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>
intt main(){
        int num;
        printf(“Enter the number=”);
        scanf(“%d”, &num);
        if((num & 1)==0){
                printf(“The number is even”);
        }
        else{
                printf(“The number is odd”);
        }
        return 0;
}
```

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

```
#include <stdio.h>

int getNthBit(int number, int n) {
   int mask = 1 << n;
   return (number & mask) != 0 ? 1 : 0;
}

int main() {
   int number, n;
   printf(“Enter an integer: “);
```

```
    scanf("%d", &number);
    printf("Enter the bit position to retrieve: ");
    scanf("%d", &n);

    int bit = getNthBit(number, n);
    printf("The %d-th bit is: %d\n", n, bit);
    return 0;
}
```

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

```
#include <stdio.h>

int setNthBit(int number, int n) {
    int mask = 1 << n;
    return number | mask;
}

int main() {
    int number, n;
    printf("Enter an integer: ");
    scanf("%d", &number);
    printf("Enter the bit position to set to 1: ");
    scanf("%d", &n);

    int result = setNthBit(number, n);
    printf("Result after setting the %d-th bit to 1: %d\n", n, result);
    return 0;
}
```

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

```
#include <stdio.h>

int clearNthBit(int number, int n) {
    int mask = ~(1 << n);
    return number & mask;
}

int main() {
    int number, n;
    printf("Enter an integer: ");
    scanf("%d", &number);
    printf("Enter the bit position to clear (set to 0): ");
    scanf("%d", &n);

    Int result = clearNthBit(number, n);
    Printf("Result after clearing the %d-th bit: %d\n", n, result);
    Return 0;
}
```

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

```
 #include <stdio.h>

Int toggleNthBit(int number, int n) {
    Int mask = 1 << n;
    Return number ^ mask;
}
```

```
Int main() {
    Int number, n;
    Printf(“Enter an integer: “);
    Scanf(“%d”, &number);
    Printf(“Enter the bit position to toggle: “);
    Scanf(“%d”, &n);

    Int result = toggleNthBit(number, n);
    Printf(“Result after toggling the %d-th bit: %d\n”, n, result);
    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 Multiply_By_Power2(int num,int n){
    Return num<<n;
}
Int main(){
    Int num,n;
    Printf(“Enter the integer:\n”);
    Scanf(“%d”, & num);
    Printf(“Enter the power of n:”);
    Scanf(“%d”,& n);
    Int result = Multiply_By_Power2(num,n);
    Printf(“%d multiplied by 2^%d is: %d\n”,num,n,result);
    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>
#include <limits.h>

Int countShiftsUntilOverflow() {
   Int count = 0;
   Int value = 1;

   While (value > 0) {
      Value <<= 1;
      Count++;
   }

   Return count;
}

Int main() {
   Int shifts = countShiftsUntilOverflow();
   Printf(“Number of left shifts before overflow: %d\n”, shifts);
   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 createBitmask(int n) {
   Return (1 << n) – 1;
```

```
}

Int main() {
   Int n;
   Printf(“Enter the number of bits to set to 1: “);
   Scanf(“%d”, &n);

   Int bitmask = createBitmask(n);
   Printf(“Bitmask with first %d bits set to 1: %d\n”, n, bitmask);
   Return 0;
}
```

9)
```
#include <stdio.h>

Unsigned int reverseBits(unsigned int num) {
   Unsigned int reversed = 0;
   Int bitCount = sizeof(num) * 8;

   For (int i = 0; i < bitCount; i++) {
      Reversed <<= 1;
      Reversed |= (num & 1);
      Num >>= 1;
   }

   Return reversed;
}

Int main() {
   Unsigned int number;
   Printf(“Enter an integer: “);
```

```
    Scanf("%u", &number);

    Unsigned int result = reverseBits(number);
    Printf("Reversed bits: %u\n", result);
    Return 0;
}
```

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

```
#include <stdio.h>

Unsigned int reverseBits(unsigned int num) {
    Unsigned int reversed = 0;
    Int bitCount = sizeof(num) * 8;

    For (int i = 0; i < bitCount; i++) {
        Reversed <<= 1;
        Reversed |= (num & 1);
        Num >>= 1;
    }

    Return reversed;
}

Int main() {
    Unsigned int number;
    Printf("Enter an integer: ");
    Scanf("%u", &number);

    Unsigned int result = reverseBits(number);
```

```
    Printf("Reversed bits: %u\n", result);
    Return 0;
}
```

11)Write a C program to extract bits from 14th to 9th bits of a number.

```
#include <stdio.h>

Int extractBits(int number) {
    Return (number >> 9) & 0x3F; // 0x3F is 111111 in binary (6 bits)
}

Int main() {
    Int number;
    Printf("Enter a number: ");
    Scanf("%d", &number);

    Int result = extractBits(number);
    Printf("Bits from 14th to 9th: %d\n", result);

    Return 0;
}
```

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

```
#include <stdio.h>

Int divideBy2n(int number, int n) {
    Return number >> n;
}

Int main() {
    Int number, n;
```

```
    Printf(“Enter a number: “);
    Scanf(“%d”, &number);
    Printf(“Enter n (power of 2 to divide by): “);
    Scanf(“%d”, &n);

    Int result = divideBy2n(number, n);
    Printf(“Result of division by 2^%d: %d\n”, n, result);

    Return 0;
}
```

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

```
#include <stdio.h>

int countRightShifts(int number) {
    int count = 0;
    while (number != 0) {
        number >>= 1;
        count++;
    }
    return count;
}

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

```
    int shifts = countRightShifts(number);
    printf("Number of right shifts before it becomes zero: %d\n", shifts);

    return 0;
}
```

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

```
#include <stdio.h>

int extractLastNBits(int number, int n) {
    return number & ((1 << n) - 1);
}

int main() {
    int number, n;
    printf("Enter a number: ");
    scanf("%d", &number);
    printf("Enter the number of bits to extract: ");
    scanf("%d", &n);

    int result = extractLastNBits(number, n);
    printf("Last %d bits: %d\n", n, result);

    return 0;
}
```

5)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 checkBits(int number, int mask) {
```

```
    Return (number & mask) == mask;
}

Int main() {
    Int number, mask;
    Printf(“Enter a number: “);
    Scanf(“%d”, &number);
    Printf(“Enter a bitmask (in hexadecimal): “);
    Scanf(“%x”, &mask);

    If (checkBits(number, mask)) {
        Printf(“The specified bits are set.\n”);
    } else {
        Printf(“The specified bits are not all set.\n”);
    }

    Return 0;
}
```