When two integers are divided, the result is truncated. That is, when the computer calculates 23/4, instead of getting 5.75 it gets 5. The computer literally asks how many times 4 goes into 23, and doesn’t care anything about the remainder.

The modulo function (%) is very useful when doing integer arithmetic because it computes the remainder.

The following program illustrates these points.

```c
#include <stdio.h>
#include <math.h>
int main(void)
{
    int n, k, r, d;
    printf("Please enter n,k\n");
    scanf("%i %i", &n, &k);
    d = n / k;
    r = n % k;
    printf("Integer division: %i divided by %i is %i\n", n, k, d);
    printf("The remainder when %i is divided by %i is %i\n", n, k, r);
    return 0;
}
```

Comments:

[1] A fairly common bug in programs is forgetting that integer arithmetic truncates things. Thus if you have two doubles \( x \) and \( y \) and have a line of C code reading \( y = (1/2) \times x \); the result will be \( y = 0 \) because the computer will set \( 1/2 \) to zero. You can get the right answer either with \( y = x/2 \); or with \( y = (1./2.) \times x \); In the latter case the decimal points force the computer to do real number arithmetic. It is a good habit to put decimal points after all numbers that are not integers to avoid this sort of bug.