

## Sample Solution: C Program for Assignment 1 Question 1

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

// Function to multiply two matrices
void matrixMultiply(double *A, double *B, double *C, int m, int n, int p) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            C[i * p + j] = 0.0; // Initialize C[i][j]
            for (int k = 0; k < n; k++) {
                C[i * p + j] += A[i * n + k] * B[k * p + j];
            }
        }
    }
}

// Function to print a matrix
void printMatrix(double *matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%.2f\t", matrix[i * cols + j]);
        }
        printf("\n");
    }
}

int main() {
    int m, n, p;

    // Input dimensions
    printf("Enter rows and columns for matrix A (m n): ");
    scanf("%d %d", &m, &n);
    printf("Enter columns for matrix B (p): ");
    scanf("%d", &p);

    // Assume B has n rows (checked below)
    int rowsB = n;

    // Check if multiplication is possible
    if (n <= 0 || m <= 0 || p <= 0) {
        printf("Invalid dimensions! Dimensions must be positive.\n");
    }
}
```

```
    return 1;
}

// Allocate memory for matrices
double *A = (double *)malloc(m * n * sizeof(double));
double *B = (double *)malloc(n * p * sizeof(double));
double *C = (double *)malloc(m * p * sizeof(double));

// Check for allocation failure
if (A == NULL || B == NULL || C == NULL) {
    printf("Memory allocation failed!\n");
    free(A);
    free(B);
    free(C);
    return 1;
}

// Input elements for matrix A
printf("Enter elements of matrix A (%d x %d):\n", m, n);
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        printf("A[%d][%d] = ", i, j);
        scanf("%lf", &A[i * n + j]);
    }
}

// Input elements for matrix B
printf("Enter elements of matrix B (%d x %d):\n", n, p);
for (int i = 0; i < n; i++) {
    for (int j = 0; j < p; j++) {
        printf("B[%d][%d] = ", i, j);
        scanf("%lf", &B[i * p + j]);
    }
}

// Multiply matrices
matrixMultiply(A, B, C, m, n, p);

// Print matrices
printf("\nMatrix A:\n");
printMatrix(A, m, n);
printf("\nMatrix B:\n");
printMatrix(B, n, p);
printf("\nResult Matrix C (A x B):\n");
printMatrix(C, m, p);

// Free memory
free(A);
```

```
    free(B);  
    free(C);  
  
    return 0;  
}
```