

## ASSIGNMENT-1

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

### Program:

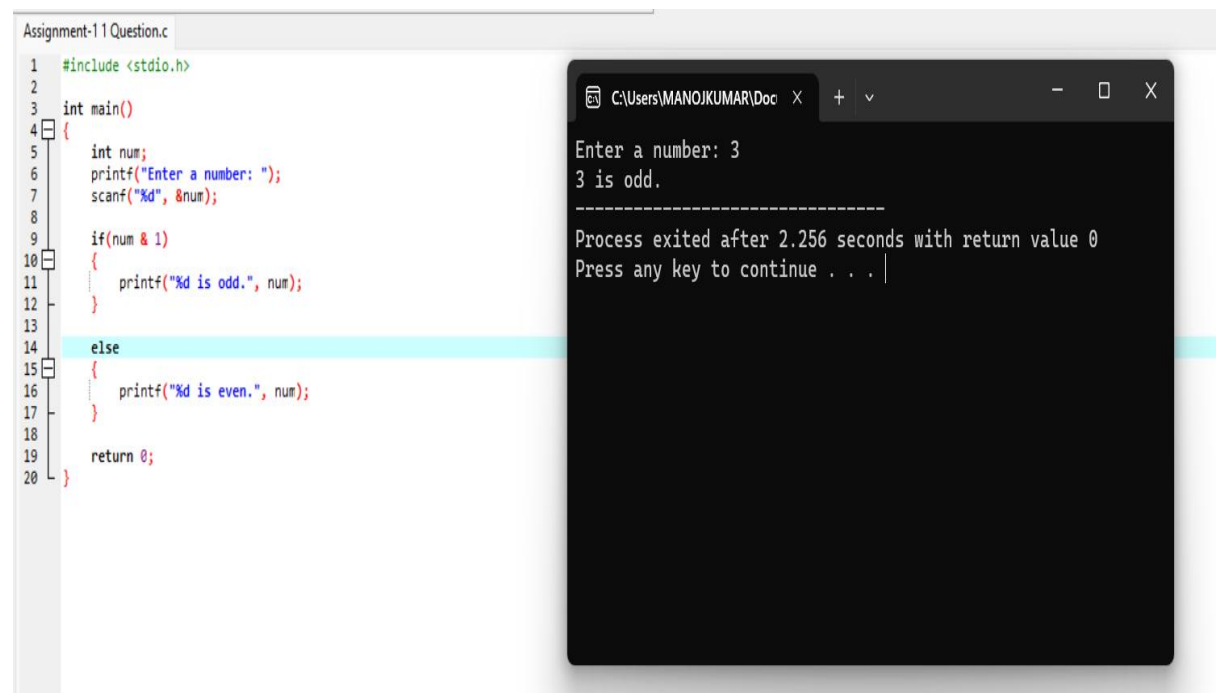
```
#include <stdio.h>

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(num & 1)
    {
        printf("%d is odd.", num);
    }

    else
    {
        printf("%d is even.", num);
    }

    return 0;
}
```



The screenshot shows a code editor window titled "Assignment-11 Question.c" with the following C code:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num;
6     printf("Enter a number: ");
7     scanf("%d", &num);
8
9     if(num & 1)
10    {
11        printf("%d is odd.", num);
12    }
13
14    else
15    {
16        printf("%d is even.", num);
17    }
18
19    return 0;
20 }
```

To the right of the code editor is a terminal window showing the execution output:

```
C:\Users\MANOJKUMAR\Doc x + v - □ X
Enter a number: 3
3 is odd.
-----
Process exited after 2.256 seconds with return value 0
Press any key to continue . . . |
```

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

Input:

144

Output:

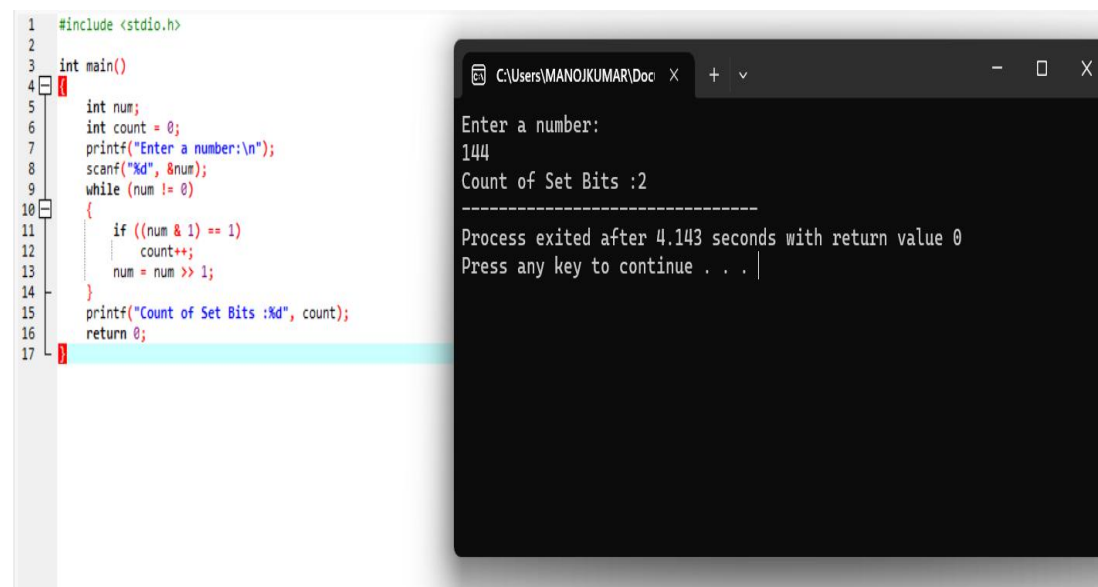
Count of Set bits: 2

**Program:**

```
#include <stdio.h>

int main()
{
    int num;
    int count = 0;

    printf("Enter a number:\n");
    scanf("%d", &num);
    while (num != 0)
    {
        if ((num & 1) == 1)
            count++;
        num = num >> 1;
    }
    printf("Count of Set Bits :%d", count);
    return 0;
}
```



The image shows a code editor on the left and a terminal window on the right. The code editor displays the C program for counting set bits, with line numbers 1 through 17. The terminal window shows the program's execution: it prompts for a number, receives '144', outputs 'Count of Set Bits :2', and then displays a message indicating the process exited after 4.143 seconds.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num;
6     int count = 0;
7     printf("Enter a number:\n");
8     scanf("%d", &num);
9     while (num != 0)
10    {
11        if ((num & 1) == 1)
12            count++;
13        num = num >> 1;
14    }
15    printf("Count of Set Bits :%d", count);
16    return 0;
17 }
```

Terminal Output:

```
C:\Users\MANOJKUMAR\Doc x + v
Enter a number:
144
Count of Set Bits :2
-----
Process exited after 4.143 seconds with return value 0
Press any key to continue . . .
```

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

Input:

84 25

Output:

25 84

**Program:**

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

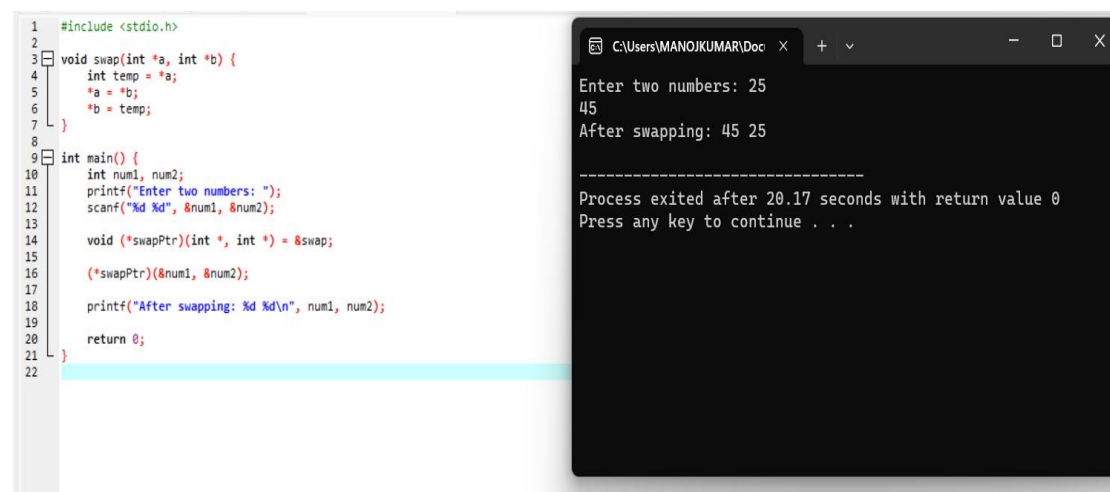
int main() {
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    void (*swapPtr)(int *, int *) = &swap;

    (*swapPtr)(&num1, &num2);

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

    return 0;
}
```



The image shows a code editor on the left and a terminal window on the right. The code editor displays the C program for swapping two numbers using a function pointer. The terminal window shows the program's execution: it prompts for two numbers (25 and 45), prints the swapped values (45 and 25), and then displays a message indicating the process exited after 20.17 seconds with a return value of 0.

```
1 #include <stdio.h>
2
3 void swap(int *a, int *b) {
4     int temp = *a;
5     *a = *b;
6     *b = temp;
7 }
8
9 int main() {
10     int num1, num2;
11     printf("Enter two numbers: ");
12     scanf("%d %d", &num1, &num2);
13
14     void (*swapPtr)(int *, int *) = &swap;
15
16     (*swapPtr)(&num1, &num2);
17
18     printf("After swapping: %d %d\n", num1, num2);
19
20     return 0;
21 }
22
```

C:\Users\MANOJKUMAR\Doc x + - □ x

Enter two numbers: 25  
45  
After swapping: 45 25

-----  
Process exited after 20.17 seconds with return value 0  
Press any key to continue . . .

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

**Program:**

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

```
int main() {
    int a[2][3][4][5] = {
        {
            {
                {111, 112, 113, 114, 115},
                {121, 122, 123, 124, 125},
                {131, 132, 133, 134, 135},
                {141, 142, 143, 144, 145}
            },
            {
                {211, 212, 213, 214, 215},
                {221, 222, 223, 224, 225},
                {231, 232, 233, 234, 235},
                {241, 242, 243, 244, 245}
            },
            {
                {311, 312, 313, 314, 315},
                {321, 322, 323, 324, 325},
                {331, 332, 333, 334, 335},
                {341, 342, 343, 344, 345}
            }
        },
        {
            {
                {411, 412, 413, 414, 415},
                {421, 422, 423, 424, 425},
                {431, 432, 433, 434, 435},
                {441, 442, 443, 444, 445}
            },
            {
                {511, 512, 513, 514, 515},
                {521, 522, 523, 524, 525},
                {531, 532, 533, 534, 535},
                {541, 542, 543, 544, 545}
            },
            {
                {611, 612, 613, 614, 615},
                {621, 622, 623, 624, 625},
                {631, 632, 633, 634, 635},
                {641, 642, 643, 644, 645}
            }
        }
    }
}
```

```

    }
};

int i = 1, j = 2, k = 3;

printf("Value Without using expression a[%d][%d][%d][2]: \n%d\n", i, j, k,
a[i][j][k][2]);

int value = *((*(a + i) + j) + k) + 2);

printf("Value After using Expression of a[%d][%d][%d][2]: \n%d\n", i, j, k, value);

return 0;
}

```

The image shows a C program in a code editor on the left and its execution output in a terminal window on the right.

**Code Editor (Left):** The code defines a 4D array `a` of type `int` with dimensions `4x4x4x4`. It contains nested loops for each dimension, with values ranging from 211 to 645. The program then declares variables `i = 1`, `j = 2`, and `k = 3`. It prints the value of `a[i][j][k][2]` without using an expression, then calculates the same value using pointer arithmetic: `*((*(a + i) + j) + k) + 2`, and prints it again. Finally, it returns 0.

**Terminal Window (Right):** The terminal shows the output of the program. It displays the value 643 for both the direct array access and the pointer arithmetic expression. Below the output, it shows the process exit time and a prompt to press any key to continue.

```

C:\Users\MANOJKUMAR\Doc X + v
Value Without using expression a[1][2][3][2]:
643
Value After using Expression of a[1][2][3][2]:
643

-----
Process exited after 0.03266 seconds with return value 0
Press any key to continue . . .

```

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

**Program:**

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

```
void multiplyMatrices(int (*matrix1), int (*matrix2), int (*result), int row, int col) {  
    int i, j, k;  
    for (i = 0; i < row; i++) {  
        for (j = 0; j < col; j++) {  
            *(result + i * col + j) = 0;  
            for (k = 0; k < col; k++) {  
                *(result + i * col + j) += *(matrix1 + i * col + k) * *(matrix2 + k * col + j);  
            }  
        }  
    }  
}
```

```

int main() {
    int row,col;
    printf("Enter the size of rows :");
    scanf("%d",&row);
    printf("Enter the size of columns :");
    scanf("%d",&col);
    int matrix1[row][col];
    int matrix2[row][col];
    int result[row][col];
    printf("Enter elements in first matrix of size %dx%d\n", row, col);
    int i = 0, j = 0;

    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            scanf("%d", (*(matrix1 + i) + j));
        }
    }
    printf("Enter elements in second matrix of size %dx%d\n", row, col);

    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            scanf("%d", (*(matrix2 + i) + j));
        }
    }

    multiplyMatrices(matrix1, matrix2, result, row, col);
    printf("Product of Matrices:\n");
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < col; j++)
        {
            printf("\n%d", result[i][j]);
        }
    }

    return 0;
}

```

```
1 #include <stdio.h>
2
3 void multiplyMatrices(int (*matrix1), int (*matrix2), int (*result), int row, int col)
4 {
5     int i, j, k;
6     for (i = 0; i < row; i++) {
7         for (j = 0; j < col; j++) {
8             *(result + i * col + j) = 0;
9             for (k = 0; k < col; k++) {
10                 *(result + i * col + j) += *(matrix1 + i * col + k) * *(matrix2 + k * col + j);
11             }
12         }
13     }
14 }
15
16 int main() {
17     int row, col;
18     printf("Enter the size of rows :");
19     scanf("%d", &row);
20     printf("Enter the size of columns :");
21     scanf("%d", &col);
22     int matrix1[row][col];
23     int matrix2[row][col];
24     int result[row][col];
25     printf("Enter elements in first matrix of size %dx%d\n", row, col);
26     int i = 0, j = 0;
27
28     for (i = 0; i < row; i++)
29     {
30         for (j = 0; j < col; j++)
31         {
32             scanf("%d", (*(matrix1 + i) + j));
33         }
34     }
```

```
5
4
3
2
1
Product of Matrices:
48
39
30
102
84
66
129
111
93
-----
Process exited after 15.98 seconds with return value 0
Press any key to continue . . . |
```

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

**Output: 8**

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

**Output:16**

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

**Output:87654321**