CPSC 121

Instructor: Seyyed Aliasghar Hosseini; Course Coordinator: Jessie Pan

July 5th - Module 00 & Module 01

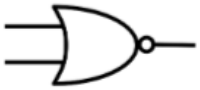
Module 00

- Learning Tips
    - Reading and understanding the answer to a problem is **NOT** learning
    - Make summaries, teach the subject to someone else or compare your solution of exercises and problems with friends
    - *Frustration* is part of learning. Learning is a stair, not a ramp, suddenly, we climb a step
    - A deep sleep (usually at night) helps to *consolidate* memory
    - *Effort* does not generate immediate reward and generates expectation, but without effort we can hardly get the reward
    - 3 Brains, Triune Brain
        - Reptilian/Primal: instincts
        - Paleo-mammalian/Emotional: emotions
        - Neo-mammalian/Rational: reasoning
- Grade Scheme
    - Assignments: 4, 14%
    - Labs: 8, 14%
        - Start on July 13th
        - MUST attend the lab registered for
        - pre-lab work done prior to the lab
    - Pre-class quizzes: 12, 4%
    - Midterm: 1, 28%
        - Midterm: July 24th
    - Final: 34%
    - Tutorials: 11, 3%
        - Start on July 11th
        - Grade is based on attendance, not submitted
    - Clicker Questions: 3%
        - Marks graded per-question for participation, not correctness
    - Worksheets
        - Posted on Canvas ahead of time and solutions will be posted after the Module is finished
    - Use Overleaf ($\LaTeX$) to complete assignments, and use Gradescope to submit assignments
- What is CPSC 121 about?

- How data is stored in the computer?
- Should we search for all points in the plane for the intersections?
- How can we use the computer to limit the points we are searching?
- How good is our final solution for the problem?
- Is it possible to solve this problem really fast?
- Can computers do anything?

Module 01 - Propositional Logic

- Statement/Proposition: A sentence that is **TRUE** or **FALSE** but not both.
- Negation (NOT)(Single Proposition Operation)
  - not p, $\neg p$
p	$\neg p$
F	T
T	F
  - 
- Conjunction (AND)
  - p and q, $p \wedge q$
p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T
  - 
  - With 2+ inputs: outputs TRUE when all inputs are TRUE and FALSE otherwise
- Disjunction (OR)
  - p or q, $p \vee q$
p	q	$p \vee q$
F	F	F
F	T	T
T	F	T
T	T	T
  - 

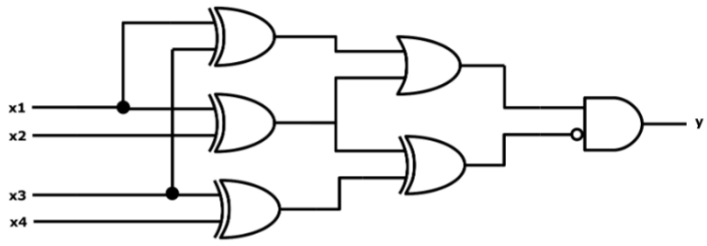
    - With 2+ inputs: outputs TRUE when at least 1 of the inputs are TRUE and FALSE otherwise
- Exclusive Or (XOR)
    - p xor q, $p \oplus q$, $(p \vee q) \wedge \neg(p \wedge q)$
p	q	$p \oplus q$
F	F	F
F	T	T
T	F	T
T	T	F
    - 
    - With 2+ inputs (**AMBIGUOUS**, only use with 2 inputs)
        - outputs TRUE only when 1 and **only** 1 input is TRUE and FALSE otherwise
        - outputs TRUE when the number of TRUE inputs is odd
- Negation Conjunction (NAND)
    - p nand q, $p|q$, $\neg(p \wedge q)$
    - AND Truth Table with opposite output
    - 
- Negation Disjunction (NOR)
    - p nor q, $p \downarrow q$, $\neg(p \vee q)$
    - OR Truth Table with opposite output
    - 
- Tautology: Statement form that is always TRUE
- Contradiction: Statement form that is always FALSE
- Two statements forms are called logically equivalent *iff* they have identical truth values
- De Morgan's Law
    - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
    - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- T/F
    - TRUE: switch on, high voltage, 1, switch open
    - FALSE: switch off, low voltage, 0, switch closed
    - Circuits have binary inputs and produce binary outputs
    - Each logical expression can be represented as a logic gate on a circuit
    - Circuit: A sequence of logic gates connected
    - A circuit can be represented as a proposition and a Truth Table

        - 2 digital logic circuits are equivalent *iff* their input/output tables are identical
- Truth Table
    - A leap year occurs when the year number Y:
        - Is divisible by 4 AND is divisible by 100 AND is divisible by 400
        - OR
        - Is divisible by 4 AND is NOT divisible by 100
    - Statements
        - p: Y is divisible by 4
        - q: Y is divisible by 100
        - r: Y is divisible by 400
    - $p, q, r \rightarrow p \wedge q \wedge r, p \wedge \neg r \rightarrow (p \wedge q \wedge r) \vee (p \wedge \neg r)$
    - k variables: nth column has 2^{k-n} FALSE follow by 2^{k-n} TRUE, repeat 2^{n-1} times
- Circuits to Propositions
    - ...

July 7th - Module 01 (Continued)

Module 01 (Continued)

- Circuits to Propositions
    - Write the operator for the gate that produces the circuit's output
    - Replace the operator's left argument by the expression for the circuit connected to the gate's first input
    - Replace the operator's right argument by the expression for the circuit connected to the gate's second input



    - $y = A \wedge \neg B$
        - $A = C \vee D$
        - $B = E \oplus F$
    - $y = (C \vee D) \wedge \neg (E \oplus F)$
        - $C = x_1 \oplus x_3$
        - $D = x_1 \oplus x_2$
        - $E = x_1 \oplus x_2$
        - $F = x_3 \oplus x_4$
    - $y = ((x_1 \oplus x_3) \vee (x_1 \oplus x_2)) \wedge \neg((x_1 \oplus x_2) \oplus (x_3 \oplus x_4))$

- What does this circuit compute?
    - Analyze the Truth Table
x_1	x_2	x_3	x_4	y
F	F	F	F	F
F	F	F	T	F
F	F	T	F	F
F	F	T	T	T
F	T	F	F	F
F	T	F	T	T
F	T	T	F	T
F	T	T	T	F
T	F	F	F	F
T	F	F	T	T
T	F	T	F	T
T	F	T	T	F
T	T	F	F	T
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F
    - Observe when the output is T, and determine what the inputs have in common
        - y is only T when there are exactly 2 T and 2 F inputs

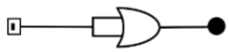
- Light Switches
    - Design a light that changes state whenever **any of the switches that control is flipped**. The ideal solution would work with any number.
    - Approach
        - Understand what we are designing: Use Open/Close to denote each state of the switch
        - Use propositional logic to model the circuit's desired output
        - Start with 1 switch and expand to 2 switches, reach n switches.
    - Most useful **input** of the circuit?
        - Input is what the user can control
        - ✓*The switch is closed.* (Input involves the switches and it should be a binary state)
            - *The switch is flipped.* (Wording is ambiguous and not well-defined)
    - Most useful **output** of the circuit?
        - Output is what the user expects to happen
        - ✓ *The light is on.*
            - *The light changed state.*
    - 1 Switch → 2 Switches
        - There is no connection between **the switch being open** with **the light being on**

- We want the light to change states when the switch changes state. There is no indication of whether or not the switch needs to be on or off.
- The correct solution should **always** do the **same** thing.

Which circuit(s) is/are correct solution(s) for one switch?

a. 

b. 

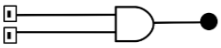
c. 

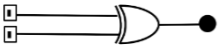
d. Two of a, b, c

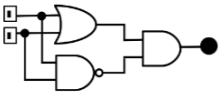
e. All three a, b, c

- - A. p
  - B. $\neg p$
  - C. $p \vee p$
  - All of them will work.

Which circuit(s) is/are correct solution(s) for two switches?

a. 

b. 

c. 

d. Both a and b

e. Both b and c

- - A. $p \wedge q$
p	q	y
F	F	F
F	T	F
T	F	F
T	T	T
  - B. $p \oplus q$
p	q	y
F	F	F
F	T	T
T	F	T
T	T	F
  - C. $(p \vee q) \wedge \neg(p \wedge q)$
$\mathfrak{b}$	$\mathfrak{d}$	$\mathfrak{A}$	$\mathfrak{y}$	$\mathfrak{A}$
F	F	F	F	F
F	T	T	F	T

p	q	$p \vee q$	$p \wedge q$	y
T	F	T	F	T
T	T	T	T	F

- Both B and C work.
- $B \equiv C$
S_1	S_2	$Bulb$
F	F	F
F	T	T
T	F	T
T	T	F

- 3 Switches
S_1	S_2	S_3	$Bulb$
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	F
T	T	F	F
T	T	T	T

- If the 3rd switch is off, the state of the bulb remains the same; if the 3rd switch is on, the state of the bulb changes respectively.
- This is also what the circuit with 2 switches does.
- **Mathematical Induction**

- Propositional Logic From Truth Table
  - Focus on the rows that output TRUE
  - Each row should be translated as a conjunction (AND) between all inputs, when each variable that is FALSE is represented by its negation
  - Connect each proposition found as a disjunction (OR)
  - Leap Year Problem
p	q	r	y
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	**F**	**F**	**T**
T	**F**	**T**	**T**
T	T	F	F
T	**T**	**T**	**T**

    - $(p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge r)$

- Brute Force Algorithm is not accepted, need to search for patterns
    - On T, F, F and T, F, T, it is essentially $p \wedge \neg q$, as r does not change the outcome.
    - On T, T, T, it must be $p \wedge q \wedge r$.
    - Thus, the above brute-forced proposition can be simplified to be $(p \wedge \neg q) \vee (p \wedge q \wedge r)$
a	b	sel	m
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	F
T	T	F	T
T	T	T	T

    - Outputs a if sel is FALSE, otherwise b if sel is TRUE
    - Brute Force:
      $(\neg a \wedge b \wedge sel) \vee (a \wedge \neg b \wedge \neg sel) \vee (a \wedge b \wedge \neg sel) \vee (a \wedge b \wedge sel)$
        - On F, T, T and T, T, T, it is $b \wedge sel$
        - On T, F, F and T, T, F. it is $a \wedge \neg sel$
        - Thus, it must be $(b \wedge sel) \vee (a \wedge \neg sel)$
    - Is $\neg(b \oplus sel) \vee (a \wedge b \wedge \neg sel)$ correct?
        - No, look at F, F, F. This row does not satisfy the proposition.
- Spotting Pattern on Truth Tables
    - A Truth Table that outputs TRUE if the number is even and not zero and FALSE otherwise.
$number$	x_1	x_2	x_3	$even, !0$
0	F	F	F	F
1	F	F	T	F
2	F	T	F	T
3	F	T	T	F
4	T	F	F	T
5	T	F	T	F
6	T	T	F	T
7	T	T	T	F

    - $\neg x_3 \wedge \neg(\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
    - Brute force:
      $(\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$

$$(\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_3)$$
$$\neg x_3 \wedge (x_1 \vee (\neg x_1 \wedge x_2))$$
$$\neg x_3 \wedge ((x_1 \vee \neg x_1) \vee (x_1 \wedge x_2))$$
$$\neg x_3 \wedge (t \wedge (x_1 \vee x_2))$$
$$\neg x_3 \wedge (x_1 \vee x_2)$$

- Approaches
    - Brute Forces
    - Spotting Patterns
    - Karnaugh Map

July 10th - Module 02

Module 02 (Logical Equivalences)

- Reading Review
    - Conditional - $p \to q$ - if p then q
        - p is **hypothesis** (antecedent), q is **conclusion** (consequent)
        - **The sentence says nothing about what will happen if the condition is not met.**
p	q	$p \to q$	$\neg p \vee q$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	T
        - Equivalence: $\neg p \vee q$ (Either **not** p or q)

          Negation: $p \wedge \neg q$

          Contrapositive: $\neg q \to \neg p$

          Converse: $q \to p$

          Inverse: $\neg p \to \neg q$
        - Order of Operation: $\neg$; $\wedge, \vee$; $\to, \leftrightarrow$
    - Biconditional - $p \leftrightarrow q$ - p if, and only if q - p iff q
        - $p \to q$, p is the **sufficient condition** of q
        - $q \to p$, p is the **necessary condition** of q
        - $p \leftrightarrow q$, p is the **necessary, and sufficient condition** of q
p	q	$p \leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

Question 3: Which of the following has the same meaning as $p \rightarrow \sim q$?

a. Anytime that p is true, q must be false.
b. p can not be true unless q is false.
c. q can not be false unless p is true.
d. if p is true then q is false.
e. q and p can never have the same truth value (both true or both false).

- A, B, D
p	q	$\neg q$	$p \rightarrow \neg q$
F	F	T	T
F	T	F	T
T	F	T	T
T	T	F	F

- Examples
  - p = Metallica is playing Vancouver

    q = I am going to a Metallica's concert

    Let me being happy means that the statement is True
  - $p \rightarrow q$: If Metallica plays in Vancouver and I miss, I will be sad;

    If Metallica does not come to Vancouver, then maybe I will travel to see them or maybe I will be home, but I am not sad because I am not missing their concert in Vancouver.
- Logic vs. Everyday English
  - *If it rains, I will bring an umbrella*
    - In English: May assume that if it does not rain, I will not bring an umbrella

      In Logic: I may bring an umbrella even if it does not rains
  - *The match is burning (p) only if there is oxygen in the room (q)*
    - p only if q, is, if p then q
    - If there is oxygen in the room, then the match is burning $\rightarrow$ This allows the match to burn without oxygen
    - If the match is burning then there is oxygen in the room $\rightarrow$ Fire needs oxygen to burn

Assume that *If you rob a bank, then you will go to jail!*

If you rob a bank, will you go to jail?

a. Yes
b. No
c. Maybe

- A, assume the statement is TRUE, when p is TRUE, then q must be TRUE.

Assume that *If you rob a bank, then you will go to jail!*

If you go to jail, have you robbed a bank?

a. Yes
b. No
c. Maybe

- - C, assume the statement is TRUE, when q is TRUE, then p could be either TRUE or FALSE

Assume that *If you rob a bank, then you will go to jail!*

If you rob a bank, can you get away with it (not go to jail)?

a. Yes
b. No
c. Maybe

- - B, assume the statement is TRUE, when p is TRUE, q **cannot** be FALSE

- Logical Equivalence
  - Showing equivalence
    - Draw both Truth Tables and check if they are equal
    - Use rules to change one proposition into the other
      - State the theorem we want to prove
      - Indicate the beginning of the proof by **Proof:**
      - Start with one side and work towards the other, one step at a time, justifying each step
      - Indicate the end of the proof by **QED** (Quod erat demonstrandum) or ■
      - Usually simplify the more complicated proposition, instead of trying to complicate the simpler one
  - Laws
    - Identity Laws: $p \wedge T \equiv p$, $p \vee F \equiv p$
    - Universal Bounds Laws: $p \wedge F \equiv F$, $p \vee T \equiv T$
    - Idempotent Laws: $p \wedge p \equiv p$, $p \vee p \equiv p$
    - Commutative Laws: $p \wedge q \equiv q \wedge p$, $p \vee q \equiv q \vee p$
    - Associative Laws: $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$, $p \vee (q \vee r) \equiv (p \vee q) \vee r$
    - Distributive Laws: $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$, $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
    - Absorption Laws: $p \vee (p \wedge q) \equiv p$, $p \wedge (p \vee q) \equiv p$
    - Negation Laws: $p \wedge \neg p \equiv F$, $p \vee \neg p \equiv T$
    - Double Negation Law: $\neg\neg p \equiv p$
    - De Morgan's Laws: $\neg(p \wedge q) \equiv \neg p \vee \neg q$, $\neg(p \vee q) \equiv \neg p \wedge \neg q$
    - Definition of XOR: $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$
    - Definition of IMP: $p \to q \equiv \neg p \vee q$
    - Contrapositive Law: $p \to q \equiv \neg q \to \neg p$

Prove that $(\sim a \wedge b) \vee a \equiv a \vee b$.

Proof:

$(\sim a \wedge b) \vee a$	$\equiv a \vee (\sim a \wedge b)$	Commutative Law
	$\equiv (a \vee \sim a) \wedge (a \vee b)$	Distributive Law
	$\equiv$?????	
	$\equiv a \vee b$	Identity Law

■

What is the expression missing?

a. $(a \vee b)$
b. $F \wedge (a \vee b)$
c. $a \wedge (a \vee b)$
d. Something else
e. Not enough info to tell

- D, $T \wedge (a \vee b)$

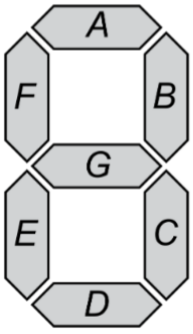
- Proofs
  - Must always apply one rule at a time, never combining steps, except in the following scenarios
    - De Morgan's Law and Double Negation Law
      - $\neg(\neg p \vee \neg q) \equiv p \wedge q$, by De Morgan's Law
    - Associative and Commutative Laws
      - $(p \vee q) \vee (r \vee s) \equiv (r \vee q) \vee (p \vee s)$, by the Associative and Commutative Laws
    - Same law applied to 2 completely independent expressions
      - $\neg(\neg p) \vee (q \wedge \neg(\neg p)) \equiv p \vee (q \wedge p)$, by Double Negation Law
  - Example: $(\neg sel \wedge a) \vee (sel \wedge b) \equiv (\neg sel \wedge a) \vee (sel \wedge b) \vee (a \wedge b)$
    - **Proof:**

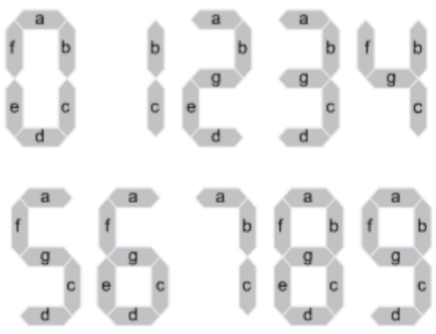
$(\sim sel \wedge a) \vee (sel \wedge b)$	
$((\sim sel \wedge a) \vee sel) \wedge ((\sim sel \wedge a) \vee b)$	by DIST
$(\sim sel \vee sel) \wedge (sel \vee a) \wedge (\sim sel \vee b) \wedge (a \vee b)$	by DIST
$T \wedge (sel \vee a) \wedge (\sim sel \vee b) \wedge (a \vee b)$	by NEG
$(sel \vee a) \wedge (\sim sel \vee b) \wedge (a \vee b)$	by I
$(sel \vee a) \wedge ((\sim sel \wedge a) \vee b)$	by DIST
$((sel \vee a) \wedge (\sim sel \wedge a)) \vee ((sel \vee a) \wedge b)$	by DIST
$((sel \vee a) \wedge a \wedge \sim sel) \vee ((sel \vee a) \wedge b)$	by COMM/ASS
$(a \wedge \sim sel) \vee ((sel \vee a) \wedge b)$	by ABS
$(a \wedge \sim sel) \vee ((sel \wedge b) \vee (a \wedge b))$	by DIST
$(\sim sel \wedge a) \vee (sel \wedge b) \vee (a \wedge b)$	by COMM/ASS

■

    - Note to self: there is absolutely no need to expand $((\neg sel \wedge a) \vee b)$ from Step 1 to Step 5

- LED segment display
  - Design a circuit that displays the numbers 0 to 9 using 7 LEDs in the shape illustrated as the following:





INPUT: NUMBER	x_1	x_2	x_3	x_4
0	F	F	F	F
1	F	F	F	T
2	F	F	T	F
3	F	F	T	T
4	F	T	F	F
5	F	T	F	T
6	F	T	T	F
7	F	T	T	T
8	T	F	F	F
9	T	F	F	T

- How many outputs (lights) are there?
    - a, b, c, d, e, f, g: one segment one output
- For example, for segment B, the output for each number input should be $T, T, T, T, T, F, F, T, T, T$
    - By negating the FALSE rows, the brute-forced logical proposition:
      $B = \neg((\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4))$
    - $$\begin{aligned} \therefore B &= \neg((\neg x_1 \wedge x_2) \wedge ((\neg x_3 \wedge x_4) \vee (x_3 \wedge \neg x_4))) \\ &\rightarrow \neg(\neg x_1 \wedge x_2 \wedge (x_3 \oplus x_4)) \end{aligned}$$

July 12th - Module 03

Module 03 (Number Representation)

- Reading Review
    - Binary
        - Representation of an integer as the sum of products in form of $d \cdot 2^n$, where d is either 0 or 1
        - Binary to Decimal: Multiply each digit with their corresponding power of 2
            - $(11011)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (27)_{10}$
        - Decimal to Binary: Find the largest power of 2 less than the given number
            - $(209)_{10} = 128 + 64 + 16 + 1 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 +$
        - Addition
            - Adding $1 + 1$ is 0 with a carry of 1
        - Subtraction
            - Borrowing 1 from 10, the remaining is 1
    - 2's complement
        - The 2-complement of an integer a in n bit representation is $2^n - a$.
        - Writing out the n bit representation for the 2-complement of a
            - Write out the n bit binary representation of a

        - Flip the digits
        - Add 1. **ALWAYS +1**
    - 8-bit 2's complement of 27
        - $(27)_{10} = (00011011)_2$

          $(-27)_{10} = (11100101)_2$
    - From 2's complement to decimal
        - Convert back to positive and convert to decimal
        - $(11010110)_2 \rightarrow 00101001 \rightarrow 00101001 + 1 \rightarrow 00101010 \rightarrow (-42)_{10}$
    - Addition
        - Convert both integers to their n-bit representations (negative integers by using two's complement)
        - Add the resulting integers using ordinary binary addition
        - Truncate any leading 1 on the 2^n-th position
        - Convert the result back to decimal form
- Hexadecimal
    - $d \cdot 16^n$, d is an integer from 0 to 15, with 10 to 15 represented as A to F
    - Hexadecimal to Decimal
        - Multiply each digit for its correspondence power of 16
    - Hexadecimal to Binary
        - Convert each digit to binary and concatenate the results, drop the leading 0s afterwards.
    - Binary to Hexadecimal
        - Group the digits in **sets of four** from right to left, adding leading 0s if necessary, and convert each set
- Unsigned and Signed Integer
    -

number	a	b	c	d
0	F	F	F	F
1	F	F	F	T
2	F	F	T	F
3	F	F	T	T
4	F	T	F	F
5	F	T	F	T
6	F	T	T	F
7	F	T	T	T
8	T	F	F	F
9	T	F	F	T

number	b_3	b_2	b_1	b_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

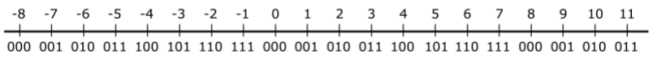
    - A sequence of bits is intrinsically neither signed nor unsigned nor anything else, we give them meaning
        - An unsigned integer is one we have decided will only represent integer values that are > 0
        - A signed integer is one we have decided can represent either a positive value or a negative one
        - Other representations are real numbers or characters
    - $\overline{(b_{n-1}b_{n-2}\ldots b_2b_1b_0)_2} = (\sum_{i=0}^{n-1} b_i \cdot 2^i)_{10}$
    - We can use any base to represent a number

        - How can we represent signed values?
            - Sign-magnitude
            - One's complement
            - Two's complement (ALWAYS USED IN THIS COURSE)
    - Why do we use two's complement?

      Because it makes subtraction easier: $x - y = x + (-y)$

      For **3-bit** integers, what is 111 + 1?

      a. 110
      b. 111
      c. 1000
      d. 000
      e. Error: we can not add these two values.

    -
        - D, the computer has no memory to store the leftmost digit which has a 1.
    - 3-bit representation number line
        -
        

        - 0_{10} to 7_{10} is represented by 3-bit unsigned integers
        - $(-4)_2$ to 3_2 is represented by 3-bit signed integers
    - Unsigned integer overflow and underflow
        - Overflow happens when you need an extra bit to represent the result
        - Underflow happens when the addition of 2 negative operands produces positive sum
    - Decimal to Binary
        - Divide by 2 until reaching 0 writing the remainder, write down the remainders from right to left
        - For negative decimal numbers, find the binary representation of the absolute value and do the 2 complement
    - We should not negate $(100...000)_2$ in a n-bit binary representation, as the two's complement of it is still $(100...000)_2$.
- Modular Arithmetic
    - Modular: Given an integer m, we partition integers based on their remainder after division by m.
        - For $m = 5$, we have classes $\{0, 1, 2, 3, 4\}$. e.g. $200 \equiv 0 \pmod 5$
        - e.g. $[1] = \{\dots, -14, -9, -4, 1, 6, 11, \dots\}$
    - There are different definitions for the modular of a negative number
        - C: $-14\%5 = -4$
        - Python: $-14\%5 = 1$
    - Modular operation
        - $x \bmod m = r$
        - r would be the remainder class

- Equivalence
    - $x \equiv y \pmod{m}$
    - x and y belongs to the same remainder class for m

What is 57 mod 8?

a. 1

b. 3

c. 5

d. 7

- A, as $57 = 8 \cdot 7 + 1$

Suppose that $x \equiv 34 \bmod 6$. Which are possible values for x?

a. 4, 17 and 28.

b. 12, 28 and 38.

c. 36, 72 and 216.

d. 10, 16 and 52.

- D, as $34 \bmod 6 = 4$, find the group that has remainder of 4 when divided by 6.

- Fundamental Theorem of Modular Arithmetic
    - If $a \equiv c \pmod{m}$ and $b \equiv d \pmod{m}$, then $ab \equiv cd \pmod{m}$
    - e.g. $(10 \cdot 15 \cdot 23) \bmod 7 = (10 \bmod 7) \cdot (15 \bmod 7) \cdot (23 \bmod 7)$
- Why does two's complement work?
    - When we flip to get to the negative group, we add 1 as 0 does not have a negative equivalent
    - $-x$ is the same as getting $2^n - x$ in a n-bit representation
        - x = sum of some combination of the powers of 2
        - $\bar{x}$ = flipping x is equal to the sum of the rest of the powers of 2
        - $x + \bar{x}$ = the sum of all powers of $2 = \sum_{i=0}^{k-1} 2^i = 2^k - 1$
        - $-x = \bar{x} + 1 = (2^n - 1 - x) + 1 = 2^n - x$
        - $-x \equiv 2^n - x \pmod{2^n}$
- Characters
    - People create arbitrary mappings for character representations
        - EBCDIC: earliest, now used only for IBM mainframes
        - ASCII: American Standard Code for Information Interchange, 7-bit per character, sufficient for upper/lowercase, digits, punctuation and a few special characters
            - Non-printable characters have their ASCII as well
        - UNICODE: 16 or 32 bits, extended ASCII for languages other than English

What does the 8-bit binary value 11111000 represent?

a. -8
b. The character °
c. 248
d. More than one of the above.
e. None of the above.

- D, 248 unsigned, -8 signed, ° Extended 8-bit ASCII

- Real Numbers
  - ...

July 12th - Module 03 (Continued) & Module 04

Module 03 (Continued) - Real Numbers

- ...
- Decimal to Binary
  - Converting a decimal number to binary, we multiply it by 2, take the value before the decimal point, and continue until we reach 0
- Scientific Notation
  - Expressing numbers that are too big or too small in the form $m \cdot b^e$, where b gives the base (usually 10 for decimal numbers)
  - Decimal Representation: $1724 = 1.724 \times 10^3$
  - Binary Representation: $(1724)_{10} = (11010111100)_2 = 1.1010111100 \times 2^{1010}$
  - Store only *mantissa* and the *exponent*
    - *mantissa*: 二进制小数点后的部分，24 bits for float, 53 bits for double.
- C/Java call this data float and Racket calls this inexact numbers
  - Floating point numbers involvements make computations imprecise
  - Computer cannot store $\frac{1}{3}$, but a number very close to $\frac{1}{3}$
  - More computations performed, the further away from the "real" value we are

```
(* (sqrt 2) (sqrt 2))
```

```
import math

math.sqrt(2) * math.sqrt(2)
```

Racket:

```
(define (addfractions x)
   (if (= x 1.0)
      0
   (+ 1 (addfractions (+ x 0.1)))))
```

Python:

```
def addfractions(x):
   r = 0
   while x != 1.0:
      x = x + 0.1
      r = r + 1
   return r
```

What value will `addfractions()` return with x=0?

a. 10
b. 11
c. Less than 10
d. More than 11
e. No value will be printed

x	r
0	0
0.1	1
0.2	2
0.3	3
0.4	4
0.5	5
0.6	6
0.7	7
0.8	8
0.9	9
1.0	9

  - E. The computer cannot be accurate enough to equal 1.0, thus it is an infinite loop. No value will be printed.

- Binary to Hexadecimal
  - Hexadecimals are represented by the 0x symbol
  - $(101010)_2 = 0x2A$
- Hexadecimal
  - Colors (0xRRGGBB)
    - Red: 0xFF0000
    - Green: 0x00FF00
    - Blue: 0x0000FF
    - Yellow: 0xFFFF00
    - Lavender: 0xB57EDC
    - Black: 0x000000
  - Instructions
    - Computers use bytes (8-bit) as basic data
    - Y86 example of adding 2 values
      - Human readable-form: addl %ebx %ecx
      - Binary form: 01100000 0011 0001
      - Decimal form: 24625 → information hard to see
      - Hexadecimal: 0x60 3 1

Module 04 - Propositional Logic Proofs

- Reading Review
    - Argument: sequence of statements ending in a conclusion; It is valid if assuming the premises to be True that the conclusion is True.

$$p \to q$$
$$p$$
$$\therefore q$$

        - Argument form: abstract representation with variables
        - All statements but the final statement are called premises
        - The final statement or statement form is called the conclusion
        - The symbol $\therefore$ means **therefore** and is placed before the conclusion
    - Validity
        - An argument's validity can be checked via the Truth Table, where it is valid if for every row that the premises are True, the corresponding conclusion is also True, otherwise, the argument is invalid
    - Rules of Inference
        - Modus Ponens (affirming the antecedent)

$$p \to q$$
$$p$$
$$\therefore q$$

        - Modus Tollens (denying the consequent)

$$p \to q$$
$$\neg q$$
$$\therefore \neg p$$

    - Fallacy
        - Converse Error

$$p \to q$$
$$q$$
$$\therefore p$$

        - Inverse Error

$$p \to q$$
$$\neg p$$
$$\therefore \neg q$$

    - Quiz 4, Q4
        - $$p \wedge \neg p$$
$$\therefore q$$
        -

p	q	$p \wedge \neg p$
F	F	F
F	T	F
T	F	F
T	T	F

        - The argument is valid, as there are no cases that the premise is True but the conclusion is False
- Proof and their Meaning
    - Proof: a rigorous formal argument that demonstrates the truth of a proposition, given the truth of the proof's premise
        - Used to convince the truth of a conditional proposition
        - Well justify each step
        - Understand how each step relates to previous steps

Suppose that you proved this:
Premise 1
...
Premise n
$\therefore$ Conclusion

Does it mean:

a. Premises 1 to n are true.
b. Conclusion is true.
c. Premises 1 to n are not a contradiction.
d. Conclusion isn't a contradiction.
e. None of the above.

- - E. We prove a relation between the premises and the conclusion, but nothing can be said about any of them.

What does this argument mean?
Premise 1
...
Premise n
$\therefore$ Conclusion

a. Premise 1 $\wedge$ ... $\wedge$ Premise n $\wedge$ Conclusion
b. Premise 1 $\vee$ ... $\vee$ Premise n $\vee$ Conclusion
c. (Premise 1 $\wedge$ ... $\wedge$ Premise n) $\rightarrow$ Conclusion
d. (Premise 1 $\wedge$ ... $\wedge$ Premise n) $\leftrightarrow$ Conclusion
e. None of the above.

- - C. If all the premises are True, then the conclusion must be True

Let's consider invalid the rule:

$p \rightarrow q$

q

$\therefore p$

What can we say about the truth value of p?

a. p is true

b. p is false

c. p might be either true or false

d. p can be neither true nor false

- C. If q is True, nothing can be said about p.

- Examples using Predicates
  - $6|n$
    $\therefore 3|n$
    - Premises are True: Are all numbers divisible by 6? No!
    - Conclusion are True: All all numbers divisible by 3? No!
    - All Premises being True, led to Conclusion being True: Yes!
    - Premises being False: If a number is not divisible by 6, is it divisible by 3? Maybe...
n	$6\|n$	$3\|n$
9	F	T
12	T	T
20	F	F
N/A	T	F
  - $5|n$
    $\therefore 2|n$
    - All Premises being True, led to Conclusion being True: does the number that satisfy the premise always is divisible by 2? NO!
n	$5\|n$	$2\|n$
8	F	T
10	T	T
7	F	F
15	T	F
    - One counterexample is enough to disprove it.
  - $0|n$
    $\therefore 10|n$
    - All premises being True, led to conclusion being True: This never happens
    - STILL VALID ARGUMENT!

- Propositional Logic Proofs

        - A sequence of propositions, where each proposition is one of either **premise** or the **result of applying a logical equivalence or a rule of inference** to one or more earlier propositions
        - Simpler than the more free-form proofs, only a limited number of choices at each step
- More rules of inference
    - Generalization
    - Specialization
    - Conjunction
    - Elimination
    - Transitivity
    - Resolution
    - Contradiction (Special case of modus tollens)
- Proving an argument is Valid

$$1.p \vee q$$
$$2.\neg p$$
$$3.q \rightarrow \neg r$$
$$4.r \vee (s \wedge t \wedge u)$$

- $$5.q$$
  $$6.\neg r$$
  $$7.s \wedge t \wedge u$$
  $$8.u$$

  $$\therefore u$$

    - 5 - Elimination from 1 and 2

      6 - Modus Ponens from 5 and 3

      7 - Elimination from 6 and 4

      8 - Specialization from 7
    - ...

July 17 - Module 04 (Continued) & Module 05

Module 04 (Continued)

- The Fire-Trolls Problem
    - *Premise 1*: If dragons are too scaly to portray dragons, then trolls must be too smelly to play trolls, and vice versa

      Premise 2: And yet, if the fire-trolls are correct, dragons are too scaly to portray dragons and yet trolls are not too smelly to play trolls

      Conclusion: Therefore, the fire-trolls are incorrect, and dragons are not too scaly to portray dragons

Fire-trolls: which definitions should we use?

a. d = dragons, t = trolls, sc = scaly, sm = smelly, ft = fire-trolls, c = correct

b. d = dragons are too scaly, t = trolls are too smelly, pd = dragons portray dragons, pt = trolls portray trolls, o = fire-trolls are correct

c. d = dragons are too scaly to portray dragons, t = trolls are too smelly to portray trolls, o = fire-trolls are correct

d. None of these, but another set of definitions works well.

e. None of these, and this problem cannot be modeled well with propositional logic.

- - C. Each variable needs to be either True or False.

Fire-trolls: do the two premises contradict each other (that is, is $p1 \wedge p2 \equiv F$)?

a. Yes

b. No

c. Not enough information to tell

- - $p_1 \equiv d \leftrightarrow t$, $p_2 \equiv o \rightarrow (d \wedge \neg t)$, $c \equiv \neg o \wedge \neg d$
  - B. Fire-trolls can be incorrect and p_2 will still be True.
- What can we prove?
  - **Can** prove fire-trolls are wrong
  - **Cannot** prove that dragons are not too scaly to play dragons
  - **Cannot** prove that trolls are not too smelly to play trolls
  - **Can** prove this argument is not valid
-

d	t	o	$d \leftrightarrow t$	$o \rightarrow (d \wedge \neg t)$	$\neg o \wedge \neg d$
F	F	F	T	T	T
F	F	T	T	F	F
F	T	F	F	T	T
F	T	T	F	F	F
T	F	F	F	T	F
T	F	T	F	T	F
T	T	F	T	T	F
T	T	T	T	F	F

$$d \leftrightarrow t$$
$$o \rightarrow (d \wedge \neg t)$$
$$\therefore \neg o$$

-
$$\begin{array}{cc} 1. d \leftrightarrow t & \\ 2. o \rightarrow (d \wedge \neg t) & \\ 3. \neg(d \oplus t), & BICON(1) \\ 4. \neg((d \vee t) \wedge \neg(d \wedge t)), & XOR(3) \\ 5. \neg(d \vee t) \vee (d \wedge t), & DM(4) \\ 6. \neg(d \vee t), & SPEC(5) \\ 7. \neg o, & MT(2,6) \\ QED & \end{array}$$

- Proving Contradictions

$$
\begin{array}{l}
p \\
p \to r \\
p \to \neg s \\
p \to (q \vee \neg r) \\
\neg q \vee s
\end{array}
$$

$$
\begin{array}{ll}
r, & MP(1,2) \\
q \vee \neg r, & MP(1,4) \\
q, & ELIM(6,7) \\
s, & ELIM(5,8)
\end{array}
$$

$$\neg s, \quad MP(1,3)$$

- Since we prove both s and $\neg s$ to be True, this is a contradiction (there is no row where all premises are true). This argument is still **valid**, since there is not a case in which all premises are True, and the conclusion is False.

- Proving an Argument is Invalid

$$
\begin{array}{c}
p \vee (q \wedge r) \\
\neg (p \wedge q) \\
\therefore r
\end{array}
$$

- Assume $p \equiv T, q \equiv F, r \equiv F$:

  $p_1 : p \vee (q \wedge r) = T \vee (F \wedge F) = T$

  $p_2 : \neg (p \wedge q) = \neg (T \wedge F) = T$

  $c : r = F$

  Why not use logical equivalences to prove that the conclusions follow from the premises?

  a. No reason; logical equivalences are enough.
  b. Logical equivalences scale poorly to large problems.
  c. Rules of inference can prove theorems that cannot be proven with logical equivalences.
  d. Logical equivalences require insight to use, while rules of inference can be applied mechanically.

-
  - C. Rules of inference show more complex relations between premises, so we can prove theorems

Module 05 - Set and Predicate Logic

- Reading Review
  - Predicates: Sentences cannot be translated into proposition statements because it depends on the value of its variables that can be more than True or False
    - $P(x) : x^2 > x, Q(x, y) : x + y \geq 10$
    - $P(2)$ is True, $P(1)$ is False
  - Truth Set: If $P(x)$ is a predicate and x has domain D, the truth set of $P(x)$ is the set of all elements of D that make $P(x)$ True
    - $x \in D | P(x)$
  - The universal quantifier: for all, $\forall$
  - The existential quantifier: there exists, $\exists$

- The negative of a universal statement is logically equivalent to an existential statement
    - $\neg(\forall x \in D, Q(x)) \equiv \exists x \in D, \neg Q(x)$
- The negative of a existential statement is logically equivalent to a universal statement
    - $\neg(\exists x \in D, Q(x)) \equiv \forall x \in D, \neg Q(x)$
- Negation of a Universal Conditional Statement:
    - $\neg(\forall x, P(x) \to Q(x)) \equiv \exists x, P(x) \wedge \neg Q(x)$
- Consider a statement of the form: $\forall x \in D, P(x) \to Q(x)$
    - Contrapositive: $\forall x \in D, \sim Q(x) \to \sim P(x)$.
    - Converse: $\forall x \in D, Q(x) \to P(x)$.
    - Inverse: $\forall x \in D, \sim P(x) \to \sim Q(x)$.

    - r is a **sufficient** condition for s: $\forall x, r(x) \to s(x)$
    - r is a **necessary** condition for s: $\forall x, s(x) \to r(x)$
    - $\forall x, r(x) \leftrightarrow s(x) \equiv \forall x, r(x) \to s(x) \wedge \forall x, s(x) \to r(x)$
- Quiz 05 Q 4
    - E. Prime$(x) \vee$ Even(x) should be True **for all** numbers for the statement to be True, so one counterexample is that 9 is odd and it is not a prime, thus the statement is False.

- Sets
    - Set: a set is a collection of definite and separate objects
    - $S = \{1, 2, 3\}, T = \{k, p, m, h\}$
    - S, T are **sets**, $1, 2, 3, k, p, m, h$ are **elements**
    - $a \in S$: a is an element of S; $a \notin S$: a is not an element of S
    - $A \subseteq B$: means all elements in A are in B, $\forall x, x \in A \to x \in B$;

      $A \nsubseteq B$: means there are elements in A that are not in B,
      $\exists x, x \in A \wedge x \notin B$.
    - $A \subset B$: means that A is a **proper subset** of B, but there is at least 1 element in B that is not in A.

      $A = B$: means that every element of A is in B and every element of B is in A.
    - $A \cup B$: $\{x \in U | x \in A \vee x \in B\}$

      $A \cap B$: $\{x \in U | x \in A \wedge x \in B\}$
    - $B - A = \{x \in U | x \in B \wedge x \notin A\}$

      $A^c = \{x \in U | x \notin A\}$
    - $\in$ vs. $\subseteq$
        - $\in$: takes in an **element** x and a **set** S.
        - $\subseteq$: takes in **two sets** S and T.
        - An element can be a set
        - ...

July 19th - Module 05 (Continued)

Module 05 (Continued)

- Set
    - $\in$ vs. $\subseteq$
        - Suppose $A = \{1, \{2\}\}$
            - $1 \in A$
            - $\{1\} \subseteq A$
            - $1 \subseteq A$, **NOT WELL DEFINED**
            - $\{1\} \in A$, **NOT WELL DEFINED**
            - $2 \notin A$
            - $\{2\} \nsubseteq A \rightarrow \{\{2\}\} \subseteq A$
            - $2 \subseteq A$, **NOT WELL DEFINED**
            - $\{2\} \in A$

Suppose $A = \{1, \{2\}\}$.

Is $\{\{2\}\} \subseteq A$?

a. True
b. False
c. The statement is ill-defined.

        - A
    - Common Sets
        - $\mathbb{Z}, \mathbb{Z}^+, \mathbb{Z}^*$; Integers, Positive Integers, Non-zero integers
        - $\mathbb{Q}, \bar{\mathbb{Q}}$; Rational numbers, Irrational numbers
        - $\mathbb{R}$; Real numbers
        - $\mathbb{N}_0, \mathbb{N}_1$; Natural numbers with 0, Natural numbers without 0.
    - Empty set - $\emptyset$ or $\varnothing$
        - A set with no elements
        - $\forall A, \emptyset \subseteq A$
        - $\forall A, A \cup \emptyset = A$
        - $\forall A, A \cap \emptyset = \emptyset$
        - $\forall x \in \emptyset, P(x)$: For every element of empty set, any property P is True (Vacuous Truth)
        - $\neg \exists x \in \emptyset, P(x)$: There is no element of empty set for which property P is True
    - Universal and Power Set
        - Universal Set: U, is the set containing all elements and of which all other sets are subsets

- Power set: Given a set A, the power set of A, denoted $P(A)$, is the set of all subsets of A.
    - $P(\{x, y\}) = \{\emptyset, \{x\}, \{y\}, \{x, y\}\}$

- Magic Box
    - $A = \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18\}$

      $B = \{1, 3, 6, 9, 12, 15, 18\}$
    - Is 0 in A: False, Is 0 is B: False; Is 0 in $A \cup B$: False, Is 0 in $A \cap B$: False; etc.
    - $\cup \leftrightarrow \vee$; $\cap \leftrightarrow \wedge$
- Predicates vs. Propositions
    - Propositional logic cannot model problems in which the variables on the statements can have different values than True or False
    - Predicate can work with elements of a set, which is a **domain**
    - Proposition: $(p \wedge q) \rightarrow r$

      Predicate: $\forall x \in \mathbb{Z}, (P(x) \wedge Q(x)) \rightarrow R(x)$
    - Predicate logics is more advantageous
- Quantifiers Scope
    - Quantifier has a variable attached to it, and its domain; variable can only be used in places which it considered defined in the statement
        - $\forall x \in D, (\exists y \in D, Q(x, y) \rightarrow \forall z \in F, R(y, z)) \vee P(x)$
- Negation Scope
    - Original Statement: All CPSC instructors are nerds

      Negation first version: **Not** all CPSC instructors are nerds

      Negation second version: *There is some* CPSC instructor who is not a nerd
    - The quantifier type changed from **all** to **some** and the property also changed from **nerd** to **not a nerd**.
    - C: all CPSC instructors

      $N(x) : x$ is nerd

      Original statement: $\forall x \in C, N(x)$

      Negation 1st version: $\neg \forall x \in C, N(x)$

      Negation 2nd version: $\exists x \in C, \neg N(x)$

      What can we do with the negation in:

      $$\sim \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \rightarrow f(n) \leq cg(n)$$

      a. It cannot be moved inward.
      b. It can only move across one quantifier because the generalized De Morgans law can only handle one quantifier.
      c. It can only be moved across all three quantifiers because a negation can't appear between quantifiers.
      d. It could be moved across one, two or all three quantifiers.
      e. None of the above.
    -
        - D. The negation can move inward layer by layer. The negation does not change the domain.
        - There is no positive real number c, that there is a natural number n_0, that for all natural numbers n, if $n \geq n_0$, then $f(n) \leq cg(n)$

- $\wedge$ vs. $\rightarrow$
    - C: CPSC instructors

      $N(x)$: x is a nerd

      $CPSC121(x)$: x teaches CPSC 121
    - All CPSC instructors are nerds an teach 121:
      $\forall x \in C, CPSC121(x) \wedge N(x)$
        - There cannot be an instructor that does not teach CPSC 121
        - There cannot be an instructor that is not a nerd
    - All CPSC 121 instructors are nerd: $\forall x \in C, CPSC121(x) \rightarrow N(x)$
        - There can be a instructor that does not teach CPSC 121 and that is a nerd
        - There cannot be a CPSC 121 instructor who is not a nerd
    - Some CPSC 121 instructors are nerds: $\exists x \in C, CPSC121(x) \wedge N(x)$
        - There has to be at least one instructor that is a nerd and teaches CPSC 121
    - There exists some CPSC instructor, if they teach CPSC 121, then they are nerds: $\exists x \in C, CPSC121(x) \rightarrow N(x)$
        - There is no need to exist one instructor that is nerd and teaches 121.

      What is the truth value of the statement:

      $$\exists x \in \mathbb{Z}, x * x = y$$

      a. True because (for example) $5 \cdot 5 = 25$.
      b. True because for every y, we know that $y = \sqrt{y} \cdot \sqrt{y}$.
      c. False, because of counterexamples like no integer multiplied by itself equals 3.
      d. It depends on y, but given a value for y, we could calculate a truth value.
      e. None of the above.
    -
        - D. We do not know the domain of y, for example with $y \in \mathbb{Z}^-$, this will never be True.
- Unbound Variable
    - $PerfectSquare(y) : \exists x \in \mathbb{Z}, x \cdot x = y$
    - Bounded Variable: It is within the scope of a quantifier, x
    - Unbounded variable: It is **not** within the scope of a quantifier, y
    - $PerfectSquare(25)$ is True

      $PerfectSquare(27)$ is False

      $\exists y \in \mathbb{Z}, PerfectSquare(y)$ is True

      $\forall y \in Z, PerfectSquare(y)$ is False

Which variables are unbound?

$$\forall i \in \mathbb{Z}^+, (i \geq n) \leftrightarrow \sim \exists v \in \mathbb{Z}^+, HasValue(l, i, v)$$

a. i and v.
b. l and n.
c. n and v.
d. i and n.
e. None of these are correct.

- - B.

Consider the following two statements:

- A rich person gave \$10,000 to every CPSC 121 student.
- Every CPSC 121 student received \$10,000 from a rich person.

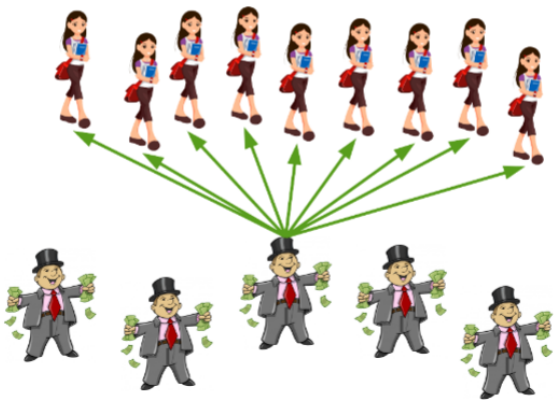
Do they mean the same thing?

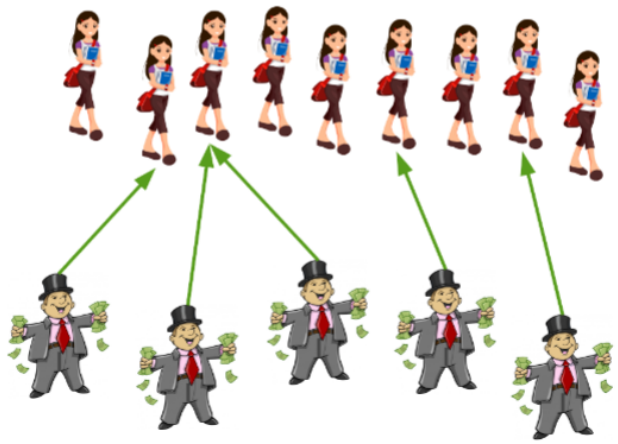
a. Yes.
b. No.
c. Its impossible to tell

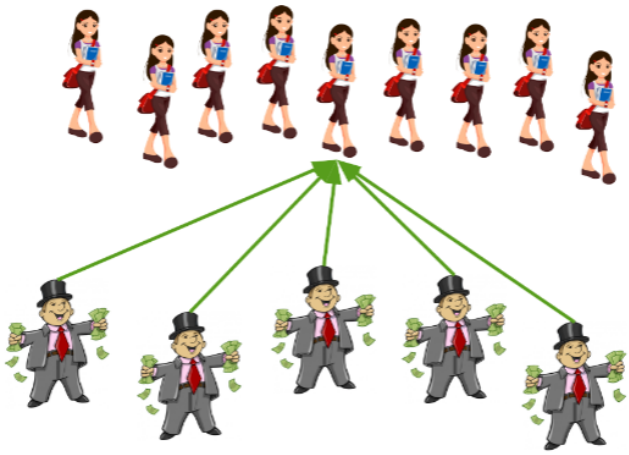
- - B. The first, a single person gave money to all students, emphasizing the rich person ($\exists r \forall s$); The second, all students received money from some rich person, emphasizing all students ($\forall s \exists r$).

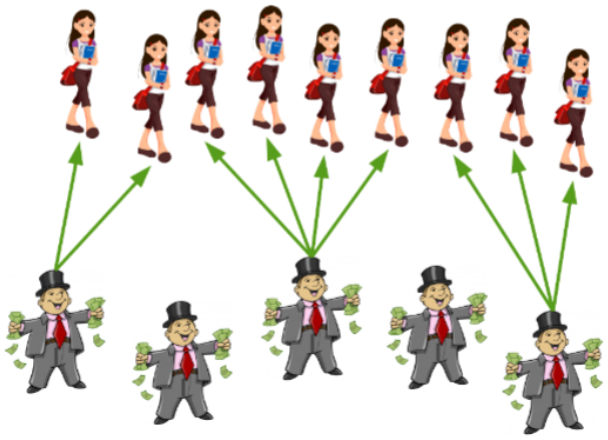
- Order of Quantifiers
  - R : the set of rich people

    S: the set of students

    $Gave(x, y)$: x gave $\$10,000$ to y
  - E_1: A rich person gave $\$10,000$ to every student
    - $\exists r \in R, \forall s \in S, Gave(r, s)$
    - 
  - E_2: Every rich person gave $\$10,000$ to a student
    - $\forall r \in R, \exists s \in S, Gave(r, s)$

- 

- E_3: A student received $\$10,000$ from every rich person
  - $\exists s \in S, \forall r \in R, Gave(r, s)$
  - 

- E_4: Every student received $\$10,000$ from a rich person
  - $\forall s \in S, \exists r \in R, Gave(r, s)$
  - 

- The order of quantifiers does matter as $\forall x, \exists y, P(x, y)$ **is not** equivalent to $\exists y, \forall x, P(x, y)$
  - $\forall x \in \mathbb{Z}^+, \exists y \in \mathbb{Z}^+, y = x \cdot x$
    - Each x can have its own y
  - $\exists y \in \mathbb{Z}^+, \forall x \in \mathbb{Z}^+, y = x \cdot x$
    - There is no y that will work for all x
  - $\forall x \in \mathbb{Z}^+, \exists y \in \mathbb{Z}, y = x \cdot 0$ vs. $\exists y \in \mathbb{Z}, \forall x \in \mathbb{Z}^+, y = x \cdot 0$

- Both are True, but mean different things

What can we say about these two statements?

$$A = \forall x \in \mathbb{Z}^*, \exists y \in \mathbb{R}, x \cdot y = 1$$

$$B = \exists y \in \mathbb{R}, \forall x \in \mathbb{Z}^*, x \cdot y = 1$$

a. A is True and B is False
b. A is False and B is True
c. A is True and B is True
d. A is False and B is False

-
  - A. A is True if we always choose $y = \frac{1}{x}$; B is False since there is no y that will this True to all x.

July 21 - Module 06

Module 06

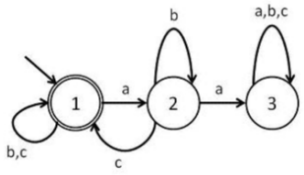
- Reading Review
  - Alphabet Σ: A finite set of characters

    String: A finite set of elements of the alphabet

    Language: A set of string over the alphabet
  - Operations in an Alphabet
    - Σ^n: The set of all strings over Σ with length n
    - Σ^+: ... have length of at least 1
    - Σ^*: The set of all strings over Σ
  - Operations in Languages:
    - L_1L_2: Concatenation
    - $L_1 \cup L_2$: Set of all strings in L_1 and L_2
    - L^*: Set of strings that are a concatenation of any finite number of of strings in L
  - Regular Expression
    - Empty String
    - Each element in Σ
    - rs
    - $r|s$
    - $(rs)^*$
  - Combinatorial Circuit vs. Sequential Circuit
  - Finite-State Automaton
    - An idealized machine that embodies the essential idea of a sequential circuit
    - Unlabeled arrow: initial state
    - Double circle: accepting state

        - Arrows: indicate what happens when a particular input is made
        - Objects
            - I: input alphabet
            - S: set of states
            - s_0: initial state
            - Set of accepting states
            - $N : S \times I \to S$ (next-state function)
        - Language accepted by an automaton
            - A: FA
            - I: set of input symbols
            - $w \in I^*$
            - If w reaches an accepting state of A after all symbols are input, w is accepted by A.
            - $L(A)$: language accepted by A, set of all strings accepted by A
    - DFA vs. NFA

      Question 7: Which of these correctly describes the language accepted by this DFA? In all cases, we use the phrase "any string" to refer to strings drawn from the alphabet of this DFA: a, b, and c.

      Answer is: **None of these correctly describes the language accepted by this DFA.**

      Any string with at least one letter in which every occurrence of the letter a is followed by an occurrence of the letter c that comes before the next occurrence of the letter a.

        - λ is accepted, thus this option is incorrect.
- Regex
    - An expression that specifies a search pattern in text
    - .
        - The period symbol matches any individual character
        - Cannot contain empty strings, it represents element of a set. Empty set is not an element of the set
    - *
        - The asterisk/star symbol means the previous item can appear 0 or as many time as it is required
    - +
        - The plus symbol means the previous item must appear at least once, otherwise, it is similar to the asterisk
    - ?
        - The question mark symbol means the previous item can appear exactly 0 or 1 times.
    - {}

    - $\{exact\}$: the item must appear some exact amount of times
    - $\{\text{min},\}$: the item must appear at least min amount of times
    - $\{\text{min}, \text{max}\}$: the item can appear between min and max times
- ()
    - Used to group items together
- |
    - The pipe symbol is for alternation.
- []
    - Specifying a range of characters within square brackets
    - $(a|b) \equiv [ab]$
- ^
    - Excluding characters with ^ symbol at the start of a section of square brackets
- \
    - To match special characters in a regex precede the special characters with a backlash
- **Special Characters**

  \d : Matches any digit
  \D: Matches any non-digit
  \s: Matches white space character
  \S: Matches a non-white space character
  \w: Matches a word character [a-zA-Z0-9]
  \W: Matches a non-word character [^a-zA-Z0-9]

  Which of the following expressions matches the same string as

  \d+

  a. [0-9]\d*
  b. [0-9][0-9]*
  c. (0-9)(0-9)*
  d. Both a and b
  e. Both a and c
-
    - D. \d means $[0-9]$, so both A and B are correct

Write a regular expression to match a standard 10 digit phone number in the format XXX-XXX-XXXX.

-
    - $[0-9]\{3\}$-$[0-9]\{3\}$-$[0-9]\{4\}$
    - \d{3}-\d{3}-\d{4}
    - From Vancouver
        - ((778)|(236)|(604))-\d{3}-\d{4}

Binary strings that have exactly one 1.

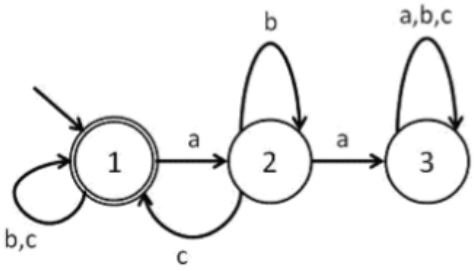
-
    - 0^*10^*
- Binary strings that have at most one 1
    - $0^*1\{0,1\}0^*$
    - $0^*1?0^*$
- Binary strings that have at least one 1

        - $(0|1)^*1(0|1)^*$
    - Binary strings that have exactly two 1s
        - $0^*10^*10^*$
        - ...

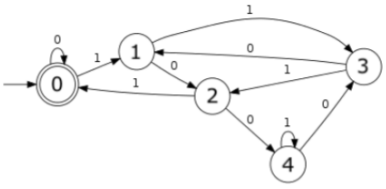
July 24th - Module 06 (Continued)

Module 06 (Continued)

- DFA definition
    - DFA = $(\Sigma, S, s_0, F, \delta)$
    - Σ: finite set of characters
    - S: finite set of states
    - $s_0 \in S$: initial state
    - $F \subseteq S$: set of accepting state
    - $\delta : S \times \Sigma \to S$: the transition function



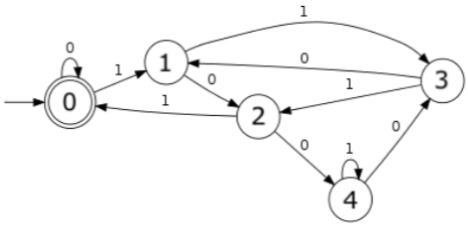
-
    - Σ: {a, b, c}
    - S: {1, 2, 3}
    - $s_0 \in S$: 1
    - $F \subseteq S$: {1}
    - $\delta : S \times \Sigma \to S$: e.g. $\delta(2, a) = 3$



Assuming this DFA is defined by the 5-tuple $(\Sigma, S, s0, F, \delta)$, what option is incorrect?

a. $\Sigma : \{0, 1, 2, 3, 4\}$
b. $S : \{0, 1, 2, 3, 4\}$
c. $s0 : 0$
d. $F : \{0\}$
e. None of the above.

-
    - A. $\Sigma = \{0, 1\}$



This DFA accepts the empty string, and which other strings?

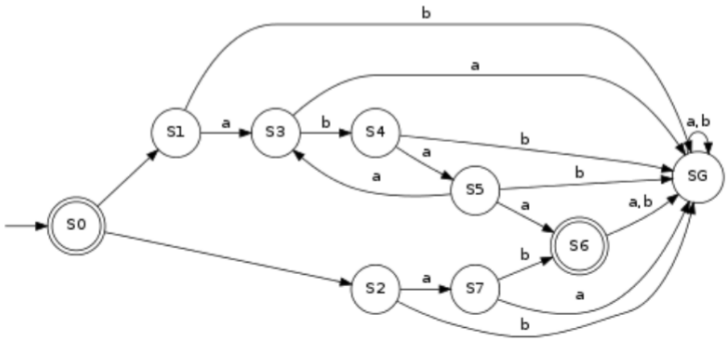
a. Strings whose length is divisible by 5.
b. Unsigned binary integers that are prime numbers.
c. Strings where every 1 is followed by 01.
d. Unsigned binary integers that are divisible by 5.
e. None of the above.

- D. The only accepting state is 0 and each state represents the value of $x \bmod 5$
- Adding the last digit to be 0 in binary, we are multiplying the number by 2. Adding the last digit to be 1, we are multiplying the number by 2 and adding 1.

- NFA
  - Similar to DFA, but
    - There can be multiple arrows with the same label leaving from a state
    - There can be arrows labelled ϵ (empty string) that we can take without reading the next input character
    - We can sometimes choose which state to go to
    - NFA accepts a string if at least 1 sequence of choices leads to an accepting state

What regular expression corresponds to the strings that this NFA accepts?

a. ε | ab
b. ε | abaa
c. ε | ab | abaa
d. ε | ab | (aba)+a
e. None of the above.



- D. Focus on s_3, s_4, s_5; this corresponds to $(aba)+$

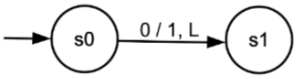
- Sequential Circuits
  - Generate output based on the current inputs, and on previous inputs
  - Feedback loops
    - Feedback loops are wires connecting the output of the circuit with some gate's input
  - DFAs can be translated into sequential circuits

Which of these sentence is False, i.e., does not describe a **difference** between Combinational and Sequential circuits?

a. The output in Combinational Circuit only relies on the input, while the output in Sequential Circuit relies on the input and previous output.
b. Combinational Circuit can only have the main operation gates (AND, OR, XOR, NOT), while Sequential Circuit can have more components (MUX, ADD).
c. Combinational Circuit doesn't have a feedback loop, while Sequential Circuit does have feedback loops.
d. Combinational Circuit is independent of time, while Sequential Circuit depends on time.
e. All sentences are True

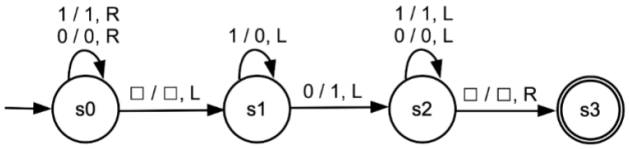
- - B. Both circuits can have other components besides the main gates

- Turing Machine
  - Mathematical model of computation that defines an abstract machine
  - Manipulates symbols on a strip of tape according to a table of rules
  - Capable of implementing that algorithm's logic can be constructed
  - Special DFA that has a tape attached to it
    - Tape is infinite
    - Head can move one step to the left or right
    - Can read/write one position of the tape
    - Each position can contain one element of the alphabet
    - Special symbol to represent a blank position: □
  - < read >/< write>, < move >
    - 1/0, R: reads value 1, replace it with 0, and move the head to the right
    - Each arrow will have 3 operations
      - Read the value of the tape
      - Write a value on the tape
      - Move the head left or right
    - **Example:**

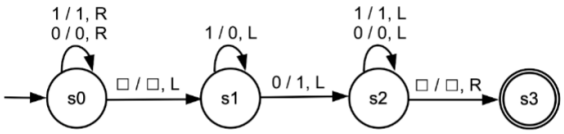


- It goes from state $s0$ to state $s1$ if the value read is 0.
- It replaces it with 1.
- It moves the head to the left.

- **Example:** a Turing Machine that adds 1 to a binary number.



When is this Turing Machine reaching state $s1$?



a. When the head reaches the first number of the binary string.
b. When the head reaches the end of the binary string.
c. When the head moves to right.
d. When the head moves to left.
e. Impossible to tell without seeing the tape.

-
  - B. On the rightmost element of the binary number

July 26th - Module 07

Module 07

- Reading Review
  - Universal Modus Ponens

$$\forall x, P(x) \to Q(x)$$
$$P(a), \textit{for a particular } a$$
$$\therefore Q(a)$$

  - Universal Modus Tollens

$$\forall x, P(x) \to Q(x)$$
$$\neg Q(a), \textit{for a particular } a$$
$$\therefore P(a)$$

  - Universal Transitivity

$$\forall x, P(x) \to Q(x)$$
$$\forall x, Q(x) \to R(x)$$
$$\therefore \forall x, P(x) \to R(x)$$

  - Direct Proof
    - Start with the hypothesis of a statement and make one deduction after another until you reach the conclusion
    - Constructive Proofs of Existence
      - $\exists x \in D, Q(x)$
      - Either: find an x that makes $Q(x)$ True
      - Or: give set of directions for finding such an x
    - Counterexample
      - $\forall x, P(x) \to Q(x)$
      - Disprove: find a x in D, such that $P(x)$ is True, but $Q(x)$ is False.
    - Method of Exhaustion (Finite elements)
      - $\forall x, P(x) \to Q(x)$
      - Prove: statement holds for every element
    - Method of Generalizing from the Generic Particular

- $\forall x, P(x) \rightarrow Q(x)$
- Suppose x is a particular but arbitrarily chosen element in the set, and show that x satisfies the property

Question 7:

- P: John, Mary, Sally, and Joe
- John is married
- Mary is married
- John is a parent of Sally
- Mary is a parent of John

$$\forall x \in P, \exists y \in P, Married(x) \rightarrow ParentOf(x, y)$$

- For x = John, we can find y = Sally for the statement to be True
- For x = Mary, we can find y = John for the statement to be True
- Neither Sally nor Joe is married, so the statement is always True for both of them
- This is Method of Exhaustion

- Direct Proofs
  - Statements are known/assumed to be True
  - Two general forms
    - Starting with a universal quantifier
    - Starting with a existential quantifier
  - Universal
    - Instantiation

$$\begin{gathered} \forall x \in D, P(x) \\ x_i \in D \\ \therefore P(x_i) \end{gathered}$$

    - Generalization

$$\begin{gathered} P(x), for\ arbitrary\ x \in D \\ \therefore \forall x \in D, P(x) \end{gathered}$$

  - Existential
    - Generalization

$$\begin{gathered} x \in D \\ P(x) \\ \therefore \exists x \in D, P(x) \end{gathered}$$

    - Instantiation

$$\begin{gathered} \exists x \in D, P(x) \\ \therefore P(w), for\ an\ unspecified\ witness\ w \in D \end{gathered}$$

  - Existential Quantifiers
    - $\exists x \in D, P(x)$
    - Template:
      - Choose x = <some value in D>
      - Verify that the x we chose satisfies the predicate

How do we translate: There is a prime number x such that $3^x + 2$ is not prime into predicate logic?

a. $\forall x \in \mathbb{Z}^+, Prime(x) \wedge \sim Prime(3^x + 2)$
b. $\exists x \in \mathbb{Z}^+, Prime(x) \wedge \sim Prime(3^x + 2)$
c. $\forall x \in \mathbb{Z}^+, Prime(x) \rightarrow \sim Prime(3^x + 2)$
d. $\exists x \in \mathbb{Z}^+, Prime(x) \rightarrow \sim Prime(3^x + 2)$
e. $\forall x \in P, \sim Prime(3^x + 2)$ where P is the set of all primes

-
  - B. for D, we can choose a composite number and the statement would be True, thus not what we are trying to prove
  - Proof:

    Choose $x = 5$,

    $Prime(5) \equiv T$, x is prime because its only factors are 1 and 5,

    $3^x + 2 = 3^5 + 2 = 243 + 2 = 245$, there is a factor for 245, which is 5

    $\neg Prime(3^x + 2) \equiv \neg Prime(245) \equiv \neg F \equiv T$.

    ■

- Universal Quantifiers
  - $\forall x \in D, P(x)$
  - Template
    - Consider an unspecified element x of D
      - We can only use properties common to all elements of D (Universal Generalization)
    - Verify that the predicate P holds for this x
  - Example: $\forall n \in \mathbb{Z}^+, n^n \geq n!$

    For every positive integer n^n is greater or equal to factorial of n.

    What fact we will use as the main **argument** for our proof?

    a. The fact that $n! \leq n^n$
    b. The fact that n is a positive integer
    c. The fact that n is greater or equal than all number from 1 to n
    d. $n^n = n.n.n \cdots n$
    e. $n! = 1.2.3 \cdot n$

    -
      - C. A is the conclusion, B is the setup of the proof, D and E are just definitions.
    - Proof:

      Consider an unspecified positive integer x

      $x! = 1 \cdot 2 \cdot 3 \ldots (x-1) \cdot x$,

      Each term of $x!$ is less than or equal to x, which is $1 \leq x$, $2 \leq x$, $3 \leq x$,..., $x \leq x$,

      Since they are all positive, we can conclude that $1 \cdot 2 \cdot 3 \ldots (x-1) \cdot x \leq x \cdot x \cdot x \ldots x \cdot x$

      The right side of the inequation has x terms, so we have $x! \leq x^x$

      ■

  - $\forall x \in D, P(x) \rightarrow Q(x)$
  - Template

- Consider an unspecified element x of D
- Assume that $P(x)$ is True
- Use this and properties of the element of D to verify that the predicate Q holds for this x

- Example: $\forall n \in \mathbb{N}, (n > 1023) \rightarrow (10n \leq n \log_2 n)$

  Why can we write Assume that P(x) is true?

  a. Because these are the only cases where Q(x) matters.
  b. Because P(x) is preceded by a universal quantifier.
  c. Because we know that P(x) is true.
  d. Both (a) and (c)
  e. Both (b) and (c)

  - A. For $P(x)$ is False, the statement is always True, nothing needs to be proved

  Prove that all natural numbers n larger than 1023, $10n \leq n \log_2 n$ holds.

  What can we assume?

  a. Nothing, since there is no implication.
  b. That n is a natural number.
  c. That n is larger than 1023.
  d. That $10n \leq n \log_2 n$.
  e. That $10n \leq n \log_2 n$ for $n \geq 1024$

  - C. We are assuming $P(x)$, which is if $n > 1023$. D and E are conclusions so they cannot be assumed.

  - Proofs:

    Consider an unspecified natural number n,

    Assume that $n \geq 1024$,

    $n \geq 2^{10}$,

    $\log_2 n \geq \log_2 2^{10}$,

    $\log_2 n \geq 10$,

    $n \log_2 n \geq 10n$.

    ■

- Nested Quantifiers
  - Start the proof from the outermost quantifier and work our way inwards
  - $\forall x \in \mathbb{R}, \forall y \in \mathbb{R}, \exists z \in \mathbb{R}, P(x, y, z)$
  - Proof:

    $\forall x \in \mathbb{R}, \forall y \in R$:

    ...

    $\exists z \in \mathbb{R}$:

    ...

    Prove $P(x, y, z)$
  - Example:
    $\forall x \in \mathbb{R}, \forall y \in \mathbb{R}, \exists z \in \mathbb{R}, (x \neq y) \rightarrow (x < z < y) \vee (y < z < x)$
    - Proof:

      $\forall x \in \mathbb{R}, \forall y \in \mathbb{R}$:

- Consider two unspecified real numbers x and y
- Assume that $x \neq y$
- Assume without loss of generality (WLOG) that $x < y$ so we prove that $(x < z < y)$

  $\exists z \in \mathbb{R}$:

  - Choose $z = \frac{x+y}{2}$

    Prove that $x < z < y$.

    - Since $x < y$, we have that $\frac{x+x}{2} < \frac{x+y}{2}$
    - Since $x < y$, we have that $\frac{x+y}{2} < \frac{y+y}{2}$
    - Therefore, we have $\frac{x+x}{2} < \frac{x+y}{2} < \frac{y+y}{2}$
    - Which is the same as $x < z < y$

    ■

For every positive integer n, there is a number m larger than n that is divisible by 7.

For this statement we can do the following on the proof:

a. Choose an exact value for *n* and assume an arbitrary value for *m*.
b. Assume an arbitrary value for *n* and choose an exact value for *m*.
c. Choose exact values for *n* and *m*.
d. Only assume arbitrary values for *n* and *m*.

- B. n is connected to a universal quantifier, while m is connected to an existential quantifier; m can be dependent on n, instead of choosing an exact value.
- $DivisibleBy7(x) : \exists c \in \mathbb{Z}, x = 7 \cdot c$

  $\forall n \in \mathbb{Z}^+, \exists m \in \mathbb{Z}^+, (m > n) \wedge DivisibleBy7(m)$

  Proof:

  $\forall n \in \mathbb{Z}^+$:

  - Consider an unspecified positive integer n

    $\exists m \in \mathbb{Z}^+$:

    - Choose $m = 7 \cdot n$

      Prove that $(m > n) \wedge DivisibleBy7(m)$.

      - Clearly $m > n$, since m and n are positive integers
      - Also, m is divisible by 7, since $\exists c \in \mathbb{Z}^+$, that $m = 7 \cdot c$, this number is n.

- What to do...
  - Check to see all variables that are with the existential quantifier. Remember you can choose those values.
  - Check what you can assume (make sure to not assume what you want to prove).
  - Remember you can use rules of inference for your justifications.
  - You can divide any proof by cases, as long as the cases combined give all numbers on your domain (even/odd or $x < y / x = y / x > y$)
- What not to do...
  - A proof that depends on drawings and not a formal description.
  - Use examples when you are trying to prove something with the universal quantifier.
  - The justification is 'by definition'
  - Start a sentence with 'we know that ...'
  - Start assuming the conclusion, and getting to the premises.
-
  - **Even:** If x is even, then
    $$\exists k \in \mathbb{Z}, x = 2k$$
  - **Odd:** If x is odd, then
    $$\exists k \in \mathbb{Z}, x = 2k + 1$$
  - **Divisible:** If $a|b$, then a divides b, i.e., then
    $$\exists k \in \mathbb{Z}, b = a \cdot k$$
  - **Rational number:** If x is rational, then
    $$\exists a \in \mathbb{Z}, \exists b \in \mathbb{Z}^*, x = a/b$$
  - **Perfect Square:** If x is a perfect square, then
    $$\exists k \in \mathbb{Z}, x = k^2$$
  - **Not Prime:** If x is not a prime, then
    $$\exists a, b \in \mathbb{Z}, x = a \cdot b \text{ where } a, b \neq 1, x$$

  Prove that $(B - A) \subseteq A^c$.
  Proof:
  - Consider an element x in $\mathcal{U}$ and two sets in $\mathcal{P}(\mathcal{U})$.
  - Assume $x \in (B - A)$, then $x \in B \wedge x \notin A$.

  What should be the rest of the prove?

  a. Then $x \notin A$.
  b. By specialization, $x \notin A$, and therefore $x \in A^c$.
  c. We can clearly see that $x \notin A$, hence $x \in A^c$.
  d. By specialization, $x \notin A$, therefore, by definition of complement operation, $x \in A^c$.
  e. None of the above

- - D.
- Examples
  - Show that $\forall x \in \mathbb{R}, \forall y \in \mathbb{R}, (x \in \mathbb{Q} \land y \in \mathbb{Q}) \rightarrow x \cdot y \in \mathbb{Q}$
    - Consider two unspecified real number x and y:

      Assume both x and y are rational numbers,

      Since x is rational, we can write x as $\frac{a}{b}$, where $a \in \mathbb{Z} \land b \in \mathbb{Z}$, and $b \neq 0$,

      Similarly, for y, we can write y as $\frac{c}{d}$, where $c \in \mathbb{Z} \land d \in \mathbb{Z}$, and $d \neq 0$,

      The product of x and y is $x \cdot y = \frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$

      Since the product of integers is also an integer, the product of non-zero integers is also a non-zero integer,

      Therefore, $\exists e \in \mathbb{Z}, e = ac$, $\exists f \in \mathbb{Z}, f = bd \land f \neq 0$,

      Hence, we can write $x \cdot y$ as $\frac{e}{f}$, where $e \in \mathbb{Z} \land f \in \mathbb{Z} \land f \neq 0$, therefore, $x \cdot y$ is rational
  - Show that $\forall n \in \mathbb{Z}^+, \exists m \in \mathbb{Z}, Even(m) \land (2n < m < 2n + 3)$
    - Consider an unspecified positive integer n:

      Choose an even integer $m = 2n + 2$.

      Since there is an integer $k = n + 1$ such that $m = 2(n + 1) = 2k$,

      m is even

      Since $m - 2n = (2n + 2) - 2n = 2 > 0$, then $m > 2n$

      Since $m - (2n + 3) = (2n + 2) - (2n + 3) = -1 < 0$, then $m < 2n + 3$

      Therefore, m is an even number, and $2n < m < 2n + 3$
  - Show that every positive odd integers is the difference between 2 perfect squares; $\forall x \in Z^+, Odd(x) \rightarrow (\exists y \in \mathbb{Z}, \exists z \in \mathbb{Z}, x = y^2 - z^2)$
    - Consider an unspecified positive integer x:

      Assume x is odd, x can be written as $x = 2n + 1$, where $n \in \mathbb{Z}$.

      Choose an integer $y = n + 1$, and $z = n$.

      Then $y^2 - z^2 = (n + 1)^2 - n^2 = 2n + 1 = x$

July 28th - Module 08

Module 08

- Reading Review
  - Method of Proof by Contradiction
    - Assumption of a given statement is not True leads to a contradiction, absurdity, or impossibility, then that assumption must be false; hence the given statement must be True.
    - Suppose the statement to be proved is false. That is, suppose the negation of the statement is True.
    - Show that this supposition leads to a contradiction
    - Conclude that the statement to be proved is True
  - Method of Proof by Contrapositive

- Take the contrapositive of a statement and prove the contrapositive by a direct proof. This concludes that the original statement is True.
- $\forall x, P(x) \rightarrow Q(x)$
- Express it in: $\forall x, \neg Q(x) \rightarrow \neg P(x)$
- Prove the contrapositive by direct proof
    - Suppose $Q(x)$ is False
    - Show that $P(x)$ is also False

Question 2: If you were going to use a proof by contradiction to prove the following theorem, which of these would be the appropriate assumption to make?

For all integers m and n, if m + n is even then m and n are both even or m and n are both odd.

$\forall n, m \in \mathbb{Z}, Even(m+n) \rightarrow ((Even(m) \wedge Even(n)) \vee (Odd(m) \wedge Odd(n)))$

We need to negate the statement!

- - There are two integers m and n, such that $m+n$ is even and either m is even and n is odd or m is odd and n is even

- Indirect Proof
    - Change the original statement and make different assumptions in order to prove it: claim which method is going to be used
    - Two techniques
        - Contrapositive
        - Contradiction
    - Contrapositive
        - Show that *if the square of a positive integer n is even, then n is even*; $\forall n \in \mathbb{Z}^+, Even(n^2) \rightarrow Even(n)$
            - Proof:

              (Showing that *if n is odd, then the square of n is odd* is much easier, we need to somehow convert the original statement into this one)

              We will prove the contrapositive: If a positive integer n is odd, then its square is odd

              Consider an unspecified positive integer n

              Assume n is odd, so $n = 2k+1$ for some integer k

              Hence
              $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2 \cdot (2k^2 + 2k) + 1$

              Therefore n^2 is odd

              Therefore, the original statement is True

              ■

What is the relationship between

If the square of a positive integer n is even, then n is even.

If a positive integer n is odd, then its square is odd.

a. They are the same exact implication and they are equivalents.
b. They are inverses and they are equivalents.
c. They are inverses and they are not equivalents.
d. They are contrapositives and they are equivalents.
e. They are contrapositives and they are not equivalents.

- - D. Even and odd are negatives of each other

Is it possible to use contrapositive in a statement that only has existential quantifiers?

a. Yes, there is no problem with that.
b. No, it only works for universal quantifiers.
c. Yes, but statements with existential quantifiers and implications do not say much at all.
d. No, we cannot have implication on a statement with only existential quantifiers.
e. None of the above.

- - C. The implication with an existential quantifier **could** be vacuously True.

- Translate: There is a positive integer n that is even and has an even square
  - Correct Translation: $\exists n \in \mathbb{Z}^+, Even(n) \wedge Even(n^2)$
  - Wrong Translation: $\exists n \in \mathbb{Z}^+, Even(n) \rightarrow Even(n^2)$
    - Choose $n = 3$: the statement is True
    - For contrapositive, choose $n^2 = 9$: the statement is still True

- Show that *Given $x \in \mathbb{R}^+$, if x is an irrational number, then $\sqrt{x}$ is irrational*; $\forall x \in \mathbb{R}^+, x \in \bar{\mathbb{Q}} \rightarrow \sqrt{x} \in \bar{\mathbb{Q}}$
  - Proof:

    We will prove the contrapositive: Given $x \in \mathbb{R}^+$, if $\sqrt{x}$ is a rational number, then x is rational.

    Consider an unspecified real number x

    Assume $\sqrt{x}$ is rational, so $\sqrt{x} = \frac{a}{b}$ for some $b \neq 0$

    Hence $x = (\sqrt{x})^2 = (\frac{a}{b})^2 = \frac{a^2}{b^2}$

    Since the square of an integer is still an integer and the square of a non-zero integer is still a non-zero integer,

    Therefore, $a^2 \in \mathbb{Z} \wedge b^2 \in \mathbb{Z} \wedge b^2 \neq 0$

    Therefore, x is a rational number

    Therefore, by contraposition, the original statement is True

- ...

July 31st - Module 08 (Continued) & Module 09

Module 08 (Continued)

- Indirect Proof
    - Contradiction
        - Assume what we are proving is False, if we can derive a contradiction from it, we can conclude it was actually True
        - Want to prove P

          Assume $\neg P \equiv T$, or $P \equiv F$

          Conclude smth False or absurd $\neg P \to F$

          Therefore, our assumption is wrong, that is $\neg\neg P \equiv T$, so $P \equiv T$
        - Show that not every student got the an above average grade on Midterm 1; $\neg \forall s \in S, g_s > \bar{g}$
            - Negated statement: $\forall s \in S, g_s > \bar{g}$
            - Proof:

              We prove by contradiction, so we:

              Assume a to be the average and g_i to be the grade for each student

              Assume $g_i > a$ for all g_i

              Therefore $\sum_{i=1}^{n} g_i > a \cdot n$

              Since $a = \frac{\sum_{i=1}^{n}}{n}$

              Hence we have $a > a$

              Which is a contradiction, so the original statement holds

              ■

              When trying to prove $\forall x \in D, P(x)$, by contradiction what should we do?

              a. Assume $\forall x \in D, \sim P(x)$ and consider an unspecified x.
              b. Assume $\exists x \in D, \sim P(x)$ and consider an unspecified x.
              c. Assume $\exists x \in D, \sim P(x)$ and choose an x.
              d. Assume $\exists x \in D, P(x)$ and consider an unspecified x.
              e. None of the above
            -
                - E. We assume the negated statement, but we are not following a direct proof afterwards, we try to look for a contradiction based on this assumption that the negated statement is True.
        - During a proof by contradiction, we negate the whole statement, including the quantifiers
            - Proof:

              We prove this by contradiction, so we assume $\exists x \in D, \neg P(x)$

              Since there is an x for which $P(x)$ is not True

              ...

              Reach to some contradiction

              The original statement is True

■

- We do not choose x, but we assume it exists

If we want to prove $P(x) \to Q(x)$ by contradiction, what can we assume?

a. $P(x)$ is True.
b. $\sim P(x) \to \sim Q(x)$ is True
c. $\sim P(x) \vee Q(x)$ is True
d. $P(x) \wedge \sim Q(x)$ is True
e. None of the above

- D. Assume the negated of $P(x) \to Q(x)$, which is $P \wedge \neg Q \equiv T$

If we want to prove $(P_1(x) \wedge P_2(x) \wedge P_3(x)) \to Q(x)$ by contradiction, what can we assume?

a. $P_1(x) \wedge P_2(x) \wedge P_3(x)$ is True.
b. $(P_1(x) \to Q(x)) \wedge (P_2(x) \to Q(x)) \wedge (P_3(x) \to Q(x))$ is True
c. $P_1(x) \wedge P_2(x) \wedge P_3(x) \wedge \sim Q(x)$ is True
d. $\sim(P_1(x) \wedge P_2(x) \wedge P_3(x)) \vee Q(x)$ is True
e. None of the above

- C.

The sum of two even numbers is even.

What can we assume to be True in order to proof this by contradiction?

a. $\forall x, y \in \mathbb{Z}, (Even(x) \wedge Even(y)) \to Even(x + y)$
b. $\exists x, y \in \mathbb{Z}, Even(x) \wedge Even(y) \wedge Odd(x + y)$
c. $\forall x, y \in \mathbb{Z}, Even(x) \wedge Even(y) \wedge Odd(x + y)$
d. $\exists x, y \in \mathbb{Z}, (Even(x) \wedge Even(y)) \to Even(x + y)$
e. None of the above

- B. Original statement
  $\forall x, y \in \mathbb{Z}, (Even(x) \wedge Even(y)) \to Even(x + y)$

- Show that the sum of 2 even numbers is even
  - Proof:

    We prove by contradiction, so we:

    Assume x is even, so $\exists a \in \mathbb{Z}, x = 2a$

    Assume y is even, so $\exists b \in \mathbb{Z}, y = 2b$

    Assume $x + y$ is odd

    $x + y = 2a + 2b = 2(a + b) = 2c$

    Since the addition of 2 integers is an integer, then c is an integer

    Therefore $\exists c \in \mathbb{Z}, x + y = 2c$

    So $x + y$ is even, but we assumed that it was odd, which is a contradiction

    So the original statement holds

    ■

- Show that $\forall n \in \mathbb{Z}^+, n! \leq n^n$
  - Proof:

We prove by contradiction, so we:

Assume x is a positive integer

Assume $x! > x^x$

$x! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-1) \cdot x$

$x^x = x \cdot x \cdot x \cdot \ldots \cdot x \cdot x$

Both have a product of the same number of terms and since $x! > x^x$ at least one term of $x!$ has to be greater than one term of x^x

The largest term in $x!$ is x, which means x must be strictly greater than x

This is a contradiction, so the original statement holds

■

- Direct Proof vs. Indirect Proof
    - Always try a direct proof, usually simpler
    - Try an indirect proof if you know that negating will help you (irrational → rational)
    - For direct proof, last step is conclusion that matches the original statement
    - For contraposition, the last step is conclusion that matches the contrapositive of the original statement: followed by a Direct Proof
    - For contradiction, the last step is a contradiction: cannot be followed by Direct Proof

Module 09 - Weak Induction Proofs

- Reading Review
    - $\sum_{k=m}^{n} a_k = a_m + a_{m+1} + a_{m+2} + \ldots + a_n$
    - $\prod_{k=m}^{n} a_k = a_m \cdot a_{m+1} \cdot a_{m+2} \ldots a_n$
    - $\sum_{k=m}^{n} a_k + \sum_{k=m}^{n} b_k = \sum_{k=m}^{n} (a_k + b_k)$
    - $c \cdot \sum_{k=m}^{n} a_k = \sum_{k=m}^{n} c \cdot a_k$
    - $\prod_{k=m}^{n} a_k \cdot \prod_{k=m}^{n} b_k = \prod_{k=m}^{n} (a_k \cdot b_k)$
    - $n! = n \cdot (n-1) \cdot \ldots 3 \cdot 2 \cdot 1$
    - $\binom{n}{x} = \frac{n!}{x!(n-x)!}$
    - $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
    - $\sum_{i=1}^{n} r^i = \frac{r^{n+1}-1}{r-1}$
    - Principle of Weak Mathematical Induction
        - Let $P(n)$ be a property that is defined for integers n, let a be a fixed integer
        - **Basic Step**: $P(a)$ is True

          Induction Hypothesis: Assume $P(n)$ is True

          Inductive Step: Prove that is True for $P(n+1)$
        - **Basic Step**: $P(a)$ is True

          Induction Hypothesis: Assume $P(n-1)$ is True

          Inductive Step: Prove that is True for $P(n)$

Question 7: Prove that it is possible to pay for postage using only stamps of 2c and 5c for any amount of n cents, for $n \geq 4$.

What can we assume for a weak induction proof of this problem?

a. It is possible to pay for postage using 2 cents and 5 cents stamps for any amount n cents. And then proof this is True for n+1.

b. It is possible to pay for postage using 2 cents and 5 cents stamps for any amount n-1 cents. And then proof this is True for n.

c. It is possible to pay for postage using 2 cents and 5 cents stamps for any amount n+1 cents. And then proof this is True for n+1.

d. It is possible to pay for postage using 2 cents and 5 cents stamps for any amount n=4 cents. And then proof this is True for n.

- - A, B. Either $P(n-1) \to P(n)$ or $P(n) \to P(n+1)$

- Induction
  - *One (or more) base case*: Proving for the smallest value

    Induction Hypothesis (IH): assume true - $n-1$ if using **weak** induction; any $k < n$ if using **strong** induction

    Induction Step: Assume that the *IH* is true and prove the problem for n
  - Example: Prove that it is possible to pay for postage using only stamps of 2c and 5c for any amount of n cents, where is n is at least 4
    - Predicate: $\forall n \in \mathbb{Z}^+, (n \geq 4) \to (\exists a \in \mathbb{N}_0, \exists b \in \mathbb{N}_0, n = 2a + 5b)$
    - Proof:

      Base: $n = 4$ - 2c + 2c

      Induction Hypothesis: It is possible to pay for postage using only stamps of 2c and 5c for an amount of $n-1$ cents

      Induction Step:

      Consider an unspecified amount n, that $n > 4$

      By IH, we know it is possible for $n-1$

      If we used one 5c stamp to make $n-1$ cents of postage, we can replace it with three 2c stamps, since $3 \cdot 2c$ - 5c = 1c.

      If we used two 2c stamps to make $n-1$ cents, we can replace it with one 5c stamp, since 5c - $2 \cdot 2c$ = 1c.

      Therefore, n cents is possible

      ■
    - ...

August 2, Module 09 (Continued)

Module 09 (Continued)

- Induction
  - Single-elimination Tournaments Problem

- 

- Teams play one another in pairs
- The winner of each pair advances to the next round

If we have n rounds of playoffs, how many teams can participate?

a. n
b. $2n$
c. n^2
d. 2^n
e. None of the above.

- D.

If t teams can participate in a playoff with n rounds, how many teams can participate in a playoff with one more round?

a. t
b. $2t$
c. t^2
d. 2^t
e. None of the above.

-

- Show that in a single-elimination tournament with $n \geq 1$ rounds 2^n teams can participate
    - Consider an unspecified playoff with $n-1$ rounds. Assume that t teams can participate in that playoff

      We can think of a playoff with n rounds as follows:

      2 playoffs with $n-1$ rounds proceed in parallel

      The 2 winners then play each other in round n

      Since each playoff with $n-1$ rounds has t teams, we thus have $2t$ teams in total.

      If we define:

      $P(t, n)$: Up to t teams can participate in a playoff with n rounds

      Then we have proved:

      $\forall t \in \mathbb{Z}^+, \forall n \in Z^+, P(t, n-1) \rightarrow P(2t, n)$

      We know that only 2 teams can play with 1 round, so $P(2, 1)$ is True

      Therefore, by Modus Ponens, $P(4, 2)$ is True;

Thus, $P(8,4)$ is True,

Then, $P(16,8)$ is True.

- Proof:

  Base: One round will need 2 teams, $t = 2^n = 2^1 = 2$

  IH: In a single-elimination tournament with $n-1$ rounds 2^{n-1} teams can participate

  Induction Step: Adding one round, doubles the number of teams, so for n rounds we have the number of teams for $n-1$ rounds doubled - $2 \cdot 2^{n-1} = 2^n$

- We know $T(n) = 2 \cdot T(n-1)$, and we also know that for 1 round, we need 2 teams $T(1) = 2$

  $T(n) = 2 \cdot T(n-1)$

  $T(n) = 2 \cdot (2 \cdot T(n-2))$

  $T(n) = 2 \cdot (2 \cdot (2 \cdot T(n-3)))$

  ...

  $T(n) = 2 \cdot (2 \cdot (2 \ldots (2 \cdot T(1))))$

  $T(n) = 2^n$

  Given that $T(n)$ is the number of teams we can have with n rounds, what we can also have?

  a. $T(n+1) = 2 \cdot T(n-1)$
  b. $T(n-1) = 2 \cdot T(n)$
  c. $T(n+2) = 2 \cdot T(n+1)$
  d. All the above
  e. None of the above.

- - C. Connecting 2 consecutive numbers

- Weak Induction Proof - 2 different ways
  - Assume $n-1$ on **IH** and prove to n, or
  - Assume n on **IH** and prove to $n+1$.
- Why does mathematical induction work
  - Want to prove $\forall n \in Z^+, P(n)$
    - We prove $P(1)$
    - We prove $\forall n \in \mathbb{Z}^+, P(n-1) \rightarrow P(n)$
      - Assume $P(n-1)$
      - Prove $P(n)$

    If we know p is True and $p \rightarrow q$, $q \rightarrow r$, and $r \rightarrow s$ are also True, what can we say about p, q, r and s?

    a. They are all False
    b. Only p is True
    c. Only p and q are True
    d. Only p, q and r are True
    e. They are all True

    - - E. By Transitivity Rule, all of the variables are True.

    - $P(1) \rightarrow P(2) \rightarrow P(3) \rightarrow P(4) \rightarrow P(5) \rightarrow \ldots$
  - Prove the weak induction theorem by contradiction

We want to prove $\forall n \in \mathbb{Z}^+, P(n)$:

- We prove P(1).
- We prove $\forall n \in \mathbb{Z}^+, P(n-1) \to P(n)$.
  - Assume $P(n-1)$
  - Prove $P(n)$

How can we resume The Weak Induction Theorem?

a. $P(1) \wedge (\forall n \in \mathbb{Z}^+, P(n-1) \to P(n)) \wedge \forall n \in \mathbb{Z}^+, P(n)$
b. $(P(1) \wedge (\forall n \in \mathbb{Z}^+, P(n-1) \to P(n))) \to \forall n \in \mathbb{Z}^+, P(n)$
c. $P(1) \wedge (\forall n \in \mathbb{Z}^+, P(n-1))$
d. $\forall n \in \mathbb{Z}^+, P(n) = P(1) \wedge (\forall n \in \mathbb{Z}^+, P(n-1) \wedge P(n))$
e. None of the above.

-
  - B. We prove $P(1)$, and $\forall n \in \mathbb{Z}^+, P(n-1) \to P(n)$, to prove $\forall n \in Z^+, P(n)$

- Show that
  $(P(1) \wedge (\forall n \in \mathbb{Z}^+, P(n-1) \to P(n))) \to (\forall n \in \mathbb{Z}^+, P(n))$
  - Proof:

    We use a proof by contradiction:

    Suppose that the premises hold, but the conclusion is False:

    $P(1) \equiv T$

    $\forall n \in \mathbb{Z}^+, P(n-1) \to P(n) \equiv T$

    $\neg \forall n \in \mathbb{Z}^+, P(n) \equiv \exists n \in \mathbb{Z}^+, \neg P(n) \equiv T$

    Let y be the smallest for which $P(y) \equiv F$

    Clearly $y > 1$, since $P(1) \equiv T$.

    Since y is the smallest value for which $P(n)$ is False, then $P(y-1) \equiv T$.

    We know that $\forall n \in \mathbb{Z}^+, P(n-1) \to P(n) \equiv T$, so $P(y-1) \to P(y) \equiv T$.

    Therefore, we have that
    $P(y) \equiv F, P(y-1) \equiv T, P(y-1) \to P(y) \equiv T$, which is a contradiction.

    Therefore $\forall n \in \mathbb{Z}^+, P(n)$ holds

    ■

What can we do when using induction on

$$S(n) = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

a. Both $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n-1) \to S(n))$ and $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n) \to S(n+1))$
b. Either $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n-1) \to S(n))$ or $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n) \to S(n+1))$
c. Only $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n-1) \to S(n))$ works.
d. Only $S(1) \wedge (\forall n \in \mathbb{Z}^+, S(n) \to S(n+1))$ works.
e. None of the above.

- B. We don't need to prove both, but both work.
- Proof of Sum: Version 1 & 2
  - Show that $S(n) = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
    - Proof:

      Base: $n = 1$

      $\sum_{k=1}^{1} k = 1 \leftrightarrow \frac{1(1+1)}{2} = 1$

      IH: Assume $S(n) = \frac{n(n+1)}{2}$

      Induction Step: Prove for $n+1$, we want to show that $S(n+1) = \frac{(n+1)(n+2)}{2}$

      $S(n+1) = \sum_{k=1}^{n+1} k = \sum_{k=1}^{n} k + (n+1)$

      $S(n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$

      ■
    - Proof:

      Base: $n = 1$, same as above

      IH: Assume $S(n-1) = \frac{(n-1)n}{2}$

      Induction Step: Prove for n, show that $S(n) = \frac{n(n+1)}{2}$

      $S(n) = \sum_{k=1}^{n} k = \sum_{k=1}^{n-1} k + n = \frac{(n-1)n}{2} + n = \frac{(n-1)n+2n}{2} = \frac{n(n+1)}{2}$

      ■

Problem: $\forall n \in \mathbb{Z}^+, n = n + 1$

- **Induction Hypothesis:** $n - 1 = (n - 1) + 1 = n$
- **Induction Step:**
- Consider an unspecified n.
- By IH we can assume that $n - 1 = n$.
- So for n we have: $n = (n - 1) + 1$
- By IH we know that $n - 1 = n$, so $n = n + 1$. ■

What is the problem with this proof?

- No base case, the base case cannot be proven to be True.

August 4 - Module 10

Module 10

- Reading Review
  - Principle of Strong Mathematical Induction
    - Let $P(n)$ be a property that is defined for integers n, and let a and b be fixed integers.
    - **Basic Step**: $P(a)$ is True, $P(a+1)$ is True, ..., $P(a+b)$ is True

      IH: Assume $P(k)$ is True for all $k < n$

Inductive Step: Proof that is True for $P(n)$

- **Basic Step**: $P(a)$ is True, $P(a+1)$ is True, ..., $P(a+b)$ is True

  IH: Assume $P(k)$ is True for all $k \leq n$

  Inductive Step: Proof that is True for $P(n+1)$

Question 4: Suppose that P(n) is a propositional function. For which positive integers n, P(n) is true if we prove that:

- $P(1)$ is True
- $\forall n \in \mathbb{Z}^+, P(\lfloor n/2 \rfloor) \rightarrow P(n)$

a. $P(2)$
b. $P(3)$
c. $P(4)$
d. $P(6)$
e. All of the above.
f. None of the above.

- - E. We can create a chain from any number ending in $P(1)$, so it will be True for all numbers in the chain.
  - $P(\lfloor \frac{2}{2} \rfloor) \rightarrow P(2)$
  - $P(\lfloor \frac{3}{2} \rfloor) \rightarrow P(3)$

- Strong Induction
  - Example 1: Consider the relation - $t_1 = 1$, $t_n = t_{\lfloor \frac{n}{2} \rfloor} + t_{\lceil \frac{n}{2} \rceil} + 1$, for all $n > 1$; Prove that $t_n = 2n - 1$, for all positive integers n
    - **Base**: $n = 1$: $t_1 = 2 \cdot 1 - 1 = 1$
    - **IH**: For any positive integer $k < n$, $t_k = 2k - 1$
    - **Inductive Step**: Consider unspecific positive integer $n > 1$

      By IH we know that $t_{\lfloor \frac{n}{2} \rfloor} = 2 \cdot \lfloor \frac{n}{2} \rfloor - 1$, and $t_{\lceil \frac{n}{2} \rceil} = 2 \cdot \lceil \frac{n}{2} \rceil - 1$

      Then $t_n = t_{\lfloor \frac{n}{2} \rfloor} + t_{\lceil \frac{n}{2} \rceil} + 1 = 2 \cdot \lfloor \frac{n}{2} \rfloor - 1 + 2 \cdot \lceil \frac{n}{2} \rceil - 1 + 1$

      Thus $t_n = 2(\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil) - 1 - 1 + 1 = 2 \cdot n - 1$
  - Strong Induction: $k = \lfloor n/2 \rfloor$

    $P(1) \rightarrow P(2) \rightarrow P(4) \rightarrow P(8) \rightarrow P(16) \rightarrow \cdots$

    $P(1) \rightarrow P(3) \rightarrow P(6) \rightarrow P(12) \rightarrow P(24) \rightarrow \cdots$

    $P(1) \rightarrow P(2) \rightarrow P(5) \rightarrow P(10) \rightarrow P(20) \rightarrow \cdots$

    $\vdots$

    - If $P(1)$ is True and $P(1) \rightarrow P(2)$ is True, then $P(2)$ is True
    - If $P(2)$ is True and $P(2) \rightarrow P(4)$ is True, then $P(4)$ is True
    - If $P(1)$ is True and $P(1) \rightarrow P(3)$ is True, then $P(3)$ is True
    - If $P(3)$ is True and $P(3) \rightarrow P(6)$ is True, then $P(6)$ is True
    - If $P(2)$ is True and $P(2) \rightarrow P(5)$ is True, then $P(5)$ is True
- Recursion vs. Induction
  -
```
(define (sum n)
    (if (= n 0)
        0
        (+ n (sum (- n 1)))))
```

-
```
def sum(n)
{
    if n == 0:
        return 0
    else:
        return n + sum(n-1)
}
```

- $$\sum_{i=0}^{0} i = 0$$
$$n + \sum_{i=0}^{n-1} i = \sum_{i=0}^{n} i$$

- Strong induction: the hypothesis will be for any value $k < n$, not only the value that is sequentially before it (not just from $n-1$ to n or from n to $n+1$)

  How many base cases we would need to prove $P(n)$, for $n \geq 1$ using each of these IH?

  a. When using $P(n-1)$ as IH, we need no base case.
  b. When using $P(n-2)$ as IH, we need at least 1 base case.
  c. When using $P(\lfloor n/2 \rfloor)$ as IH, we need at least 1 base case.
  d. When using $P(\lfloor n/2 \rfloor)$ as IH, we need at least 2 base cases.

-
  - C. We should prove all $P(k)$ that represents the first value of a chain

- Strong Induction (Continued)
  - Example 2: Eggs are sold in packs of 4 and 7 eggs, prove that: Fred can buy exactly n dinosaur eggs for any $n \geq 18$ using only pack of 4 and 7 eggs

    Problem: Fred can buy exactly n dinosaur eggs for any integer $n \geq 18$ using only pack of 4 and 7 eggs.

    What is the value we can use on our IH that gives us the **simplest** proof?

    a. $n-1$
    b. $n-2$
    c. $n-3$
    d. $n-4$
    e. Any value other than $n-1$, $n-2$, $n-3$, $n-4$.

  -
    - D. The simplest induction step is just adding a new pack of eggs, with 4 eggs being the smallest.

    How many base cases would we need when using the **simplest** value for our IH?

    a. 1
    b. 2
    c. 3
    d. 4
    e. 7

  -
    - D. Assuming that $n-4$ this is True, we need 4 base cases in our proof

- How do we decide which base cases we need?
    - Look at the induction hypothesis. We need to decide for which values of n we can be assume to be True to prove $P(n)$
    - The inductive step might not be usable for some values, for example
        - If we assume that $P(k)$ is True and k is smaller than smallest n possible
        - Because some mathematical steps only work for values of n greater than a fixed integer
    - If the induction step is not usable for x, then x needs to be a base case!

Problem: Fred can buy exactly n dinosaur eggs for any integer $n \geq 18$ using only pack of 4 and 7 eggs.

Base:

- $18 = 2 \cdot 7 + 1 \cdot 4$

Induction Hypothesis:

- Fred can buy exactly k dinosaur eggs for any integer $k < n$ using only pack of 4 and 7 eggs.

Induction Step:

- Then Fred can buy any n eggs as follows:
    - Fred wants to buy n eggs.
    - By IH he can buy $n - 4$ eggs.
    - Fred adds a pack of 4 eggs to the solution for $n - 4$ eggs.

Problem: Fred can buy exactly n dinosaur eggs for any integer $n \geq 18$ using only pack of 4 and 7 eggs.

What is the problem with the proof?

a. The step is wrong, since we don't use the pack with 7 eggs
b. The IH is wrong and k should be replaced by n-4
c. The base cases should be $x = 18$, $x = 19$, $x = 20$, $x = 21$
d. We should add $x = 19$ to the base case
e. All of the above.

-
    - C. In our induction step we are doing $x - 4$ so our base should have the first 4 cases
- **Base**: $18 = 1 \cdot 4 + 2 \cdot 7$

  $19 = 3 \cdot 4 + 1 \cdot 7$

  $20 = 5 \cdot 4$

  $21 = 3 \cdot 7$

  IH: Fred can buy exactly k dinosaur eggs for any integer $k < n$ using only pack of 4 and 7 eggs

  Inductive Step: Fred can buy any n eggs as follows, Fred wants to buy n eggs, by IH he can buy $n - 4$ eggs, Fred add pack of 4 to the solution for $n - 4$ eggs.

- Weak vs. Strong Induction

What is the difference between weak and strong induction when proving $P(n)$?

a. In weak induction we assume $P(n-1)$, while in strong induction we assume $P(n)$.
b. In weak induction we assume $P(n)$, while in strong induction we assume $P(n+1)$.
c. In weak induction we prove for $P(n)$, while in strong induction we prove for $P(n+1)$.
d. In weak induction we assume either $P(n-1)$ or $P(n)$, while in strong induction we assume $P(k)$ for $k < n$.
e. In weak induction we prove for $P(n-1)$, while in strong induction we prove for all $P(k)$, when $k < n$.

- - D. Weak induction is $P(n-1) \to P(n)$ or $P(n) \to P(n+1)$, while in strong induction $P(k) \to P(n)$ for when $k < n$

What is the correct sentence?

a. Every strong induction proof can be transformed trivially into a weak induction proof.
b. Every weak induction proof can be transformed trivially into a strong induction proof.
c. Every induction proof can be transformed trivially into a weak and a strong induction proof.
d. It is impossible to have a weak induction proof and a strong induction proof for the same problem.

- - B. In strong induction we prove $P(k) \to P(n)$, for some $k < n$, since $n-1 < n$ and $n < n+1$, then all weak induction are strong induction as well

- **Problem: Prove that it is possible to pay for postage using only stamps of 2c and 5c for any amount of n cents, where n is at least 4.**

  - **Base:**
    $n = 4$: 2c + 2c
    $n = 5$: 5c
  - **Induction Hypothesis:** It is possible to pay for postage using only stamps of 2c and 5c for any amount $k < n$ cents.
  - **Induction Step:**
  - Consider an unspecific amount n.
  - By IH we know it is possible for any amount of $k < n$ cents, so it is possible for an amount $n-2$.
  - We can add a 2c stamp to that solution to get the correct stamps for amount n.

- ...

August 9th - Module 11

Module 11

- Reading Review
  - Ω

        - f is of order *at least* g, written $f \in \Omega(g)$, *iff*, $\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n > n_0 \rightarrow c \cdot g(n) \leq f(n)$
    - O
        - f is of order *at most* g, written $f \in O(g)$, *iff*, $\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n > n_0 \rightarrow f(n) \leq c \cdot g(n)$
    - Θ
        - f is of order g, written $f \in \Theta(g)$, *iff*, $\exists c_1, c_2 \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n > n_0 \rightarrow c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
    - Properties
        - $f \in \Omega(g)$ and $f \in O(g)$, *iff*, $f \in \Theta(g)$
        - $f \in \Omega(g)$, *iff*, $g \in O(f)$
        - $f \in \Theta(g) \rightarrow g \in \Omega(f)$
    - Orders of Power Functions
        - $r < s \rightarrow x^r \in O(x^s)$

        Question 7: Choose the right answer.

        a. $2 \in O(2^x)$
        b. $x \in O(2^x)$
        c. $x \cdot \log_2 x \in O(2^x)$
        d. All of the above
        e. None of the above

    -
        - D. Power function $\in O$(Exponential function)
- Algorithm Competition

    Which algorithm is more efficient?

Algorithm A	Algorithm B
10 seconds	35 seconds

    a. Algorithm A
    b. Algorithm B
    c. They are equally fast
    d. Not enough information to tell

    -
        - D. Probably Bob's machine has an old and bad processor.
- Algorithm Efficiency
    - It does not depend on
        - The programming language used to implement it
        - The quality of the compiler or interpreter
        - The speed of the computer it is executed on
    - Can count the number of **elementary steps** of the algorithm

Which algorithm is more efficient?

Algorithm A	Algorithm B
50 steps using the same language, processor and compiler	125 steps using the same language, processor and compiler

a. Algorithm A
b. Algorithm B
c. They are equally fast
d. Not enough information to tell

- D. There is no specification about the size of the problem

- Consider the **size** of the data it is using
  - Use the variable n to give the size of the input
  - The number of steps will be based on n
  - A step is anything that can be computed in constant time, that is, independent from n

Which algorithm is more efficient?

Algorithm A	Algorithm B
3n steps using the same language, processor and compiler	6n steps using the same language, processor and compiler

a. Algorithm A
b. Algorithm B
c. They are equally fast
d. Not enough information to tell

- C. For small values of n, algorithm A may be better, but for large values, the constant will not make much of a difference

- Facts about execution times
  - Can not rely on the values of the constants in front of the functions describing the number of steps
  - It's almost impossible to compute the number of steps exactly
  - Focus on the **order** of the function that determines the number of steps
- So come up with
  - A way to count steps that ignores these constants
  - An approximation of the correct number of steps
- Comparison between algorithms
  - Compare how each algorithm reacts when we start to input large values of n

        - The function that grows fast, means the algorithm runs slow
    - Consider this problem
        - Given: a sorted list of names with phone numbers
        - Find: the phone number for a given name N
        - Algorithm L: check the first name, if it's not N, then check the second name, etc
            - Has approximately n steps
        - Algorithm B: check the name in the middle of the list. If N comes earlier alphabetically, search the first half of the list, if N comes later alphabetically, search the second half of the list.
            - Has approximately $10 \cdot \log_2 n$ steps
        - Experimenting
            - For $n = 15$, L takes 15 steps, B takes 40 steps
            - For $n = 63$, L takes 63 steps, B takes 60 steps
            - For $n = 1048575$, L takes 1048575 steps, B takes 200 steps
- Big O
    - $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$
    - Time of execution
        -

$f(n)$	$n = 10$	$n = 100$	$n = 1000$	$n = 10,000$	$n = 10^{12}$
n	10^{-11} s	10^{-10} s	10^{-9} s	10^{-8} s	1 s
$n \log n$	10^{-11} s	10^{-9} s	10^{-8} s	10^{-7} s	40 s
n^2	10^{-10} s	10^{-8} s	10^{-6} s	10^{-4} s	10^{12} s
n^3	10^{-9} s	10^{-6} s	10^{-3} s	1 s	10^{24} s
2^n	10^{-9} s	10^{18} s	10^{289} s	too long	too long

        - $10^4 s \approx 3 hrs$, $10^{12} s \approx 32 yrs$, $10^{18} s \approx 30 byrs$.
    - Approximations
        - For a sum of terms, we can eliminate all low order terms
            - $4n + 5 \to 4n$
            - $0.5n \log_3 n - 2n + 7 \to 0.5n \log_3 n$
            - $2^n + n^3 + 3n \to 2^n$
            - $5n^3 + 3n^2 + 6n \to 5n^3$
            - $10^4 + 10^5 n + 10^6 \log n + 10n^4 \to 10n^4$
        - For one term, we can eliminate any constant
            - $4n \to n$
            - $0.5n \log_3 n \to n \log_3 n \to n \log n$
            - $2^n \to 2^n$
            - $5n^3 \to n^3$
            - $10n^4 \to n^4$
        - Big O set for our functions
            - $4n + 5 \in O(n)$
            - $0.5n \log_3 n - 2n + 7 \in O(n \log n)$
            - $2^n + n^3 + 3n \in O(2^n)$
            - $5n^3 + 3n^2 + 6n \in O(n^3)$
            - $10^4 + 10^5 n + 10^6 \log n + 10n^4 \in O(n^4)$

- Note that $n \in O(n^2)$ and $n \in O(n \log n)$, but we want to find the lowest order $g(n)$ to show that the algorithm is fast.
- Algorithm L, usually called Linear Search, runs in $O(n)$ time
- Algorithm B, usually called Binary Search, runs in $O(\log n)$ time

Which of the following two functions grows faster?

a. $f_a(n) = n$
b. $f_b(n) = n \cdot \log_2 n$
c. Neither; they both grow equally fast.

- - B. For large values of n, f_b is always getting bigger than f_a

- $f \in O(g) \leftrightarrow (\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq n_0 \rightarrow f(n) \leq c \cdot g(n))$
  - $f(n)$ grows no faster than $g(n)$
  - $g(n)$ is an upper bound on the growth rate of $f(n)$
  - Rules for inequalities
    - When dealing with $<$, allowed to make the expression on the right-hand side bigger, but never smaller
    - When dealing with $>$, you are allowed to make the expression on the right-hand side smaller, but never bigger
  - Example: show that $f(n) = 8n^2 - 14n$ and $g(n) = 2n^3$, $f \in O(g)$
    - Proof:

      Choose $c = 1, n_0 = 1$.

      Assume an unspecified natural number n, that $n \geq n_0 = 1$.

      $c \cdot g(n) - f(n) = 2n^3 - 8n^2 + 14n = 2n(n^2 - 4n + 7)$.

      Since $n \geq n_0 > 0$, $2n \geq 0$

      And $n^2 - 4n + 7 = (n-2)^2 + 3 \geq 3 \geq 0$

      Thus $c \cdot g(n) - f(n) \geq 0$

      So $f(n) \leq c \cdot g(n)$

      Given the premise, we have proved the conclusion, thus $f \in O(g)$ by definition.
    - Proof:

      Consider $c = 4$ and $n_0 = 0$

      Consider an unspecified natural number $n \geq n_0$

      $8n^2 - 14n \leq 8n^2 \leq 8n^3$, for $n \geq 0$

      $8n^3 = 4 \cdot 2n^3$

      Therefore, $f(n) \leq 4 \cdot g(n)$, for $n \geq 0$

      ■
  - Example: show that $f(n) = 4n + 20$ and $g(n) = 2n^2 - 10n$, $f \in O(g)$
    - Proof:

      Consider $c = 1$ and $n_0 = 17$

Consider unspecified natural number $n \geq n_0$

$4n + 20 = 4n + 20 + 10n - 10n \leq 4n + 20n + 10n - 10n = 34n - 10n = 17 \cdot (2n) - 1$, for $n \geq 1$

$17 \cdot (2n) - 10n \leq 17 \cdot (2n) \cdot \frac{n}{17} - 10n = 2n^2 - 10n$, for $n \geq 17$

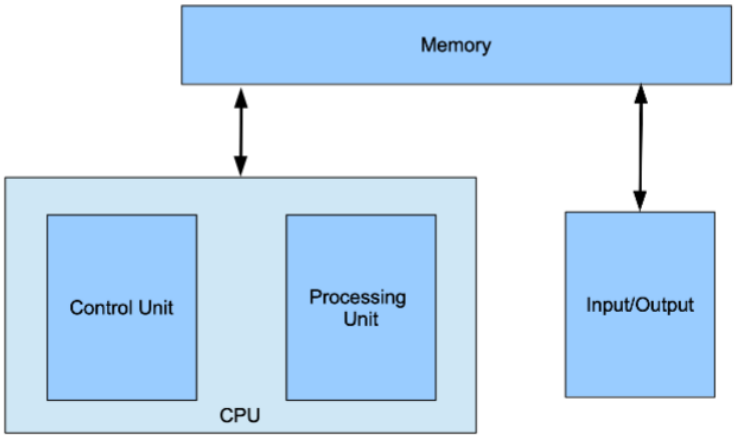
Thus $4n + 20 \leq 1 \cdot (2n^2 - 10n)$, for $n \geq 17$.

■

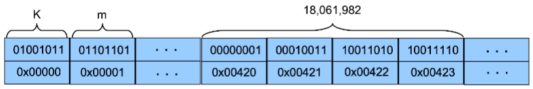
- Proving O
    - Start with c and n_0 blank
    - Do not assume $f(n) \leq c \cdot g(n)$ to be True
    - Change the right side of the equation to reach to $f(n) \leq c \cdot g(n)$
    - Go back and annotate the value of c and n_0 on the first line of the proof

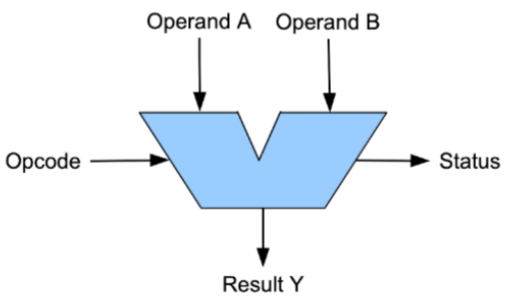
August 11th - Module 11 (Continued)

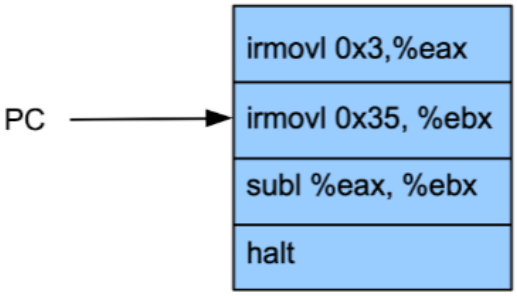
Module 11 (Continued)

- A working computer - Von Neumann Architecture
    - Processing Unit
        - Contains an arithmetic logic unit (ALU) and processor registers
    - Control Unit
        - Contains an instruction register and program counter
    - Memory
        - Stores data and instructions
    - External mass storage
    - Input and Output mechanisms
    - 

    - Component details
        - Memory
            - Contains both instructions and data
            - Divided into number of memory locations, like a list or array

        - Each memory location contains a fixed number of bits (most commonly 8 bits, or 1 byte)
        - Values that use more than 8 bits are stored in multiple consecutive memory locations
        - 

        - 32-bit computers can have addresses with 32 bits (2^{32} addresses, or 4GB); 64-bit computers can have addresses with 64 bits (2^{64} addresses, or 18EB $\approx 1.8 \cdot 10^{10}$ GB).
    - Arithmetic Logic Unit
        - Performs arithmetic and logical operations
        - Can detect overflow and other status
        - Opcode usually refers to the operation ($+, -, *, /$, and , or)
        - 

    - Control Unit
        - Decides which instructions to execute
        - Executes these instructions sequentially (not quite sure, but this is how it appears to the user)
        - Contains the Program Counter (PC), a special register that has the memory address of the next instruction
        - 

- Our working computer
    - Implements the design presented in the textbook by Bryant and O'Hallaron (CPSC 213/313)
    - Small subset of the IA32 (Intel 32-bit) architecture
    - Has
        - 12 types of instructions
        - 1 PC register
        - 8 general-purpose 32-bit registers, used for values that we are currently working with

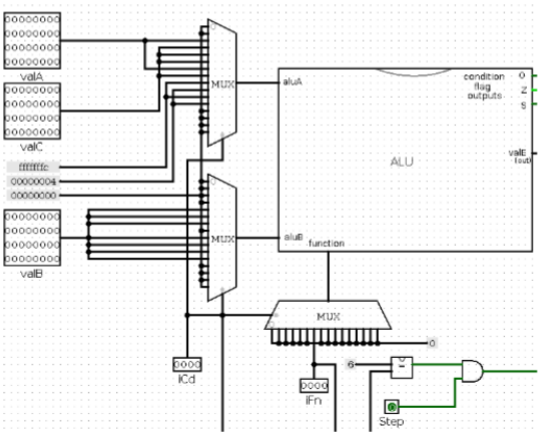
- **Register** is faster to access than memory and are used for values that the CPU are currently working with (usually copied from memory)
- Instructions
    - ```
      1 irmovl 0x1A, %ecx
      ```
        - Stores a constant in a register
        - The value $1A$ (26 in decimal) is stored in the register %ecx
    - ```
      1 subl %eax, %ebx
      ```
        - **subl** subtracts its arguments
        - ebx = ebx - eax
    - ```
      1 jge label
      ```
        - Conditional jump
        - Changes the value of **PC** according to the result of the last arithmetic or logic operation

      What does this sample program do?

      ```
      irmovl 0xE9,%eax
      irmovl 0x0C, %ebx
      addl %ebx, %eax
      halt
      ```

      a. Add values and looses the answer
      b. Move values to two registers
      c. Add values and save it to a register
      d. Impossible to tell
      e. It generates an error

    - C. Moves 233 into register %eax, moves 12 into register %ebx, adds the two and moves it to %ebx.
- Sample Program
    - ```
      1 label: irmovl 0x35, %eax
      2         irmovl 0x3, %ebx
      3         subl %eax, %ebx
      4         jge label
      5         halt
      ```
        - Moves 0x35 to %eax, eax = 53
        - Moves 0x3 to %ebx, ebx = 3

        - Subtracts eax from ebx and stores the result in ebx, ebx = ebx - eax = 3 - 53 = -50
        - Checks if the last result was greater or equal to 0, if (ebx $\geq$ 0) go to label
        - Since the result was negative, the next instruction is the sequential, so the program ends (halts). %ebx will be 0xCC, and %eax will be 0x35
- How does the computer know which instruction does what?
    - Each instruction is a sequence of 8 to 48 bits
    - Some of the bits tell it which instruction it is
    - Other bits tell it what operands to use
    - Bits are used as selects inputs for several **multiplexers**
`subl %eax, %ebx`	`irmovl 0xcafe, %ecx`	`jge label`
6 1 0 3	3 0 F 1 0000CAFE	7 5 0000CAFE
Arithmetic or logic operation	Move constant into register	Jump instruction
Subtraction	Ignored	Greater than or equal
eax	No register used	Address of label
ebx	ecx	
	Constant value	
    - 

    - ```
      label: irmovl 0x35, %eax
             irmovl 0x3, %ebx
             subl %eax, %ebx
             jge label
             halt
      ```
30	F0	00	00	00	35	30	F3	00	00	00	03	61	03	75	00	00	01	89	00
0x189	0x18A	0x18B	0x18C	0x18D	0x18E	0x18F	0x190	0x191	0x192	0x193	0x194	0x195	0x196	0x197	0x198	0x199	0x19A	0x19B	0x19C
    - Program starts at PC = 0x189
    - Value of label is 0x189
    - 3 0 F 0 0000 0035
        - ```
          irmovl 0x35 %eax
          ```

        - Stores 0x35 into register 0, and moves PC to label 0x18F
    - 3 0 F 3 0000 0003
        - ```
          1 irmovl 0x3 %ebx
          ```
        - Stores 0x3 into register 3, and moves PC to label 0x195
    - 6 1 0 3
        - ```
          1 subl %eax, %ebx
          ```
        - Subtracts the values in register 3 from register 0, and moves PC to label 0x197
    - 7 5 0000 0189
        - ```
          1 jge label
          ```
        - Jumps to either 0x189 or 0x19C depending on the value
    - 00
        - ```
          1 halt
          ```
        - Ends on halt
- CPU divided into 6 stages
    - Fetch: read instruction and decide on new PC value
    - Decode: read values from registers
    - Execute: use the ALU to perform computations
    - Memory: read data from or write data to memory
    - Write-back: store values into registers
    - PC update: store the new PC value

      What can we say about PC?

      a. It is always incremented by 4 each instruction
      b. It is only updated after some instructions
      c. It gives the memory address of the next instruction to be executed
      d. It is updated during the Memory stage of execution
      e. None of the above
    -
        - C. It is incremented based on the size of the instruction, it always must be updated after an execution.