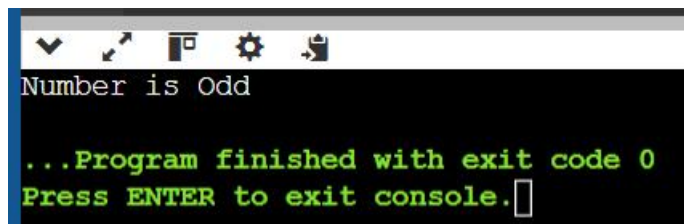


## ASSIGNMENT 4

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 a= 13;
    if ((a & 1) == 1){
        printf("Number is Odd");
    }
    else{
        printf("Number is Even");
    }
    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,n;

    printf("Enter a Integer");

    scanf("%d",&num);

    printf("Enter the nth digit");

    scanf("%d",&n);

    if (((num>>n)&1)==1){

        printf("Value of %dth bit of %d = 1 ",n,num);

    }

}
```

```

else{

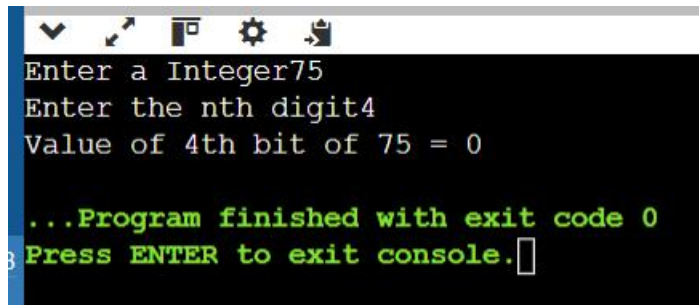
    printf("Value of %dth bit of %d = 0 ",n,num);

}

return 0;

}

```



```

Enter a Integer75
Enter the nth digit4
Value of 4th bit of 75 = 0

...Program finished with exit code 0
Press ENTER to exit console.

```

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

```

#include<stdio.h>

int main(){

    int num,n;

    int mask,result;

    printf("Enter a Number :");

    scanf("%d",&num);

    printf("Enter the nth bit value :");

    scanf("%d",&n);

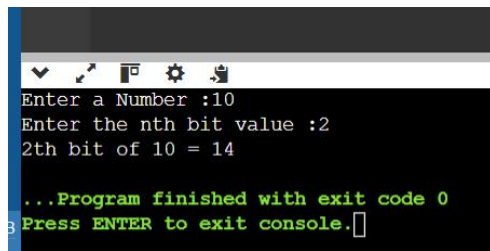
    mask=1<<n;

    result=num|mask;

    printf("%dth bit of %d = %d",n,num,result);

}

```



```
Enter a Number :10
Enter the nth bit value :2
2th bit of 10 = 14

...Program finished with exit code 0
Press ENTER to exit console.
```

**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;

    int mask,result;

    printf("Enter a Number :");

    scanf("%d",&num);

    printf("Enter the nth bit value :");

    scanf("%d",&n);

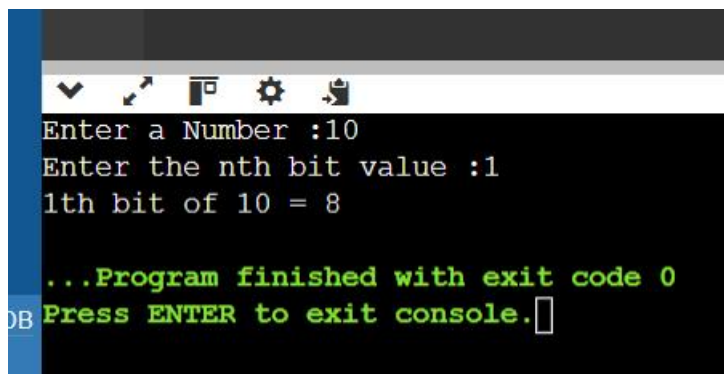
    mask=1<<n;

    mask=~(mask);

    result=num&mask;

    printf("%dth bit of %d = %d",n,num,result);

}
```



```
Enter a Number :10
Enter the nth bit value :1
1th bit of 10 = 8

...Program finished with exit code 0
Press ENTER to exit console.
```

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

```
#include<stdio.h>

int main(){

    int num,n;

    int mask,result;

    printf("Enter a Number :");

    scanf("%d",&num);

    printf("Enter the nth bit value :");

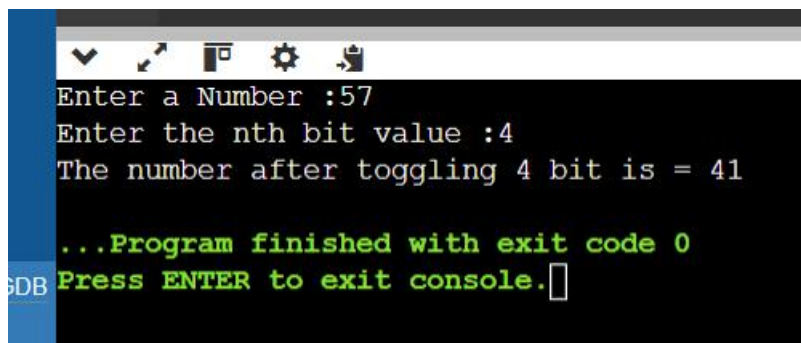
    scanf("%d",&n);

    mask=1<<n;

    result=num^mask;

    printf("The number after toggling %d bit is = %d",n,result);

}
```

A screenshot of a terminal window with a dark background and light-colored text. The window has a title bar with standard icons. The text in the terminal shows the program's execution: it prompts for a number (57) and a bit value (4), then displays the result (41) after toggling the 4th bit. At the bottom, it shows the program finished with exit code 0 and prompts the user to press ENTER to exit the console. A blue vertical bar is visible on the left side of the terminal window.

```
Enter a Number :57
Enter the nth bit value :4
The number after toggling 4 bit is = 41

...Program finished with exit code 0
Press ENTER to exit console.
```

**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 num, n;

    printf("Enter a number: ");

    scanf("%d", &num);
```

```

printf("Enter the value of n: ");

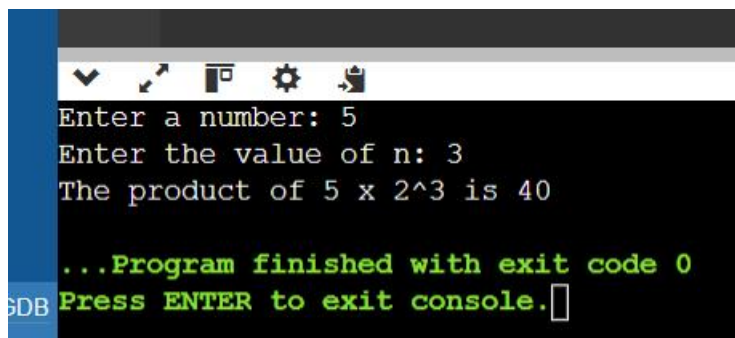
scanf("%d", &n);

printf("The Product of %d x 2^%d is %d", num, n, num << n);

return 0;

}

```



```

Enter a number: 5
Enter the value of n: 3
The product of 5 x 2^3 is 40
...Program finished with exit code 0
Press ENTER to exit console.

```

**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 main() {

    int num, count = 0;

    printf("Enter a number: ");

    scanf("%d", &num);

    while (num <= INT_MAX / 2) {

        num <<= 1;

        count++;

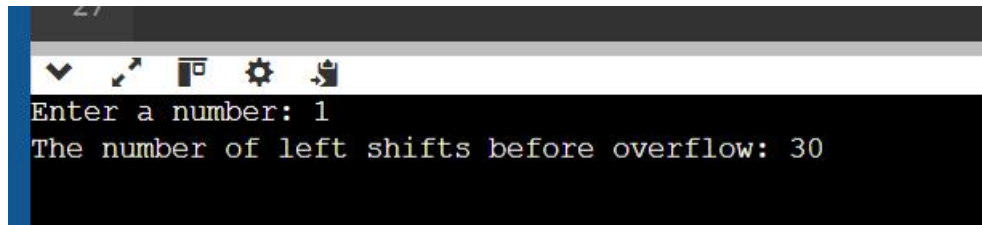
    }

    printf("The number of left shifts before overflow: %d\n", count);
}

```

```
return 0;

}
```



```
Enter a number: 1
The number of left shifts before overflow: 30
```

**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 num, n;

    printf("Enter number: ");

    scanf("%x", &num);

    printf("Enter value of n: ");

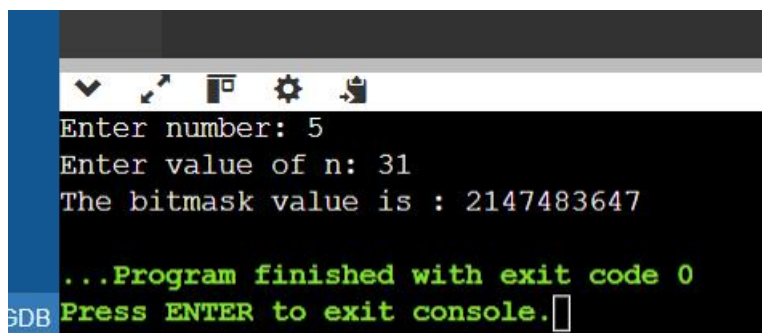
    scanf("%d", &n);

    int bitmask = (1 << n) - 1;

    printf("The bitmask value is : %d", bitmask);

    return 0;

}
```



```
Enter number: 5
Enter value of n: 31
The bitmask value is : 2147483647

...Program finished with exit code 0
Press ENTER to exit console.
```

**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_bit = 0;

    printf("Enter a integer: ");

    scanf("%u", &num);

    for (int i = 0; i < 32; i++) {

        reverse_bit = (reverse_bit << 1) | (num & 1);

        num = num >> 1;

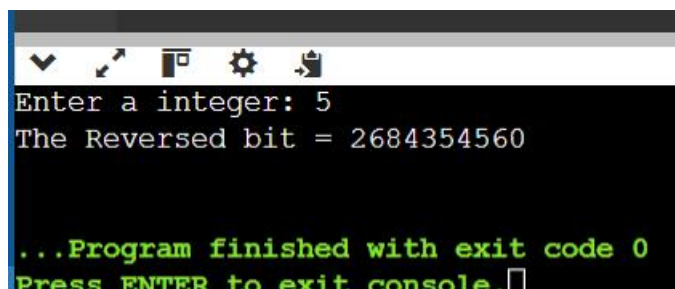
    }

    printf("The Reversed bit = %u\n", reverse_bit);

    return 0;

}

```



```

Enter a integer: 5
The Reversed bit = 2684354560

...Program finished with exit code 0
Press ENTER to exit console.

```

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

```

#include <stdio.h>

int main() {

    unsigned int num, shifts, result;

    printf("Enter a integer: ");

    scanf("%u", &num);

    printf("Enter shifts: ");

    scanf("%u", &shifts);

```

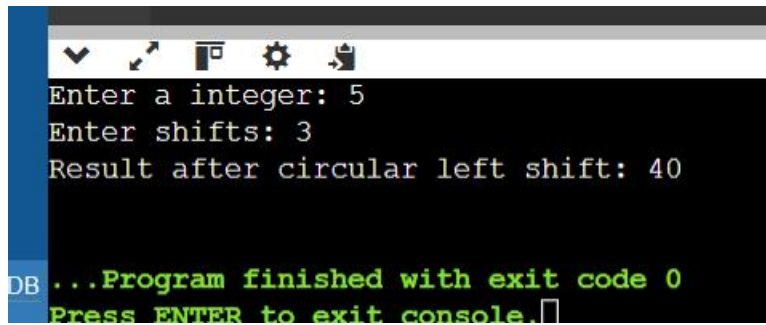
```

    result = (num << shifts) | (num >> (32 - shifts));

    printf("Result after circular left shift: %u\n", result);

    return 0;
}

```



```

Enter a integer: 5
Enter shifts: 3
Result after circular left shift: 40
DB ...Program finished with exit code 0
Press ENTER to exit console.

```

**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 num, n, out;

    printf("Enter an integer: ");

    scanf("%d", &num);

    printf("Enter n: ");

    scanf("%d", &n);

    out = num >> n;

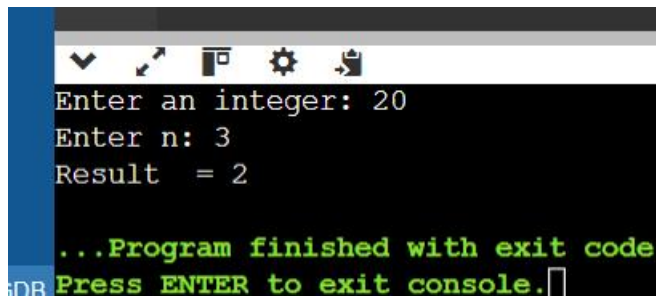
    printf("Result = %d", out);

    return 0;

}

```





```
Enter an integer: 20
Enter n: 3
Result = 2
...Program finished with exit code
Press ENTER to exit console.
```

**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() {

    unsigned int num, count = 0;

    printf("Enter the number: ");

    scanf("%d", &num);

    int out = num;

    while (num > 0) {

        num >>= 1;

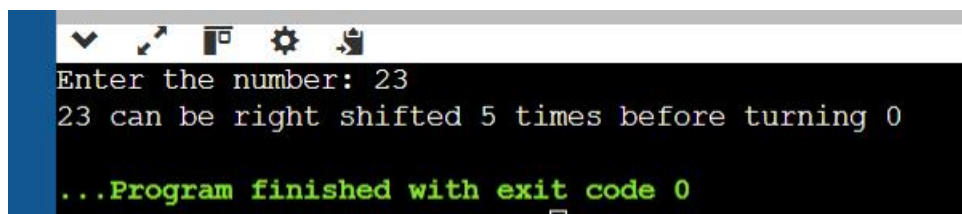
        ++count;

    }

    printf("%d can be right shifted %d times before turning 0", out, count);

    return 0;

}
```



```
Enter the number: 23
23 can be right shifted 5 times before turning 0
...Program finished with exit code 0
Press ENTER to exit console.
```

**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, result;

    printf("Enter a number: ");

    scanf("%d", &num);

    printf("Enter n : ");

    scanf("%d", &n);

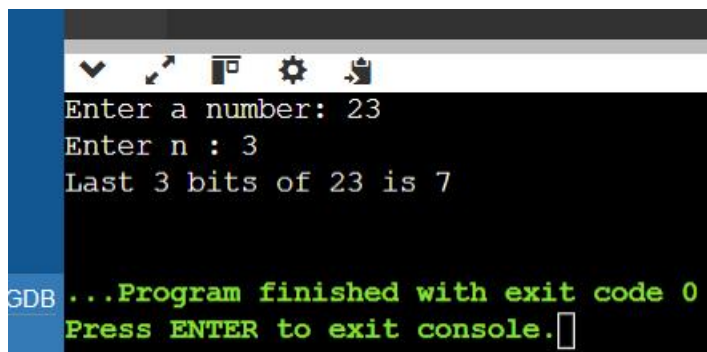
    result = num & ((1 << n) - 1);

    printf("Last %d bits of %d is %d\n",n,num,result);

    return 0;

}

```



```

Enter a number: 23
Enter n : 3
Last 3 bits of 23 is 7

GDB ...Program finished with exit code 0
Press ENTER to exit console.

```

**13. 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, n;

    printf("Enter a number: ");

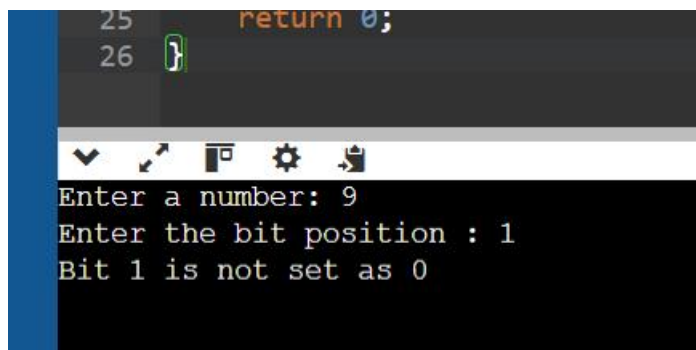
    scanf("%d", &num);

    printf("Enter the bit position : ");

    scanf("%d", &n);

```

```
if ((num >> n) & 1) {  
    printf("Bit %d is set as 1\n", n);  
} else {  
    printf("Bit %d is not set as 0\n", n);  
}  
  
return 0;  
}
```



The screenshot shows a code editor with two lines of code: line 25 contains `return 0;` and line 26 contains a closing curly brace `}`. Below the editor is a terminal window with a dark background and light-colored text. The terminal displays the following prompts and user input:   
Enter a number: 9  
Enter the bit position : 1  
Bit 1 is not set as 0