Week 3 Announcements

Pre-class reading before Wednesday's class Please look over the annotations from pages 12-13 of the Week 2 notes on **signed-number representations**. We'll go over them in the first 10 minutes of class, but it'll be useful for you to look at them before you come.

https://discrete-math-for-cs.github.io/website/unit2.html

Homework 1 scores were published on Gradescope this evening. Overall, we saw many great solutions that are applying the data types and notations we're discussing precisely and in correct contexts. Please make sure to look at the personalized feedback for your homework that the TAs and tutors wrote for you.

---- Gradescope message ----

Here is your graded feedback for the first CSE 20 homework assignment. We encourage you to open up your homework submission in Gradescope and read through the rubric item descriptions and personalized comments left by the graders. We hope you will find this feedback useful. You may also benefit from looking at the sample solutions published on the assignment page: use the toggle labelled "Solutions" to see the PDF of solutions on the website.

<<link to your submission>>

Link not working? Please see our Why can't I see my grades? FAQ article for further information.

If you have never logged in to this site before, you can set your password. The login for your Gradescope account is minnes@eng.ucsd.edu. (If you have any problems accessing the site, please contact help@gradescope.com.)

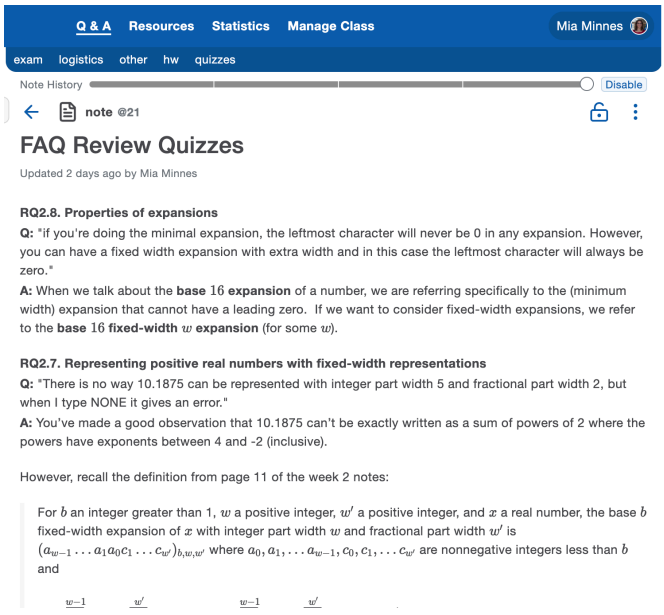
Letter grades for this homework assignment have the approximate cutoffs: 90% for A, 80% for B, 70% for C. Statistics for this homework:
Mean: 44.85
Median: 45.0
Standard Deviation: 4.47

Some reminders:
Precise use of terminology and notation is an important skill to practice; you can expect our attention to these details to continue for future homework.
For many of the question parts, there were multiple possible approaches. We recommend looking at the sample solutions or coming by during discussion to see additional strategies.

Our regrade policy (from the website syllabus) is "The window for requesting regrades once assignment feedback has been published will be set in Gradescope and will typically be 2-3 days. If you submit a regrade request, please include a brief but detailed explanation of why you think there was an error in the grading. A regrade request may lead to us reviewing the entire assignment and may lead to the score being adjusted up or down depending on any errors found in the original grading. All regrades will only be considered once the regrade window closes; thank you in advance for your patience while we carefully look through them."

Best wishes,
Prof. Minnes and the Winter 2026 CSE 20 team

(on Canvas)



(on Piazza)

Week 3 at a glance

We will be learning and practicing to:

- Model systems with tools from discrete mathematics and reason about implications of modelling choices. Explore applications in CS through multiple perspectives, including software, hardware, and theory.
  - Determining the properties of positional number representations, including overflow and bit operations
  - Connecting logical circuits and compound proposition and tracing to calcluate output values
- Translate between different representations to illustrate a concept.
  - Translating between symbolic and English versions of statements using precise mathematical language
- Use precise notation to encode meaning and present arguments concisely and clearly
  - Listing the truth tables of atomic boolean functions (and, or, xor, not, if, iff)
- Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems. Use mathematical techniques to solve problems. Determine appropriate conceptual tools to apply to new situations. Know when tools do not apply and try different approaches. Critically analyze and evaluate candidate solutions.
  - Evaluating compound propositions
  - Judging logical equivalence of compound propositions using symbolic manipulation with known equivalences, including DeMorgan's Law
  - Judging logical equivalence of compound propositions using truth tables
  - Rewriting compound propositions using normal forms
  - Judging whether a collection of propositions is consistent

TODO:

Review quiz based on Week 2 class material (extended because signed number representations weren't covered until Week 3; now due Monday 01/26/2026)

Review quiz based on Week 3 class material (due Monday 01/26/2026)

RQ2.1. Tracing the pseudocode for the algorithm to compute the integer part of base b logarithm

Complete the trace table for running the algorithm $logb$ for calculating the integer part of base b logarithm with inputs $b = 5$ and $n = 128$.

Fill in the missing values in the table below. For the condition column, enter **T** for True or **F** for False.

	b	n	i	$n > b - 1$?
Initial Value	5	128	0	T
After 1 iteration	5			T
After 2 iterations	5	5	2	
After 3 iterations	5	1	3	F

Definition: Algorithm for computing base b logarithm

Calculating integer part of base b logarithm

procedure $logb(b, n$: positive integers with $b > 1)$
  $i := 0$
  while $n > b - 1$
    $i := i + 1$
    $n :=$ n **div** b
  return i $\{i$ holds the integer part of the base b logarithm of $n\}$

Definition: Division algorithm

Let n be an integer and d a positive integer. There are unique integers q and r, with $0 \leq r < d$, such that $n = dq + r$. In this case, d is called the divisor, n is called the **dividend**, q is called the **quotient**, and r is called the **remainder**. Because these numbers are guaranteed to exist, the following functions are well-defined: **div** $: \mathbb{Z} \times \mathbb{Z}^+ \to \mathbb{Z}$ given by **div** $((n, d))$ is the quotient when n is the dividend and d is the divisor. **mod** $: \mathbb{Z} \times \mathbb{Z}^+ \to \mathbb{Z}$ given by **mod** $((n, d))$ is the remainder when n is the dividend and d is the divisor. Because these functions are so important, we sometimes use the notation n **div** $d =$ **div** $((n, d))$ and n **mod** $d =$ **mod** $((n, d))$.

Save & Grade *Single attempt* | Save only

Additional attempts available with new variants

Review Quiz 2

Assessment overview

Total points: 0/20

Score: 0%

Question RQ2.1

Value: 2

All variants: Open

Total points: — /2

Auto-graded question

Report an error in this question

Previous question | Next question

Personal Notes

No attached notes

Attach a file

Add text note

RQ2.7. Representing positive real numbers with fixed-width representations

For positive real numbers x, we have a definition for the **base** b **fixed-width expansion of** x **with integer part width** w **and fractional part width** w', for b an integer greater than 1, w a positive integer, and w' a positive integer. For example, the base 2 (binary) fixed-width expansion of 4 with integer part width 3 and fractional part width 2 is

$$(100.00)_{2,3,2}$$

Also, the base 2 (binary) fixed-width expansion of 4.1 with integer part width 3 and fractional part width 2 is

$$(100.00)_{2,3,2}$$

Give the binary fixed-width expansion of

$$27$$

with integer part width 5 and fractional part width 2.

$27 = ($ $)_{2,5,2}$

Note: Your answer must be a string of seven 0s and 1s and a period (radix point), with no other symbols or parentheses. All spaces will be removed from your answer. If the number cannot be represented with the given base and widths, enter **NONE**.

Note: some powers of 2 are $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32, 2^{-1} = 0.5, 2^{-2} = 0.25, 2^{-3} = 0.125, 2^{-4} = 0.0625, 2^{-5} = 0.03125$.

Give the binary fixed-width expansion of

$$1.90625$$

with integer part width 4 and fractional part width 3.

$1.90625 = ($ $)_{2,4,3}$

Note: Your answer must be a string of seven 0s and 1s and a period (radix point), with no other symbols or parentheses. All spaces will be removed from your answer. If the number cannot be represented with the given base and widths, enter **NONE**.

Note: some powers of 2 are $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^{-1} = 0.5, 2^{-2} = 0.25, 2^{-3} = 0.125, 2^{-4} = 0.0625$.

Definition: Fixed-Width Fractional Expansion

For b an integer greater than 1, w a positive integer, w' a positive integer, and x a real number, the **base** b **fixed-width expansion of** x **with integer part width** w **and fractional part width** w' is $(a_{w-1} \cdots a_1 a_0 . c_1 \cdots c_{w'})_{b,w,w'}$ where $a_0, a_1, \ldots, a_{w-1}, c_1, \ldots, c_{w'}$ are nonnegative integers less than b and

$$x \geq \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j}$$

and

$$x < \sum_{i=0}^{w-1} a_i b^i + \sum_{j=1}^{w'} c_j b^{-j} + b^{-w'}$$

Review Quiz 2

Assessment overview

Total points: 0/20

Score: 0%

Question RQ2.7

Value: 2

All variants: Open

Total points: — /2

Auto-graded question

Report an error in this question

Previous question | Next question

Personal Notes

No attached notes

Attach a file

Add text note

Week 3 Wednesday: Fixed-width Addition and Circuits

Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. *Does this give the right value for the sum?*

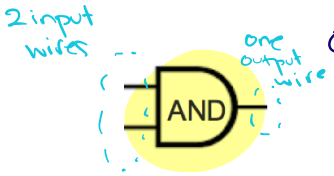
$$\begin{array}{r} {\scriptstyle 1} \\ (2026)_{10,4} \\ +(9025)_{10,4} \\ \hline (1051)_{10,4} \end{array}$$

Fixed width binary (width 6) | Sign magnitude (width 6) | 2s complement (width 6)

	Fixed width binary (width 6)	Sign magnitude (width 6)	2s complement (width 6)
Summand 1	$(1\,1\,0\,1\,0\,0)_{2,6}$	$[1\,1\,0\,1\,0\,0]_{s,6}$	$[1\,1\,0\,1\,0\,0]_{2c,6}$
Summand 2	$+(0\,0\,0\,1\,0\,1)_{2,6}$	$+[0\,0\,0\,1\,0\,1]_{s,6}$	$+[0\,0\,0\,1\,0\,1]_{2c,6}$
result	$1\,1\,1\,0\,0\,1$	$1\,1\,1\,0\,0\,1$	$1\,1\,1\,0\,0\,1$
Value of Summand 1	$(110100)_{2,6} = 52$ (32+16+4)	$[110100]_{s,6} = -20$ (Neg 16+4)	$[110100]_{2c,6} = -12$ (−32+16+4)
Value of Summand 2	$(000101)_{2,6} = 5$ (4+1)	$[000101]_{s,6} = 5$ (Pos 4+1)	$[000101]_{2c,6} = 5$ (4+1)
Value of result	$(111001)_{2,6} = 57$ (32+16+8+1)	$[111001]_{s,6} = -25$ (Neg 16+8+1)	$[111001]_{2c,6} = -7$ (−32+16+8+1)
	Fixed-width addition does give expected result	Fixed-width addition does not give expected result	Fixed-width addition does give expected result

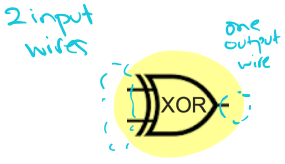
In a **combinatorial circuit** (also known as a **logic circuit**), we have **logic gates** connected by **wires**. The inputs to the circuits are the values set on the input wires: possible values are 0 (low) or 1 (high). The values flow along the wires from left to right. A wire may be split into two or more wires, indicated with a filled-in circle (representing solder). Values stay the same along a wire. When one or more wires flow into a gate, the output value of that gate is computed from the input values based on the gate's definition table. Outputs of gates may become inputs to other gates.

Inputs x	Inputs y	Output x AND y
1	1	1
1	0	0
0	1	0
0	0	0



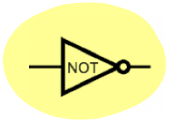
output is 1 when both inputs are 1

Inputs x	Inputs y	Output x XOR y
1	1	0
1	0	1
0	1	1
0	0	0



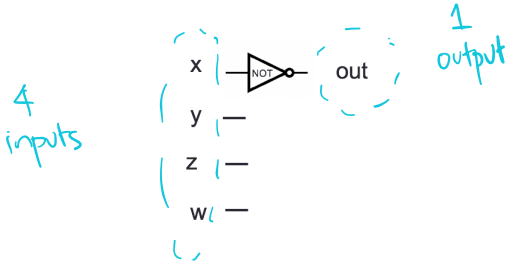
output is 1 when exactly one of inputs is 1

Input x	Output NOT x
1	0
0	1



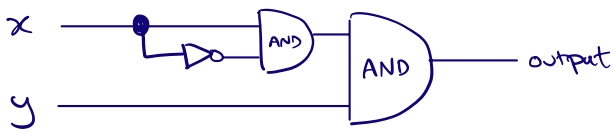
output is 1 when input is 0

Example digital circuit:



Output when $x = 1, y = 0, z = 0, w = 1$ is _____
Output when $x = 1, y = 1, z = 1, w = 1$ is _____
Output when $x = 0, y = 0, z = 0, w = 1$ is _____

Draw a logic circuit with inputs x and y whose output is always 0. *Can you use exactly 1 gate?*



Fixed-width addition: adding one bit at time, using the usual column-by-column and carry arithmetic, and dropping the carry from the leftmost column so the result is the same width as the summands. In many cases, this gives representation of the correct value for the sum when we interpret the summands in fixed-width binary or in 2s complement.

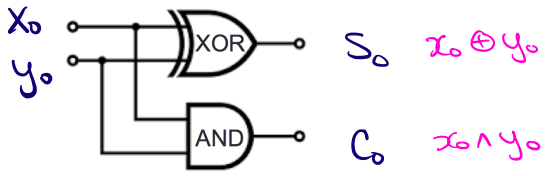
$c_0 = 0$

$$\begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array}$$

For single column:

Input		Output	
x_0	y_0	c_0	s_0
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

AND XOR



Draw a logic circuit that implements binary addition of two numbers that are each represented in fixed-width binary:

- Inputs x_0, y_0, x_1, y_1 represent $(x_1x_0)_{2,2}$ and $(y_1y_0)_{2,2}$
- Outputs z_0, z_1, z_2 represent $(z_2z_1z_0)_{2,3} = (x_1x_0)_{2,2} + (y_1y_0)_{2,2}$ (may require up to width 3)

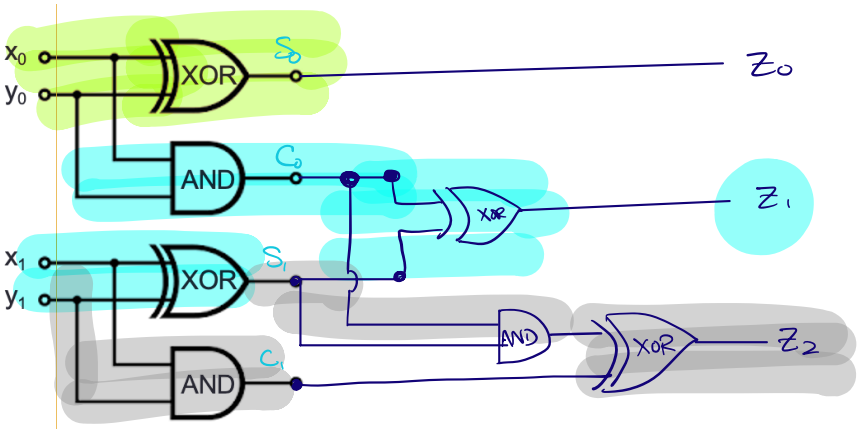
First approach: half-adder for each column, then combine carry from right column with sum of left column

Write expressions for the circuit output values in terms of input values:

$z_0 = x_0 \oplus y_0$

$z_1 = (x_1 \oplus y_1) \oplus (x_0 \wedge y_0)$

$z_2 = (x_1 \wedge y_1) \oplus ((x_1 \oplus y_1) \wedge (x_0 \wedge y_0))$



x_1	x_0	y_1	y_0	z_2	z_1	z_0
1	1	1	1			
1	1	1	0	1	0	1
1	1	0	1			
1	1	0	0			

15

14

$$\begin{array}{r} x_1\ x_0 \\ (1\ 1)_{2,2} \\ +\ (1\ 0)_{2,2} \\ \hline (1\ 0\ 1)_{2,3} \\ z_2\ z_1\ z_0 \end{array} \qquad c_0 = 1 \qquad \begin{array}{r} (1\ 1)_{2,2} \\ +\ (1\ 1)_{2,2} \\ \hline (1\ 1\ 0)_{2,3} \end{array}$$

There are other approaches, for example: for middle column, first add carry from right column to x_1, then add result to y_1

Week 3 Friday: Propositional Logic and Logical Equivalence

Logical operators aka propositional connectives

Conjunction	AND	$\wedge$	`\land`	2 inputs	Evaluates to T exactly when **both** inputs are T
Exclusive or	XOR	$\oplus$	`\oplus`	2 inputs	Evaluates to T exactly when **exactly one** of inputs is T
Disjunction	OR	$\vee$	`\lor`	2 inputs	Evaluates to T exactly when **at least one** of inputs is T
Negation	NOT	$\neg$	`\lnot`	1 input	Evaluates to T exactly when its input is F

Truth tables: Input-output tables where we use T for 1 and F for 0.

definition table

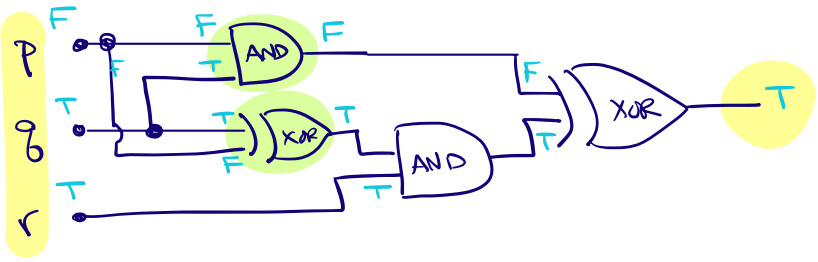
Input		Output		
		Conjunction	**Exclusive or**	**Disjunction**
p	q	$p \wedge q$	$p \oplus q$	$p \vee q$
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F
		AND	XOR	OR

$4 = 2^2$ rows

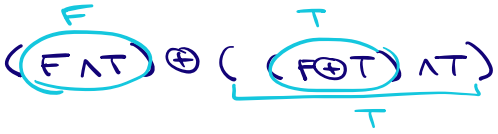
Input	Output
	Negation
p	$\neg p$
T	F
F	T
	NOT

$2 = 2^1$ row

Circuit



Input			Output	
p	q	r	$(p \wedge q) \oplus ((p \oplus q) \wedge r)$	$(p \wedge q) \vee ((p \oplus q) \wedge r)$
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	T	T
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F



Given a truth table, how do we find an expression using the input variables and logical operators that has the output values specified in this table?

Application: design a circuit given a desired input-output relationship.

Two possible output values T or F

Input		Output	
p	q	$mystery_1$	$mystery_2$
T	T	T	F
T	F	T	F
F	T	F	F
F	F	T	T

Avoid this row

Expressions that have output $mystery_1$ are

To avoid "p is F and q is T", I could make "p be T" or "q be F": $p \vee \neg q$

Alternatively: ok to be at top: $p \wedge q$; ok to be at 2nd: $p \wedge \neg q$; ok to be at bottom: $\neg p \wedge \neg q$

So mystery 1 is given by $(p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge \neg q)$

Expressions that have output $mystery_2$ are

Avoid top: $\neg p \vee \neg q$
Avoid 2nd: $\neg p \vee q$
Avoid 3rd: $p \vee \neg q$

mystery2 is given by $(\neg p \vee \neg q) \wedge (\neg p \vee q) \wedge (p \vee \neg q)$

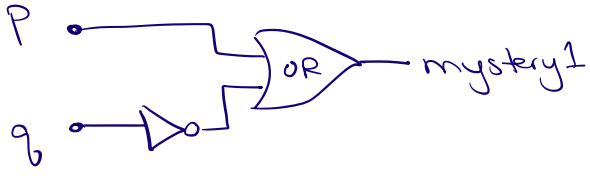
evaluates at $p=T, q=T$

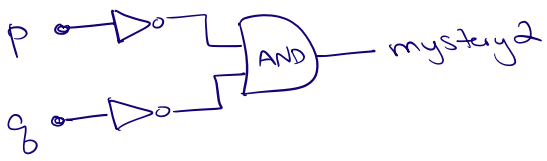
Alternatively: row where evaluate to T is $\neg p \wedge \neg q$

Idea: To develop an algorithm for translating truth tables to expressions, define a convenient **normal form** for expressions.

Definition An expression built of variables and logical operators is in **disjunctive normal form** (DNF) means that it is an OR of ANDs of variables and their negations. "Any row that evaluates to T"

Definition An expression built of variables and logical operators is in **conjunctive normal form** (CNF) means that it is an AND of ORs of variables and their negations. "Avoid all rows that evaluate to F"





Proposition: Declarative sentence that is true or false (not both).

Propositional variable: Variable that represents a proposition.

Compound proposition: New proposition formed from existing propositions (potentially) using logical operators. *Note*: A propositional variable is one example of a compound proposition.

Truth table: Table with one row for each of the possible combinations of truth values of the input and an additional column that shows the truth value of the result of the operation corresponding to a particular row.

Logical equivalence : Two compound propositions are **logically equivalent** means that they have the same truth values for all settings of truth values to their propositional variables.

Tautology: A compound proposition that evaluates to true for all settings of truth values to its propositional variables; it is abbreviated T.

Contradiction: A compound proposition that evaluates to false for all settings of truth values to its propositional variables; it is abbreviated F.

Contingency: A compound proposition that is neither a tautology nor a contradiction.

Label each of the following as a tautology, contradiction, or contingency.

$p \wedge p$ $\equiv p$ Contingency

$p \oplus p$ contradiction

$p \vee p$ $\equiv p$ contingency

$p \vee \neg p$ tautology

$p \wedge \neg p$ contradiction

Extra Example: Which of the compound propositions in the table below are logically equivalent?

Input		Output				
p	q	$\neg(p \wedge \neg q)$	$\neg(\neg p \vee \neg q)$	$(\neg p \vee q)$	$(\neg q \vee \neg p)$	$(p \wedge q)$
T	T	T	T	T	F	T
T	F	F	F	F	T	F
F	T	T	F	T	T	F
F	F	T	F	T	T	F

Output

$(p \wedge q) \oplus ((p \oplus q) \wedge r)$	$(p \wedge q) \vee ((p \oplus q) \wedge r)$
T	T
T	T
T	F
F	T
F	F
F	F
F	F
F	F

The propositions

$(p \wedge q) \oplus ((p \oplus q) \wedge r)$

and

$(p \wedge q) \vee ((p \oplus q) \wedge r)$

are logically equivalent!