announcements

Reading assignment

Modular arithmetic
  4.1-4.2, 7th edition
  3.4-3.5, 6th edition

Homework 3 due now
  Graded Homework 2 and Solutions available
  Homework 4 out later today

functions review

• A function from $A$ to $B$
  • an assignment of exactly one element of $B$
    to each element of $A$.
  • We write $f: A \rightarrow B$.
  • “Image of $a$” = $f(a)$

• Domain of $f: A$    Codomain of $f: B$

• Range of $f$ = set of all images of elements of $A$

review: set theory

$x \in A : \text{“}x \text{ is an element of } A\text{”}$
$x \notin A : \neg(x \in A)$

$A \subseteq B \equiv \forall x \ (x \in A \rightarrow x \in B)$

$A = B \equiv (A \subseteq B \land B \subseteq A)$

$A \cup B = \{ x : (x \in A) \lor (x \in B) \}$

$A \cap B = \{ x : (x \in A) \land (x \in B) \}$

$\mathcal{P}(A) = \{ B : B \subseteq A \}$  $A \times B = \{(a,b) : a \in A, b \in B \}$

Some applications: Characteristic vectors, private key cryptography
number theory (and applications to computing)

- Branch of Mathematics with direct relevance to computing
- Many significant applications
  - Cryptography
  - Hashing
  - Security
- Important tool set

modular arithmetic

- Arithmetic over a finite domain
- In computing, almost all computations are over a finite domain
what are the values computed?

```csharp
public void Test1() {
    byte x = 250;
    byte y = 20;
    byte z = (byte) (x + y);
    Console.WriteLine(z);
}

public void Test2() {
    sbyte x = 120;
    sbyte y = 20;
    sbyte z = (sbyte) (x + y);
    Console.WriteLine(z);
}
```

what are the values computed?

```csharp
namespace ConsoleApplication1 {
    class Program {
        static void Main(string[] args) {
            byte x = 250;
            byte y = 20;
            byte z = (byte) (x + y);
            Console.WriteLine(z);
        }
    }
}
```

```csharp
namespace ConsoleApplication1 {
    class Program {
        static void Main(string[] args) {
            sbyte x = 120;
            sbyte y = 20;
            sbyte z = (sbyte) (x + y);
            Console.WriteLine(z);
        }
    }
}
```

divisibility

Integers $a, b$, with $a \neq 0$, we say that $a$ divides $b$ if there is an integer $k$ such that $b = ka$. The notation $a \mid b$ denotes “$a$ divides $b$.”

division theorem

Let $a$ be an integer and $d$ a positive integer. Then there are unique integers $q$ and $r$, with $0 \leq r < d$, such that $a = dq + r$.

$$q = a \div d \quad r = a \mod d$$

Note: $r \geq 0$ even if $a < 0$. Not quite the same as $a \% d$
Let a and b be integers, and m be a positive integer. We say a is congruent to b modulo m if m divides a – b. We use the notation $a \equiv b \pmod{m}$ to indicate that a is congruent to b modulo m.

Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then
- $a + c \equiv b + d \pmod{m}$
- $ac \equiv bd \pmod{m}$
Let $n$ be an integer. Prove that $n^2 \equiv 0 \pmod{4}$ or $n^2 \equiv 1 \pmod{4}$

**Signed integer representation**

- **n-bit signed integers**
  - Suppose $-2^{n-1} < x < 2^{n-1}$
  - First bit as the sign, n-1 bits for the value
  
  $99 = 64 + 32 + 2 + 1$
  $18 = 16 + 2$

  For $n = 8$:
  
  99: 0110 0011
  -18: 1001 0010

  Any problems with this representation?

**Two's complement representation**

- **n bit signed integers**, first bit will still be the sign bit
  - Suppose $0 \leq x < 2^{n-1}$,
    - $x$ is represented by the binary representation of $x$
  - Suppose $0 \leq x \leq 2^{n-1}$,
    - $-x$ is represented by the binary representation of $2^n - x$

  **Key property**: Two's complement representation of any number $y$ is equivalent to $y \mod 2^n$ so arithmetic works $\mod 2^n$

  $99 = 64 + 32 + 2 + 1$
  $18 = 16 + 2$

  For $n = 8$:
  
  99: 0110 0011
  -18: 1110 1110
### signed vs two’s complement

<table>
<thead>
<tr>
<th></th>
<th>Signed</th>
<th>Two’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7</td>
<td>1111</td>
<td>1000</td>
</tr>
<tr>
<td>-6</td>
<td>1110</td>
<td>1001</td>
</tr>
<tr>
<td>-5</td>
<td>1101</td>
<td>1100</td>
</tr>
<tr>
<td>-4</td>
<td>1100</td>
<td>1101</td>
</tr>
<tr>
<td>-3</td>
<td>1011</td>
<td>0000</td>
</tr>
<tr>
<td>-2</td>
<td>1010</td>
<td>0001</td>
</tr>
<tr>
<td>-1</td>
<td>1001</td>
<td>0010</td>
</tr>
<tr>
<td>0</td>
<td>1000</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>2</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>3</td>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>4</td>
<td>0111</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>1001</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>1010</td>
<td>1100</td>
</tr>
<tr>
<td>8</td>
<td>1101</td>
<td>1101</td>
</tr>
</tbody>
</table>

- For \(0 < x \leq 2^{n-1}\), \(-x\) is represented by the binary representation of \(2^n - x\).
- To compute this: Flip the bits of \(x\) then add 1:
  - All 1’s string is \(2^n - 1\), so
    - Flip the bits of \(x\) \(\equiv\) replace \(x\) by \(2^n - 1 - x\)