# Assignment - 1

1. **Write a C program to determine the given number is odd or even using Bitwise operators.**

**Soln:-**

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

if (num & 1) {

printf("%d is an odd number.\n", num);

} else {

printf("%d is an even number.\n", num);

}

return 0;

}

**2. Write a C program to count the number of bits set in a number.**

**Input:**

**144**

**Output:**

**Count of Set bits: 2**

**Soln:-**

#include <stdio.h>

int main() {

int num;

int count=0;

printf("Enter an integer: ");

scanf("%d", &num);

while (num > 0) {

count += num & 1;

num >>= 1;

}

printf("The Number of Bits is: %d\n",count);

return 0;

}

**3. Write a C program to swap two numbers. Use a function pointer to do this operation.**

**Input:**

**84 25**

**Output:**

**25 84**

**Soln:-**

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1 = 10, num2 = 20;

printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

void (\*swaptr)(int \*, int \*);

swaptr = swap;

swaptr(&num1, &num2);

printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

**4. Write an equivalent pointer expression for fetching the value of array element a[i][j][k][2]**

**Soln:-** \*(\*(\*(\*(a + i) + j) + k) + 2)

**5. Write a C program to Multiply two matrix (n\*n) using pointers.**

**Input:  Output:**

**Size of Row: 3  Product:**

**Size of Column: 3  48 39 30**

**Matrix 1:  102 84 66**

**2 3 4  129 111 93**

**5 6 7**

**8 9 1**

**Matrix 2:**

**9 8 7**

**6 5 4**

**3 2 1**

**Soln:-**

#include <stdio.h>

#include <stdlib.h>

void MatrixMultiplication(int \*\*matrixA, int rA, int cA, int \*\*matrixB, int rB, int cB, int \*\*matrixC) {

int i, j, k;

for (i = 0; i < rA; ++i) {

for (j = 0; j < cB; ++j) {

matrixC[i][j] = 0;

for (k = 0; k < cA; ++k) {

matrixC[i][j] += matrixA[i][k] \* matrixB[k][j];

}

}

}

}

int main() {

int i, j;

int rA, cA, rB, cB;

// Enter Matrix A

printf("Enter Number of Rows in MatrixA: ");

scanf("%d", &rA);

printf("Enter Number of columns in MatrixA: ");

scanf("%d", &cA);

int \*\*matrixA = (int \*\*)malloc(rA \* sizeof(int \*));

for (i = 0; i < rA; ++i) {

matrixA[i] = (int \*)malloc(cA \* sizeof(int));

}

printf("Enter MatrixA: \n");

for (i = 0; i < rA; ++i) {

for (j = 0; j < cA; ++j) {

scanf("%d", &matrixA[i][j]);

}

}

// Enter Matrix B

printf("Enter Number of Rows in MatrixB: ");

scanf("%d", &rB);

printf("Enter Number of columns in MatrixB: ");

scanf("%d", &cB);

int \*\*matrixB = (int \*\*)malloc(rB \* sizeof(int \*));

for (i = 0; i < rB; ++i) {

matrixB[i] = (int \*)malloc(cB \* sizeof(int));

}

printf("Enter MatrixB: \n");

for (i = 0; i < rB; ++i) {

for (j = 0; j < cB; ++j) {

scanf("%d", &matrixB[i][j]);

}

}

// Allocate memory for MatrixC

int \*\*matrixC = (int \*\*)malloc(rA \* sizeof(int \*));

for (i = 0; i < rA; ++i) {

matrixC[i] = (int \*)malloc(cB \* sizeof(int));

}

// Perform Matrix Multiplication

MatrixMultiplication(matrixA, rA, cA, matrixB, rB, cB, matrixC);

// Printing MatrixC

printf("\nThe Matrix Multiplication of MatrixA and MatrixB: \n");

for (i = 0; i < rA; ++i) {

for (j = 0; j < cB; ++j) {

printf("%d ", matrixC[i][j]);

}

printf("\n");

}

// Free dynamically allocated memory

for (i = 0; i < rA; ++i) {

free(matrixA[i]);

}

free(matrixA);

for (i = 0; i < rB; ++i) {

free(matrixB[i]);

}

free(matrixB);

for (i = 0; i < rA; ++i) {

free(matrixC[i]);

}

free(matrixC);

return 0;

}

1. **Find the output of the following // Consider the compiler is 32-bit machine**

**Soln:-** Size of Structure = 8

1. **Find the output of the following // Consider the compiler is 32-bit machine**

**Soln:-** Size of Structure = 24

1. **Find the output of the following // Consider the compiler is 32-bit machine**

**Soln:-** 87654321