## The Integers and Division

This area of discrete mathematics belongs to the area of Number Theory. Some applications of the concepts in this section include generating pseudorandom numbers, assigning computer memory locations to files, and cryptology.

The set of integers will be denoted by

$$\mathbf{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$$

## Division

**Def**. If *a* and *b* are integers with $a \neq 0$, we say that ***a* divides *b*** if there is an integer *c* such that *b* = *ac*. When *a* divides *b* we say that ***a* is a factor of *b*** and that ***b* is a multiple of *a***. The notation $a \mid b$denotes that a divides b. We write $a \nmid b$ when *a* does not divide *b*. Note that $a \mid b$ is different from *a/b* and $\frac{a}{b}$.

**Exercises:**

1. True or False. $2 \mid 6$
2. True or False. $12 \mid 6$
3. True or False. $2 \mid 6 = 3$
4. True or False. *7* is a multiple of *35*
5. True or False. *7* is a factor of *35*

**Exercise**: Let *n* and *d* be positive integers. How many positive integers not exceeding *n* are divisible by *d*?

**Solution**:

**Theorem 1**. Let *a*, *b* and *c* be integers such that *a* and *b* are not zero. Then

1. if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$
2. if $a \mid b$, then $\forall c \in Z,\ a \mid cb$
3. if $a \mid b$ and $b \mid c$, then $a \mid c$

**Proof of part 1.**

## THE DIVISION ALGORITHM

**Theorem 4** The division algorithm
Let *a* be an integer and *d* a positive integer. Then there are unique integers *q* and *r*, with $0 \le r < d$, such that $a = dq + r$. *Note*: *d* is called the **divisor**, *a* is called the **dividend**, *q* is called the **quotient**, and *r* is called the **remainder**.
Proof: given later if time permits.

**Examples:**

1. 101 = 11(9) + 2
2. Suppose we are given -101 and 9. Find the quotient and remainder as asserted by the division algorithm.

## Modular Arithmetic

**Def**. Let *a* be an integer and *m* be a positive integer. We denote by ***a* mod *m*** the remainder when *a* is divided by *m*. ***Note*** that from the definition of remainder the remainder is a nonnegative integer less than *m*.

**Examples:**

9 mod 5 = 4
8 mod 5 = 3
7 mod 5 = 2
6 mod 5 = 1
5 mod 5 = 0
4 mod 5 = 4
3 mod 5 = 3
2 mod 5 = 2
1 mod 5 = 1
0 mod 5 = 0

-1 mod 5 = 4
-2 mod 5 = ____
-3 mod 5 = ____
-4 mod 5 = ____
-5 mod 5 = 0

8 mod 2 = ___
14 mod 3 = ___
5 mod 2 = ___
36 mod 8 = ___

Def. If *a* and *b* are integers and *m* is a positive integer, then ***a* is congruent to *b* modulo *m*** if *m* divides *a* - *b*. We use the notation $a \equiv b \bmod m$ to indicate that *a* and *b* are congruent modulo *m*, otherwise we write $a \not\equiv b \bmod m$.

**Examples:**

TRUE/FALSE

i. $13 \equiv 3 \bmod 5$
ii. $13 \equiv 3 \bmod 4$
iii. $14 \equiv 26 \bmod 6$
iv. $18 \equiv 32 \bmod 2$

**Theorem 6** Let $m$ be a positive integer. The integer $a$ and $b$ are congruent modulo $m$ if and only if there is an integer $k$ such that $a = b + km$.

**Proof.**

**Theorem** Let $m$ be a positive integer. If $a \equiv b \bmod m$ and $c \equiv d \bmod m$, then $a + c \equiv b + d (\bmod m)$ and $ac \equiv bd (\bmod m)$.

**Proof.**

## Applications of Congruences

**Def**. A **hashing function** *h* assigns memory location *h(k)* to the record that has *k* as its key, with the objective that the record be easily and quickly retrieved without much of a search.

**Example**: A university campus maintains records for each of its 111 students. Memory locations can be assigned so that student records can be retrieved quickly using the hashing function *h(S)* = *S* mod *111* where *S* a students social security number (which is the key that uniquely identifies each student.)

Suppose that the following students with the given social security numbers enroll, and that their data must be entered into the computer. Which memory locations would they be assigned to using the hashing function *h(S)* = *S* mod *111?*

253 45 6849
253 45 6626

| Memory location | Social Security |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| ⋮ | |
| 97 | |
| 98 | |
| 99 | |
| 100 | |

Since hashing functions are not one-to-one, more than one file may be assigned to a memory location. When this happens we say that a **collision** occurs. One way to resolve a collision is to assign the first free location following the occupied memory location (We will use this method, but this is not the only way.)

**Example**: Using the hash function *h(S)* = *S* mod *10*, and the collision resolution scheme described, store the sequence of values 23, 14, 85, 40, 24, 18, 33, 58, 50.

| Memory location | Social Security |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |

- Hashing functions are also used when a computer program is compiled, the compiler builds a symbol table to store information about the identifiers used the program.