ASSIGNMENT:3

1. Write a C program to determine if the least significant bit of a given integer is set (i.e., check if the number is odd).

```
#include <stdio.h>

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

    if (num & 1) {
        printf("The number is odd.\n");
    } else {
        printf("The number is even\n");
    }

    return 0;
}
```

OUTPUT

—----------

Enter an integer: 4

The number is even

2. Create a C program that retrieves the value of the nth bit from a given integer.

```
#include <stdio.h>

int main() {
    int num, n;

    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Enter the bit position to  ");
    scanf("%d", &n);
    int Value = (num >> n) & 1;
    printf("The value of bit %d is: %d\n", n, Value);

    return 0;
}
```

OUTPUT
—----------
Enter an integer: 22
Enter the bit position to 2
The value of bit 2 is: 1

3. Develop a C program that sets the nth bit of a given integer to 1.

```
#include <stdio.h>

int main() {
    int num, n;

    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Enter the position : ");
    scanf("%d", &n);
    num = num | (1 << n);

    printf("The result after setting bit %d is: %d\n", n, num);

    return 0;
}
```

4. Write a C program that clears (sets to 0) the nth bit of a given integer.

```
#include <stdio.h>

int main() {
    int num, n;

    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Enter the position to be cleard ");
    scanf("%d", &n);
    int mask = ~(1 << n);
    num = num & mask;
    printf("The result after clearing bit %d is: %d\n", n, num);

    return 0;
}
```

5. Create a C program that toggles the nth bit of a given integer.

```
#include <stdio.h>

int main() {
    int num, n;

    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Enter the bit position to toggle ");
    scanf("%d", &n);
    int mask = 1 << n;
    num = num ^ mask;
    printf("The result after toggling bit %d is: %d\n", n, num);
    return 0;
}
```

OUTPUT

—----------

```
Enter an integer: 5
Enter the bit position to toggle 2
The result after toggling bit 2 is: 1
```

1. Write a C program that takes an integer input and multiplies it by

```
#include <stdio.h>

int main() {
    int num, n, result;

    printf("Enter the number: ");
    scanf("%d", &num);
    printf("Enter the value of n: ");
    scanf("%d", &n);
    result = num << n;
    printf("Result of %d multiplied by 2^%d is: %d\n", num, n, result);

    return 0;
}
```

OUTPUT

—----------

```
Enter the number: 5
Enter the value of n: 3
Result of 5 multiplied by 2^3 is: 40
```

2. Create a C program that counts how many times you can left shift a number before it overflows (exceeds the maximum value for an integer).

```
#include <stdio.h>
int main() {
    int num = 123;
    int shift_count = 0;
    while (num>0) {
        num = num << 1;
        shift_count++;
    }
    printf("The number of times  overflow is: %d\n", shift_count);

    return 0;
}
```

OUTPUT

—----------

The number of times overflow is: 25

3. Write a C program that creates a bitmask with the first n bits set to 1 using the left shift operator.

```
 #include <stdio.h>

int main() {
    int n, bitmask;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    bitmask = (1 << n) - 1;
    printf("Bitmask with the first %d bits set to 1 is: 0x%X\n", n, bitmask);
    return 0;
}
```

OUTPUT

—----------

Enter the value of n: 8

Bitmask with the first 8 bits set to 1 is: 0xFF

4. Develop a C program that reverses the bits of an integer using left shift and right shift operations.

```c
#include <stdio.h>

int main() {
    unsigned int num, reversed = 0;
    printf("Enter a number: ");
    scanf("%u", &num);

    for (int i = 0; i < 32; i++) {
        if (num & 1) {
            reversed |= (1 << (31 - i));
        }
        num >>= 1;
    }

    printf("Reversed bits: %u\n", reversed);
    return 0;
}
```

OUTPUT

—----------

Enter a number: 13

Reversed bits: 2952790016

5. Create a C program that performs a circular left shift on an integer.

1. Write a C program that takes an integer input and divides it by 2^ n using the right shift operator

```c
#include <stdio.h>

int main() {
    int num, n, result;
    printf("Enter the number: ");
    scanf("%d", &num);
    printf("Enter the value of n: ");
    scanf("%d", &n);
    result = num >> n;
    printf("Result of %d divided by 2^%d is: %d\n", num, n, result);

    return 0;
}
```

OUTPUT

1. Write a C program that takes an integer input and divides it by 2^ n using the right shift operator

```
#include <stdio.h>

int main() {
    int num, n, result;
    printf("Enter the number: ");
    scanf("%d", &num);
    printf("Enter the value of n: ");
    scanf("%d", &n);
    result = num >> n;
    printf("Result of %d divided by 2^%d is: %d\n", num, n, result);

    return 0;
}
```

output

```
------
Enter the number: 32
Enter the value of n: 3
Result of 32 divided by 2^3 is: 4
```

2 Create a C program that counts how many times you can right shift a number before it becomes zero.

```
#include <stdio.h>

int main() {
    int num;
    int count = 0;
    printf("Enter a number: ");
    scanf("%u", &num);
    while (num > 0) {
        num = num >> 1;
        count++;
    }
    printf("The number of count is: %d\n", count);

    return 0;
}
```

output
Enter a number: 123
The number of count is: 7

3. Write a C program that extracts the last n bits from a given integer using the right shift operator.

```
#include <stdio.h>

int main() {
    int num;
    int n;
    int mask;
   int result;
    printf("Enter the number: ");
    scanf("%u", &num);
    printf("Enter the number of bits to extract: ");
    scanf("%d", &n);
    mask = (1 << n) - 1;
    printf("The last %d bits of the number %d are: %d\n", n, num, result);


    return 0;
}
```

OUTPUT

Enter the number: 7
Enter the number of bits to extract: 1
The last 1 bits of the number 7 are: 0

4 .Develop a C program that uses the right shift operator to create a bitmask that checks if specific bits are set in an integer.

```
#include <stdio.h>

int main() {
```

```
    unsigned int num;
    int bitPos;
    unsigned int bitmask;

    printf("Enter the number: ");
    scanf("%u", &num);
    printf("Enter the bit position to check: ");
    scanf("%d", &bitPos);
    bitmask = 1 << bitPos;
    unsigned int shiftedNum = num >> bitPos;
    if (shiftedNum & 1) {
        printf("The bit at position %d is set (1).\n", bitPos);
    } else {
        printf("The bit at position %d is not set (0).\n", bitPos);
    }

    return 0;
}
```

CLASS WORK

—----------

```
#include<stdio.h>
void myfun (void);
int main()
{
myfun();
myfun();
myfun();
myfun();
}
void myfun(){
    static int count =0;
    count =count +1;
    printf("the function is executed %d times\n",count);
}
```

```
the function is executed 1 times
the function is executed 2 times
the function is executed 3 times
the function is executed 4 times
```

```
#include<stdio.h>
void TestFile_myfun(void);
int mainPrivatedata;
int main(){

   mainPrivatedata=100;

   printf("001 main private data %d",mainPrivatedata);

   TestFile_myfun();

   printf("002 main private data %d",mainPrivatedata);

   return 0;
}
```

```
extern int  mainPrivatedata;

void TestFile_myfun(){
   mainPrivatedata =500;
}
```

Output

```
—---
002 main private data 100
002 main private data 500
```

```
#include<stdio.h>
void TestFile_myfun(void);

int main(){

   TestFile_myfun();


   return 0;
}
static void change_clock(int system_clock)
{
```

```
    printf(" the system clock changed %d",system_clock);
}


extern void change_clock(int);

void TestFile_myfun(){

    change_clock(500);

}


#include<stdio.h>
int main(){
    char A =40;
    char B = 30;
    printf("the output after OR(|)%d \n",A|B);
    printf("the output after AND(&)%d\n ",A&B);
    printf("the output after XOR(^)%d\n ",A^B);
    printf("the output after NOT(~)%d\n ",~A);

}



output
----------
the output after OR(|)62
the output after AND(&)8
the output after XOR(^)54
the output after NOT(~)-41
```