Week 4 Part A highlights

- Represent numbers in different ways (base expansions and prime factorization)
- Define the base expansion of a positive integer, specifically decimal, binary, hexadecimal, and octal.
- Convert between expansions in different bases of a positive integer.
- Define and use the div and mod operators.
- Trace an algorithm specified in pseudocode
- Euclid's algorithm for efficiently computing gcd of two numbers
- Modular arithmetic and applications

Representing positive integers

Definition (Rosen p. 246, zyBook 6.3) For b an integer greater than 1 and n a positive integer, the **base b expansion of n** is

$$(a_{k-1} \cdots a_1 a_0)_b$$

where k is a positive integer, $a_0, a_1, \ldots, a_{k-1}$ are nonnegative integers less than b, $a_{k-1} \neq 0$, and

$$n = a_{k-1}b^{k-1} + \cdots + a_1 b + a_0$$

The base b expansion of a positive integer n is a string over the alphabet $\{x \in \mathbb{N} \mid x < b\}$ whose leftmost character is nonzero.

Base b	Collection of possible coefficients in base b expansion of a positive integer
Binary ($b = 2$)	$\{0, 1\}$
Ternary ($b = 3$)	$\{0, 1, 2\}$
Octal ($b = 8$)	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
Decimal ($b = 10$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Hexadecimal ($b = 16$)	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ letter coefficient symbols represent numerical values $(A)_{16} = (10)_{10}$ $(B)_{16} = (11)_{10}$ $(C)_{16} = (12)_{10}$ $(D)_{16} = (13)_{10}$ $(E)_{16} = (14)_{10}$ $(F)_{16} = (15)_{10}$

Binary $b = 2$	Octal $b = 8$	Decimal $b = 10$	Hexadecimal $b = 16$
$(1401)_2$			
	$(1401)_8$		
		$(1401)_{10}$	
			$(1401)_{16}$

New! An algorithm is a finite sequence of precise instructions for solving a problem.

Algorithm for calculating integer part of log

```
1  procedure log(n: a positive integer)
2  r := 0
3  while n > 1
4     r := r + 1
5     n := n div 2
6  return r {r holds the result of the log operation}
```

n	r	$n > 1$?
6		

Two algorithms for constructing base b expansion from decimal representation

Algorithm 1: Start with highest power of b, i.e. at left-most coefficient of expansion

Calculating integer part of $\log_b$

```
1  procedure logb(n,b: positive integers with b>1)
2  r := 0
3  while n > 1
4     r := r + 1
5     n := n div b
6  return r {r holds the result of the log_b(n,b) operation}
```

Calculating base b expansion, from left

```
1  procedure baseb1(n,b: positive integers with b>1)
2  v := n
3  k := logb(n,b) + 1
4  for i := 1 to k
5     a_{k-i} := 0
6     while v ≥ b^{k-i}
7        a_{k-i} := a_{k-i} + 1
8        v := v - b^{k-i}
9  return (a_{k-1},...,a_0){(a_{k-1}...a_0)_b is the base b expansion of n}
```

The Division Algorithm (Rosen 4.1 Theorem 2, p. 239, zBook 6.2) Let n be an integer and d a positive integer. There are unique integers q and r, with $0 \leq r < d$, such that $n = dq + r$. In this case, d is called the divisor, n is called the dividend, q is called the quotient, and r is called the remainder. We write $q = n$ **div** d and $r = n$ **mod** d.

Extra example: How do **div** and **mod** compare to / and % in Java and python?

Algorithm 2: Start with right-most coefficient of expansion

n	b	q	k	a_k	$q \neq 0$?

Calculating base b expansion, from right

```
1  procedure baseb2(n,b: positive integers with b>1)
2  q := n
3  k := 0
4  while q ≠ 0
5     a_k := q mod b
6     q := q div b
7     k := k+1
8  return (a_{k-1},...,a_0){(a_{k-1}...a_0)_b is the base b expansion of n}
```

Idea: (when $k > 1$) $n = a_{k-1}b^{k-1} + \cdot + a_1 b + a_0 = b(a_{k-1}b^{k-2} + \cdots + a_1) + a_0$ so $a_0 = n$ **mod** b and $a_{k-1}b^{k-2} + \cdots + a_1 = n$ **div** b.

Fundamental Theorem of Arithmetic (zyBook 6.4): Every positive integer greater than 1 can be written uniquely as a prime or as the product of two or more primes where the prime factors are written in order of non-decreasing size.

Another way to represent positive integers is in terms of _____

Which of the following are consistent with the Fundamental Theorem of Arithmetic for representing positive numbers
A. $3 = 3$
B. $100 = (1)(2)(2)(5)(5)$
C. $20 = (4)(5)$
D. $9 = (3)(3)$
E. All of the above

Definition: Greatest Common Divisor (GCD): Let a and b be integers, not both zero. The largest positive integer d such that $d \mid a$ and also $d \mid b$ is called the greatest common divisor of a and b. The greatest common divisor of a and b is denoted by $gcd(a, b)$.

$gcd(24, 36) =$ $gcd(17, 22) =$

Finding gcd using prime factorization

Definition: Least Common Multiple (LCM): Let a and b be positive integers. Their least common multiple is the smallest positive integer that is divisible by both a and b. It is denoted by $lcm(a, b)$.

$lcm(24, 36) =$ $lcm(17, 22) =$

Euclid's algorithm is an efficient method for computing gcd. Find $gcd(198, 252)$:

Euclid's algorithm for calculating gcd

```
1  procedure gcd(x,y: positive integers with x<y)
2  a := x
3  b := y
4  while a≠0
5     r := b mod a
6     b := a
7     a := r
8  return ________          { gcd(x,y) is                    }
```

GCD Theorem: Let x and y be two positive integers. Then $gcd(x, y) = gcd(y \ mod \ x, x)$

Modular Arithmetic: If we select an integer $m > 1$, then mod m can be seen as a function that takes a single integer x as input and outputs x mod m, resulting in a number from the set $\mathbb{Z}_m = \{0, 1, 2, ..., m-1\}$

Addition and multiplication modulo m : We can define addition and multiplication on the elements in this set $\mathbb{Z}_m$ in the usual way, except that the mod m function is applied afterwards to ensure that the result will again be in $\mathbb{Z}_m$.

Theorem: Let m be an integer larger than 1. Let x and y be any integers. Then

$(x + y)$ `mod` $m = (\ (x$ `mod` $m) + (y$ `mod` $m)\ )$ `mod` m

(xy) `mod` $m = (\ (x$ `mod` $m)(y$ `mod` $m)\ )$ `mod` m

Finding Multiplicative inverses mod n (when they exist)