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).

```c
/*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 a;
    printf("Enter the number : ");
    scanf("%d",&a);

    if(a&1==1)
    {
        printf("\nThe number is odd");

    }
    else{
        printf("The number is even ");
    }
    return 0;
 }
```

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

```c
/*2. Create a C program that retrieves the value

of the nth bit from a given integer.*/
#include <stdio.h>
int main()
{
    int num;
    int n;
    printf("\nEnter the number :");
    scanf("%d",&num);
    printf("\nEnter the position of the bit :  ");
    scanf("%d",&n);
    int bit = (num>>n)&1;
    printf("\nThe bit is : %d",bit);
    return 0;

}
```

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

```c
/*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("\nEnter the number : ");
    scanf("%d",&num);
    printf("\nBit position : ");
    scanf("%d",&n);
    int res_bit = (num>>n)|1;
    printf("The bit at position %d is : %d",n,res_bit);
    return 0;
}
```

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

```c
/*

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("\nEnter the number : ");
    scanf("%d",&num);
    printf("\nBit position : ");
    scanf("%d",&n);
    int mask = 1<<n;
    int res_1 = ~mask;
    int res_bit = res_1 & num;
    printf("After clearing the nth bit the result is %d",res_bit);
    return 0;
}
```

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

```c
/*
5. Create a C program that toggles the nth bit of a given integer.

*/
#include <stdio.h>
int main()
{
    int num,n;
```

```
    printf("\nEnter the number : ");
    scanf("%d",&num);
    printf("\nBit position : ");
    scanf("%d",&n);
    int mask = 1<<n;
    int res = num^mask;
    printf("After toggling the result is %d",res);
    return 0;

}
```

SET 2 :

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

```
/*
Write a C program that takes an integer input and multiplies it by
2^n using the left shift operator.
************/
#include <stdio.h>
int main(){
    int num,n;
    printf("\nEnter an integer : ");
    scanf("%d",&num);//25
    printf("\nEnter the position bit : ");
    scanf("%d",&n);
    int res = (num<<n);
    printf("\nThe result is : %d",res);
    return 0 ;
}
```

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).

```
/*
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>
#include <limits.h> //for the inbuilt constant INT_MAX
int main()
{
    int num;
```

```
    int count = 0;
    printf("\nEnter an integer : ");
    scanf("%d",&num);
    while(num <= (INT_MAX/2))
    {
        num=num<<1;
        count++;
    }
    printf("\nCount of left shift before the number overflows = %d",count);
    return 0;
}
```

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

```
/*
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 num,n;
    int bitMask;
    printf("\nEnter an integer : ");
    scanf("%d",&num);
    printf("\nEnter the position bit : ");
    scanf("%d",&n);
    int bit = (1<<n);
    int res = num|bit;
    printf("Bitmask value = ")
}
```

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

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

 #include <stdio.h>
 int main()
```

```
{
    int num,rev_int=0;
    printf("\nEnter an integer : ");
    scanf("%d",&num);

    printf("\nThe given number in binary format :");
    for(int i=sizeof(num)*8-1;i>=0;i--)
    {
        printf("%d",(num>>i)&1);
    }
    // Reversing the number
    for (int i = 0; i <sizeof(num)*8; i++)
    {
        int bit = (num>>i)&1;
        rev_int = (rev_int<<1)|bit;
    }
    printf("\nThe reversed integer in binary format : ");
    for(int i=sizeof(rev_int)*8-1;i>=0;i--){
        printf("%d",(rev_int>>i)&1);
    }
    printf("\nThe Reversed number in decimal format : %u",rev_int);
    return 0;


}
```

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

```
/*
Create a C program that performs a circular left shift on an integer.
*/
#include <stdio.h>
#include <stdint.h>
int main(){
    uint32_t num;
    int shift;

    printf("Enter an integer: ");
    scanf("%u", &num);

    printf("Enter the number of positions to shift: ");
    scanf("%d", &shift);
    int bit = sizeof(num)*8;
    shift %= bit;
    uint32_t rev_res = (num<<shift)|(num>>(bit-shift));
    printf("Result after circular left shift: %u\n", rev_res);
    return 0;
```

```
}
```

SET 3 :

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

```
/*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;
    printf("\nEnter an integer : ");
    scanf("%d",&num);
    printf("\nEnter the position bit : ");
    scanf("%d",&n);
    int res = (num>>n);
    printf("\nThe result is : %d",res);
    return 0 ;
}
```

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

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


#include <stdio.h>
#include <limits.h> //for the inbuilt constant INT_MAX
int main()
{
    int num;
    int count = 0;
    printf("\nEnter an integer : ");
    scanf("%d",&num);
    while(num <= (INT_MAX/2))
    {
        num=num>>1;
        count++;
    }
    printf("\nCount of left shift before the number overflows = %d",count);
```

```
    return 0;
}
```

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

```
/*3. Write a C program that extracts the last n bits from a given integer
using the right shift operator.*/
#include <stdio.h>
#include <limits.h> //for the inbuilt constant INT_MAX
int main()
{
    int num;
    printf("\nEnter an integer : ");
    scanf("%d",&num);
    int mask = (1<<n)|1;
    int res_bit = num&mask;
    printf("\nThe last n bits  = %d",res_bit);
    return 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.

```
/*

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>
#include <stdbool.h>
int main() {
    int num, n;

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

    printf("Enter the bit position to check (0 for least significant bit): ");
    scanf("%d", &n);
    int shift = (num >> n) & 1;
    if (shift)
    {
        printf("Bit %d is set (1).\n", n);
```

```
    }
    else
    {
        printf("Bit %d is not set (0).\n", n);
    }

    return 0;


}
```