Dat_4_Assignment_Abhirami

1. 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 how many times to double the number: ");
    scanf("%d", &n);

    int result = num << n;
    printf("Result: %d\n", result);

    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).

```
#include <stdio.h>

int main() {
    int num = 1;
    int count = 0;

    while (num > 0) {
        num = num << 1;
        count++;
    }

    printf("You can shift left %d times before overflow.\n", count - 1);
    return 0;
}
```

3. 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;
    printf("Enter number of bits to set to 1: ");
    scanf("%d", &n);

    int bitmask = (1 << n) - 1;
    printf("Bitmask: %d\n", bitmask);

    return 0;
}
```

4. 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, reversed = 0;
    printf("Enter a number: ");
    scanf("%u", &num);

    for (int i = 0; i < 32; i++) {
        reversed = reversed << 1;
        reversed = reversed | (num & 1);
        num = num >> 1;
    }

    printf("Reversed bits: %u\n", reversed);
    return 0;
}
```

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

```
#include <stdio.h>

int main() {
    unsigned int num;
    int shift;
    printf("Enter a number: ");
    scanf("%u", &num);
    printf("Enter shift amount: ");
    scanf("%d", &shift);

    unsigned int result = (num << shift) | (num >> (32 - shift));
```

```
    printf("After circular left shift: %u\n", result);

    return 0;
}
```

6. 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("Enter a number: ");
    scanf("%d", &num);
    printf("Enter the value of n: ");
    scanf("%d", &n);

    int result = num >> n;

    printf("Result after dividing %d by 2^%d is: %d\n", num, n, result);
    return 0;
}
```

7. 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;
    int count = 0;
    printf("Enter a number: ");
    scanf("%u", &num);

    while (num > 0) {
        num = num >> 1;
        count++;
    }

    printf("The number can be right-shifted %d times before it becomes zero.\n", count);
    return 0;
}
```

8. 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, n;
    printf("Enter a number: ");
    scanf("%u", &num);
    printf("Enter the number of bits to extract: ");
    scanf("%u", &n);

    unsigned int mask = (1 << n) - 1;
    unsigned int result = num & mask;

    printf("The last %u bits of %u are: %u\n", n, num, result);
    return 0;
}
```

9. 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, bit_position;
    printf("Enter a number: ");
    scanf("%u", &num);
    printf("Enter the bit position to check (0 to 31): ");
    scanf("%u", &bit_position);

    if ((num >> bit_position) & 1) {
        printf("Bit %u is set.\n", bit_position);
    } else {
        printf("Bit %u is not set.\n", bit_position);
    }

    return 0;
}
```