STATIC:

```
#include<stdio.h>
void myfun(void);
int main(){
myfun();
myfun();
myfun();
myfun();
return 0;
}
void myfun(){
static int count=0;
count=count+1;
printf("The function is executed %d times \n",count);
}
```

EXTERN:

main.c

```
#include<stdio.h>
int mainPrivateData;
void Testfile_myfun();
int main(){
mainPrivateData=100;
printf("001mainPrivateData = %d\n",mainPrivateData);
Testfile_myfun();
printf("002mainPrivateData = %d",mainPrivateData);
return 0;
}
```

Testfile.c

```
extern int mainPrivateData;
void Testfile_myfun(){
mainPrivateData=500;
}
```

STATIC AND EXTERN:

main.c:

```
#include<stdio.h>
int mainPrivateData;
void Testfile_myfun();
int main(){
Testfile_myfun();
return 0;
}
static void change_clock(int system_clock){
printf("System clock changed to %d \n",system_clock);
}
```

Testfile.c:

```
extern int mainPrivateData;
extern void change_clock(int);
void Testfile_myfun(){
```

```
change_clock(500);
}
```

BITWISE OPERATORS

```
#include<stdio.h>
int main(){
char A=40;
char B=30;
printf("The output after Bitwise OR(|) operation is %d \n",(A|B));
printf("The output after Bitwise AND(&) operation is %d \n",(A&B));
printf("The output after Bitwise XOR(^) operation is %d \n",(A^B));
printf("The output after Bitwise NOT(~) operation is %d \n",(~A));
return 0;
}
```

Output:

```
The output after Bitwise OR(|) operation is 62
The output after Bitwise AND(&) operation is 8
The output after Bitwise XOR(^) operation is 54
The output after Bitwise NOT(~) operation is -41
```

BITWISE OPERATION QUESTIONS

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 num;
   printf("enter the value of integer");
   scanf("%d",&num);
   if(num & 1)
   {
      printf("odd");
   }
   else{
      printf("even");
   }
}
```

Output

```
enter the value of integer:20
Even
```

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, bit_value;


    printf("Enter an integer: ");
    scanf("%d", &num);

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


    bit_value = (num >> n) & 1;


    printf("The value of the %dth bit is: %d\n", n, bit_value);

    return 0;
}
```

Output
Enter an integer:15
Enter the bit position:3
The value of the 3th bit is: 1

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

```
#include <stdio.h>

int main() {
    int num, n;


    printf("Enter an integer: ");
    scanf("%d", &num);

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


    num = num | (1 << n);


    printf("The number after setting the %dth bit to 1 is: %d\n", n, num);
```

```
    return 0;
}
```

Output
Enter an integer:12
Enter the bit position:4
The number after setting the 4th bit to 1 is: 28

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;


    printf("Enter an integer: ");
    scanf("%d", &num);

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

    num = num & ~(1 << n);


    printf("The number after clearing the %dth bit is: %d\n", n, num);

    return 0;
}
```

Output
Enter an integer:25
Enter the bit position:4
The number after clearing the 4th bit is 9

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

```
#include <stdio.h>

int main() {
    int num, n;


    printf("Enter an integer: ");
```

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

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


    num = num ^ (1 << n);


    printf("The number after toggling the %dth bit is: %d\n", n, num);

    return 0;
}
```

Output

```
Enter an integer:23
Enter the bit position:4
The number after toggling the 4th bit is 7
```

LEFT SHIFT PROGRAMS

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 the value of integer");
    scanf("%d",&num);
```

```
    printf("enter the no of bits to shift");
    scanf("%d",&n);
    int x=num<<n;
    printf("the value of x after shifting %d bits to left is %d",n,x);

}
```

Output
enter the value of integer:4
enter the no of bits to shift:2
the value of x after shifting 2 bits to left is 8

2)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>
#include <limits.h>

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


    while (num <= INT_MAX / 2) {
        num = num << 1;
        count++;
    }

    printf("The number can be left-shifted %d times before overflowing.\n", count);

    return 0;
}
```

Output
The number can be left-shifted 30 times before overflowing.

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 the number of bits to set : ");
    scanf("%d", &n);
```

```
    int mask = (1 << n) - 1;


    printf("The bitmask with the first %d bits set to 1 is: %d\n", n, mask);

    return 0;
}
```

Output
Enter the number of bits to set : 3
The bitmask with the first 3 bits set to 1 is: 7

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

```
#include <stdio.h>

unsigned int reverse_bits(unsigned int num) {
    unsigned int reversed_num = 0;
    int bit_count = 32;

    for (int i = 0; i < bit_count; i++) {
        reversed_num <<= 1;
        int bit = num & 1;
        reversed_num |= bit;
        num >>= 1;
    }
    return reversed_num;
}
    int main() {
    unsigned int num;
    printf("Enter a number: ");
    scanf("%u", &num);
    unsigned int reversed_num = reverse_bits(num);
    printf("Original number: %u\n", num);
    printf("Reversed number: %u\n", reversed_num);
    return 0;
}
```

Output:
Enter a number: 637219
Original number: 637219
Reversed number: 3298660352

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

RIGHT SHIFT PRGMS

1)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, result;


    printf("Enter an integer: ");
    scanf("%d", &number);

    printf("Enter the value of n: ");
    scanf("%d", &n);


    result = number >> n;

    printf("Result of dividing %d by 2^%d is: %d\n", number, n, result);

    return 0;
}
```

OUTPUT

```
Enter an integer:15
Enter the value of n:4
Result of dividing 15 by 2^4 is: 0
```

2) 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 an integer: ");
    scanf("%d", &num);
```

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

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

    return 0;
}
```

OUTPUT
Enter an integer:20
The number can be right shifted 5 times before it becomes zero

3)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 an integer: ");
    scanf("%d", &num);

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


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

    printf("The last %d bits of %d are: %d\n", n, num, result);

    return 0;
}
```

Output
Enter an integer:26
Enter the number of bits to extract:3
The last 3 bits of 26 are:2

4) 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 check(int num, int position[], int count) {
    int result = 1;
    for (int i = 0; i < count; i++) {
        int mask = 1 << position[i];
        if ((num & mask) == 0) {
            result = 0;
            break;
        }
    }
    return result;
}
int main() {
    int num;
    int count;
    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Enter the number of bits to check: ");
    scanf("%d", &count);
    int position[count];
    printf("Enter the bit positions to check (0-indexed from right):\n");
    for (int i = 0; i < count; i++) {
        scanf("%d", &position[i]);
    }
    if (check(num, position, count)) {
        printf("All specified bits are set to 1.\n");
    } else {
        printf("At least one of the specified bits is not set.\n");
    }
    return 0;}
```