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 the integer:");

   scanf("%d",&n);

   if(n&1){

      printf("LSB is set,number is odd \n");

   }

   else{

      printf("LSB is not set,number is even \n");

   }

   return 0;

}
```

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

```
#include<stdio.h>

int main(){

   int num,bit;

   printf("Enter the number");

   scanf("%d",&num);

   printf("Enter bit position");

   scanf("%d",&bit);


   int mask=1;

   for(int i=0;i<bit;i++){

      mask=mask*2;
```

```
    }
    int bitValue=(num/mask)%2;
    printf("Value of %d bit is:%d \n",bit,bitValue);
    return 0;
}
```

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

```
#include<stdio.h>
int main()
{
    int num,bit;
    printf("Enter the number");
    scanf("%d",&num);
    printf("Enter bit position");
    scanf("%d",&bit);
    int mask=1;
    for(int i=0;i<bit;i++){
        mask=mask*2;
    }
    num=num+mask;
    printf("New value after setting %d bit to 1 is:%d \n",bit,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,bit;
    printf("Enter the number");
    scanf("%d",&num);
    printf("Enter bit position");
    scanf("%d",&bit);
    int mask=1;
    for(int i=0;i<bit;i++){
        mask=mask*2;
    }
    num=num-(num%(mask*2));
    printf("New value after clearing %d bit is:%d \n",bit,num);
    return 0;
}
```

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

```
#include<stdio.h>
int main()
{
    int num,bit;
    printf("Enter the number");
    scanf("%d",&num);
    printf("Enter bit position");
    scanf("%d",&bit);
    int mask=1;
    for(int i=0;i<bit;i++){
        mask=mask*2;
    }
    if (num%(mask*2)>=mask){
        num=num-mask;
    }else{
        num=num+mask;
    }
    }
    printf("New value after toggling %d bit  is:%d \n",bit,num);
    return 0;
}
```

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 number,n;
    printf("Enter the number:");
    scanf("%d",&number);
    printf("Enter the power");
    scanf("%d",&n);
    int a=number<<n;
    printf("output is %d \n",a);
    return 0;
}
```

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,count=0;
    printf("Enter the number:");
    scanf("%d",&num);
    while(num>0){
        num<<=1;
        count=count+1;
    }
    printf("number of times left shifted before overflow:%d\n",count-1);
    return 0;
}
```

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;
    printf("Enter the number of bits:");
    scanf("%d",&n);
    mask=(1<<n)-1;
    printf("Bitmask creating first n bits is %d \n",mask);
    return 0;
}
```

9.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,reverse=0;
    printf("Enter the integer:")
    scanf("%d",&num);
    while(num>0){
        reverse=(reverse<<1)|(num&1);
        num>>1;
    }
    printf("Reversed bits %d\n",reverse);
}
```

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

```
#include <stdio.h>
int main() {
   int num,shift;
   printf("Enter an integer: ");
   scanf("%d", &num);
   printf("Enter number of positions to shift: ");
   scanf("%d", &shift);
   int bits = sizeof(num) * 7;
   shift = shift % bits;
   int res = (num << shift) |  (bits - shift);
   printf("Result after circular left shift: %d\n", result);
   return 0;
}
```

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 number,n;
   printf("Enter the number:");
   scanf("%d",&number);
   printf("Enter the power");
   scanf("%d",&n);
```

```
    int a=number>>n;

    printf("output is %d \n",a);

    return 0;

}
```

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,count=0;

    printf("Enter the number:");

    scanf("%d",&num);

    while(num>0){

        num=num>>1;

        count=count+1;

    }

    printf("number of times right shifted before it becomes zero:%d\n",count-1);

    return 0;

}
```

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(){

    int num,n;

    printf("Enter the number :");

    scanf("%d",&n);

    printf("Enter the number of bit :");

    scanf("%d",&n);

    int bits=num>>(n-1);
    printf("extracting last n bits from a given integer %d \n",n,num,bits);
    return 0;
}
```

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()
{
    int num,bit;
    printf("Enter the integer:");
    scanf("%d",&num);
    printf("Enter the bit position");
    scanf("%d",&bit);
    int bit_position=(num>>bit)&1;
    if(bit_position){
        printf("Bit at position %d set",bit);
    }
    else{
        printf("Bit at position %d not set",bit);
    }
    return 0;
}
```