder == NULL)
    {
        printf("Could Not Allocate Memory For 'answer_remainder'. Exitting Now...\n\n");
        exit(0);
    }

    MathematicalOperations(a, b, answer_sum, answer_difference, answer_product, answer_quotient, answer_remainder);

    printf("\n\n");
    printf("****** RESULTS ****** \n\n");
    printf("Sum = %d\n\n", *answer_sum);
    printf("Difference = %d\n\n", *answer_difference);
    printf("Product = %d\n\n", *answer_product);
    printf("Quotient = %d\n\n", *answer_quotient);
    printf("Remainder = %d\n\n", *answer_remainder);

    if (answer_remainder)
    {
        free(answer_remainder);
        answer_remainder = NULL;
        printf("Memory Allocated For 'answer_remainder' Successfully Freed !!!\n\n");
    }

    if (answer_quotient)
    {
        free(answer_quotient);
        answer_quotient = NULL;
        printf("Memory Allocated For 'answer_quotient' Successfully Freed !!! \n\n");
    }

    if (answer_product)
    {
        free(answer_product);
        answer_product = NULL;
        printf("Memory Allocated For 'answer_product' Successfully Freed !!!\n\n");
    }

    if (answer_difference)
    {
        free(answer_difference);
        answer_difference = NULL;
        printf("Memory Allocated For 'answer_difference' Successfully Freed !!!\n\n");
    }

    if (answer_sum)
    {
        free(answer_sum);
        answer_sum = NULL;
        printf("Memory Allocated For 'answer_sum' Successfully Freed !!!\n\n");
    }

    return(0);
}

void MathematicalOperations(int x, int y, int *sum, int *difference, int *product, int *quotient, int *remainder)
{
    //code
    *sum = x + y;          // Value at address 'sum' = (x + y)
    *difference = x - y;   // Value at address 'difference' = (x - y)
    *product = x * y;      // Value at address 'product' = (x * y)
    *quotient = x / y;     // Value at address 'quotient' = (x / y)
    *remainder = x % y;    // Value at address 'remainder' = (x % y)
}

```