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 n;
  printf("enter a number \n");
  scanf("%d",&n);
  if((n&1)==1)
     printf("odd");

  else
  printf("even");
  return 0;
}
```

Output

```
enter a number
12
even
```

---

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

```
#include<stdio.h>
int main()
{
   int n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number to be found \n");
  scanf("%d",&n);
  printf("the answer is %d",(num >> n) & 1);
  return 0;
}
```

Output

```
enter a number
15
enter the bit number to be found
3
the answer is 1
```

---

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

```
#include<stdio.h>
int main()
{
   int n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number to be set \n");
  scanf("%d",&n);
  printf("the answer is %d",((1 << n)|num));
  return 0;
}
```

Output

```
Enter a number
12
Enter the bit number to be set
3
The answer is 12
```

---

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

```
#include<stdio.h>
int main()
{
   int n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number to be cleared \n");
  scanf("%d",&n);
  printf("the answer is %d",~(1 << n)&num);
  return 0;
}
```

Output

```
Enter a number
16
Enter the bit number to be cleared
4
The answer is 0
```

---

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

```
#include<stdio.h>
int main()
{
   int n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number to be toggled \n");
  scanf("%d",&n);
  printf("the answer is %d",(1 << n)^num);
  return 0;
}
```

Output

```
Enter a number
18
Enter the bit number to be toggled
2
The answer is 22
```

---

6. 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 n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number  \n");
  scanf("%d",&n);
  printf("the answer is %d",num<<n);
  return 0;
```

Output

```
Enter a number
10
Enter the bit number to be toggled
3
The answer is 80
```

---

7. 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;
   printf("enter the number \n");
   scanf("%d",&num);
   int count = 0;
   while (num > 0 && (num << 1) > 0)
   {
      num <<= 1;
      count++;
   }

   printf(" %d \n", count);
```

Output

Enter the number
12
27

---

8. 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,mask=0;
   printf("Enter the number of bits \n ");
   scanf("%d", &n);
   for (int i = 0; i < n; i++)
   {
      mask = (mask << 1) | 1;
   }

   printf("Bitmask : %d \n",mask);

   return 0;
```

```
}
```

Output

```
Enter the number of bits
4
Bitmask :15
```

---

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

```
#include <stdio.h>
int main()
{
   unsigned int num, reverse = 0;
   int n;
   printf("Enter an integer: ");
   scanf("%u", &num);
   for (n = 0; num >> n; n++);
   for (int i = 0; i < n; i++)
   {
      reverse = reverse << 1;
      reverse |= (num & 1);
      num = num >> 1;
   }
   printf("Reversed bits: %u\n", reverse);
   return 0;
}
```

Output

```
Enter an integer: 13
Reversed bits: 11
```

---

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

11. 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 n,num;
   printf("enter a number \n");
   scanf("%d",&num);
  printf("enter the bit number  \n");
  scanf("%d",&n);
  printf("the answer is %d",num>>n);
  return 0;
}
```

Output

```
Enter a number
12
Enter the bit number
2
The answer is 3
```

---

12. 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;
   printf("enter the number \n");
   scanf("%d",&num);
   int count = 0;
   while (num != 0)
```

```
    {
       num >>= 1;
       count++;
    }
    printf(" the answer is %d \n", count);
    return 0;
}
```

Output

Enter the number

13

The answer is 4

---

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

```
#include <stdio.h>
int main()
{
   unsigned int num,bits;
   int n;
   printf("Enter an integer: ");
   scanf("%u", &num);
   printf("Enter the number of bits to extract: \n");
   scanf("%d", &n);
   bits = num & ((1 << n) - 1);
   printf("The last : %u\n",bits);
   return 0;
}
```

Output

Enter an integer
13
Enter the number of bits to extract
2
The last: 1

---

14. 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 bit;
   printf("Enter an integer: \n ");
   scanf("%u", &num);
   printf("Enter the bit position to check \n ");
   scanf("%d", &bit);
   unsigned int bitmask = 1 << bit;
   if (num & bitmask) {
      printf("The bit is 1");
    else
      printf("The bit is 0");
   return 0;
}
```

Output

```
Enter an integer
12
Enter the bit position to check
3
The bit is 1
```
