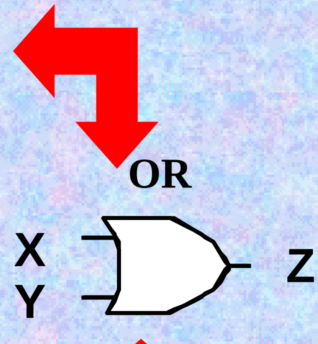
# Binary Arithmetic, ASCII, & Boolean Algebra

**Today:**

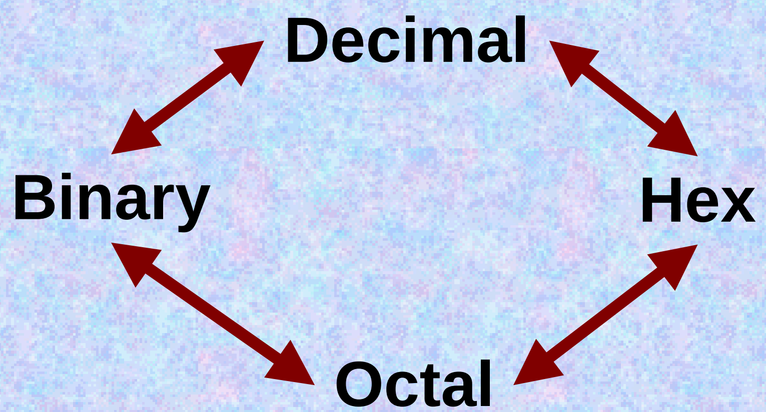
- **First Hour: Computer Arithmetic, Representation of Symbols**
  - **Representing Symbols – the ASCII code**
  - **Appendix A.2 & A.3 of Katz's Textbook**
  - **In-class Activity #1**
- **Second Hour: Boolean Algebra**
  - **Section 2.1 of Katz's Textbook**
  - **In-class Activity #2**

# Recap

| X | Y | X OR Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




Z = X + Y



# Binary Arithmetic

- You already know the rules for decimal addition and subtraction (how to handle sums, carries, differences, and borrows).

- Analogously, we develop the rules for binary addition and subtraction.

Taken from Appendix A.3

# Decimal Addition

**Refresher**

$$\begin{array}{rr} & 95_{10} \\ + & 16_{10} \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{rr} & 9 \times 10^1 + \ \ 5 \times 10^0 \\ + & 1 \times 10^1 + \ \ 6 \times 10^0 \\ \hline & 10 \times 10^1 + 11 \times 10^0 \end{array}$$

$$111_{10} = 1 \times 10^2 + (0+1) \times 10^1 + 1 \times 10^0$$

## Summary

$11$ ← Column carries

$$\begin{array}{rr} & 95_{10} \\ + & 16_{10} \\ \hline & 111_{10} \end{array}$$

# Binary Addition

**This table calculates the sum for pairs of binary numbers**

**0 + 0 = 0**

**0 + 1 = 1**

**1 + 0 = 1**

**1 + 1 = 0 with a carry of 1**

**Also known as the Half Adder Table**

# Binary Addition with Carry

**This table shows all the possible sums for binary numbers with carries**

| carry | | addend | | augend | | sum | |
|---|---|---|---|---|---|---|---|
| 0 | + | 0 | + | 0 | = | 0 | |
| 0 | + | 0 | + | 1 | = | 1 | |
| 0 | + | 1 | + | 0 | = | 1 | |
| 0 | + | 1 | + | 1 | = | 0 | with a carry of 1 |
| 1 | + | 0 | + | 0 | = | 1 | |
| 1 | + | 0 | + | 1 | = | 0 | with a carry of 1 |
| 1 | + | 1 | + | 0 | = | 0 | with a carry of 1 |
| 1 | + | 1 | + | 1 | = | 1 | with a carry of 1 |

**Also known as the Full Adder Table**

# Binary Addition

## Similar to the decimal case

**Example: Add 5 and 3 in binary**

| | | | | | |
|---|---|---|---|---|---|
| **1** | **1** | **1** | | | **(carries)** |
| | **1** | **0** | $\mathbf{1_2}$ | = | $\mathbf{5_{10}}$ |
| **+** | | **1** | $\mathbf{1_2}$ | = | $\mathbf{3_{10}}$ |
| **1** | $^{1}$**0** | $^{1}$**0** | $^{1}\mathbf{0_2}$ | = | $\mathbf{8_{10}}$ |

# Decimal Subtraction

**Refresher**

| | | | |
|---|---|---|---|
| | 9 | $^{1}5_{10}$ | $95 = 9x10^1 + 5x10^0 = 9x10^1 + 15x10^0$ |
| - | 1 | $6_{10}$ | $-16 = -1x10^1 + -6x10^0 = -1x10^1 + -6x10^0$ |
| - | 1 | | borrow = $-1x10^1$ |
| | 7 | $9_{10}$ | $7x10^1 + 9x10^0$ |

**Note: borrows are shown as explicit subtractions.**

# Binary Subtraction

**This table calculates the difference for pairs of binary numbers**

**0 - 0 = 0**

**0 - 1 = 1** with a borrow of 1

**1 - 0 = 1**

**1 - 1 = 0**

**Also known as the Half Subtractor Table**

# Binary Subtraction with Borrow

**This shows all the possibile differences for binary numbers with borrows**

| minuend | | subtrahend | | borrow | | difference | |
|---|---|---|---|---|---|---|---|
| 0 | - | 0 | - | 0 | = | 0 | |
| 0 | - | 0 | - | 1 | = | 1 | with a borrow of 1 |
| 0 | - | 1 | - | 0 | = | 1 | with a borrow of 1 |
| 0 | - | 1 | - | 1 | = | 0 | with a borrow of 1 |
| 1 | - | 0 | - | 0 | = | 1 | |
| 1 | - | 0 | - | 1 | = | 0 | |
| 1 | - | 1 | - | 0 | = | 0 | |
| 1 | - | 1 | - | 1 | = | 1 | with a borrow of 1 |

**Also known as the Full Subtractor Table**

# Binary Subtraction

**Similar to the decimal case**

**Example: Subtract 3 from 5 in binary**

$$
\begin{array}{rccclc}
 & 1 & {}^{1}0 & 1_2 & = & 5_{10} \\
- & & 1 & 1_2 & = & 3_{10} \\
- & 1 & & & \text{(borrows)} & \\
\hline
 & 0 & 1 & 0_2 & = & 2_{10}
\end{array}
$$

# Representing Symbols

**American Standard Code for Information Interchange**

**7-bits per symbol**

**$2^7$ = 128 different symbols**

- **26 uppercase letters (A-Z)**
- **26 lowercase letters (a-z)**
- **10 digits (0-9)**
- **1 blank space (SP)**
- **32 special-character symbols**
- **32 non-printing control characters**
- **1 delete character (DEL)**

# The 7-bit ASCII Code

Leftmost 3 bits

Rightmost 4 bits

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | HT | EM | ) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ECS | + | ; | K | [ | k | { |
| C | FF | FS | , | < | L | \ | l | \| |
| D | CR | GS | - | = | M | ] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | DEL |

# ASCII Example

**Example string: "I am here @ RPI\r\n"**

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | SP | a | m | SP | h | e | r | e | SP | @ | SP | R | P | I | CR | LF | NUL |
| 49 | 20 | 61 | 6D | 20 | 68 | 65 | 72 | 65 | 20 | 40 | 20 | 52 | 50 | 49 | 0D | 0A | 00 |

**(hexadecimal notation)**

# Control Codes

**Control codes are non-printing**

- **How do you type them on a conventional keyboard?**
- **For example, how do you get the ESC control if your keyboard doesn't have such a key?**
- **The Ctrl key forces the 2 most significant bits to 00**
- **Hold the Control key, Ctrl, and type [ to get Ctrl-[**
- **[ = $5B = 101 1011 → 001 1011 = $1B = ESC**
- **Similarly, Ctrl-c changes 110 0011 to 000 0011 (ETX)**

# Do Activity #1 Now

- Learn to add and subtract binary numbers
- Get to know the ASCII code

**Leftmost 3 bits**

**Rightmost 4 bits**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | HT | EM | ) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ECS | + | ; | K | [ | k | { |
| C | FF | FS | , | < | L | \ | l | \| |
| D | CR | GS | - | = | M | ] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | DEL |

# Boolean Algebra

A set of theorems for manipulating Boolean variables.

They are useful because they

- help simplify circuits to reduce cost
- help debug circuits
- help us with “reverse engineering”
- help us with “re-engineering”

Most of them are similar to ordinary algebra, but a few are very different.

# Theorems – Set #1

**Operations with 0 and 1**

1. X + 0 = X — 1D. X • 1 = X
2. X + 1 = 1 — 2D. X • 0 = 0

**Idempotent Law**

3. X + X = X — 3D. X • X = X

**Involution Law**

4. $\overline{\overline{X}} = X$

**Laws of Complementarity**

5. $X + \overline{X} = 1$ — 5D. $X \bullet \overline{X} = 0$

**Commutative Law**

6. X + Y = Y + X — 6D. X • Y = Y • X

**Associative Law:**

7. (X + Y) + Z = X + (Y + Z) = X + Y + Z — 7D. (X • Y) • Z = X • (Y • Z) = X • Y • Z

# Duality

**The dual of a Boolean equation is derived by replacing**
**AND operations by ORs,**
**OR operations by ANDs,**
**constant 0s by 1s, and**
**1s by 0s (literals are left unchanged).**

**For any equation that is true, its dual is also true!**

**Example:**

**X + 0 = X**

**Dual equation: X • 1 = X**

**Use duality to derive laws 1D – 7D from 1 – 7 !**

# Theorems – Set #2

Freaky!!

**Distributive Law**

8. $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$

8D. $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$

**Simplification Theorems**

9. $X \bullet Y + X \bullet \overline{Y} = X$

9D. $(X + Y) \bullet (X + \overline{Y}) = X$

10. $X + (X \bullet Y) = X$

10D. $X \bullet (X + Y) = X$

11. $(X + \overline{Y}) \bullet Y = X \bullet Y$

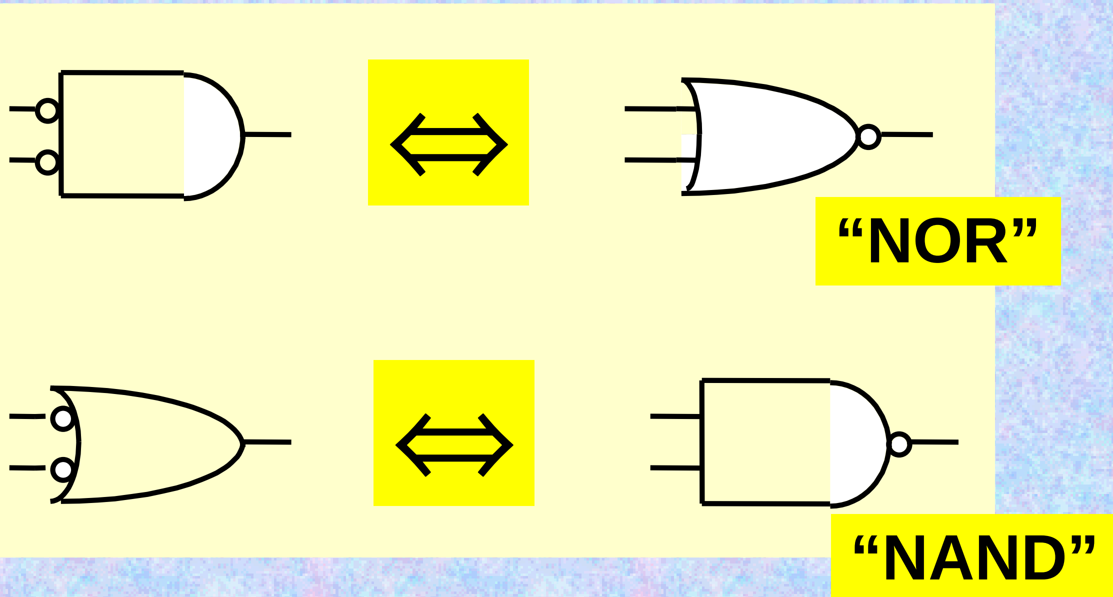
11D. $(X \bullet \overline{Y}) + Y = X + Y$

# Theorems – Set #3

## De Morgan's Laws

$$\overline{X}\bullet\overline{Y} = \overline{(X+Y)}$$

$$\overline{X}+\overline{Y} = \overline{(X\bullet Y)}$$



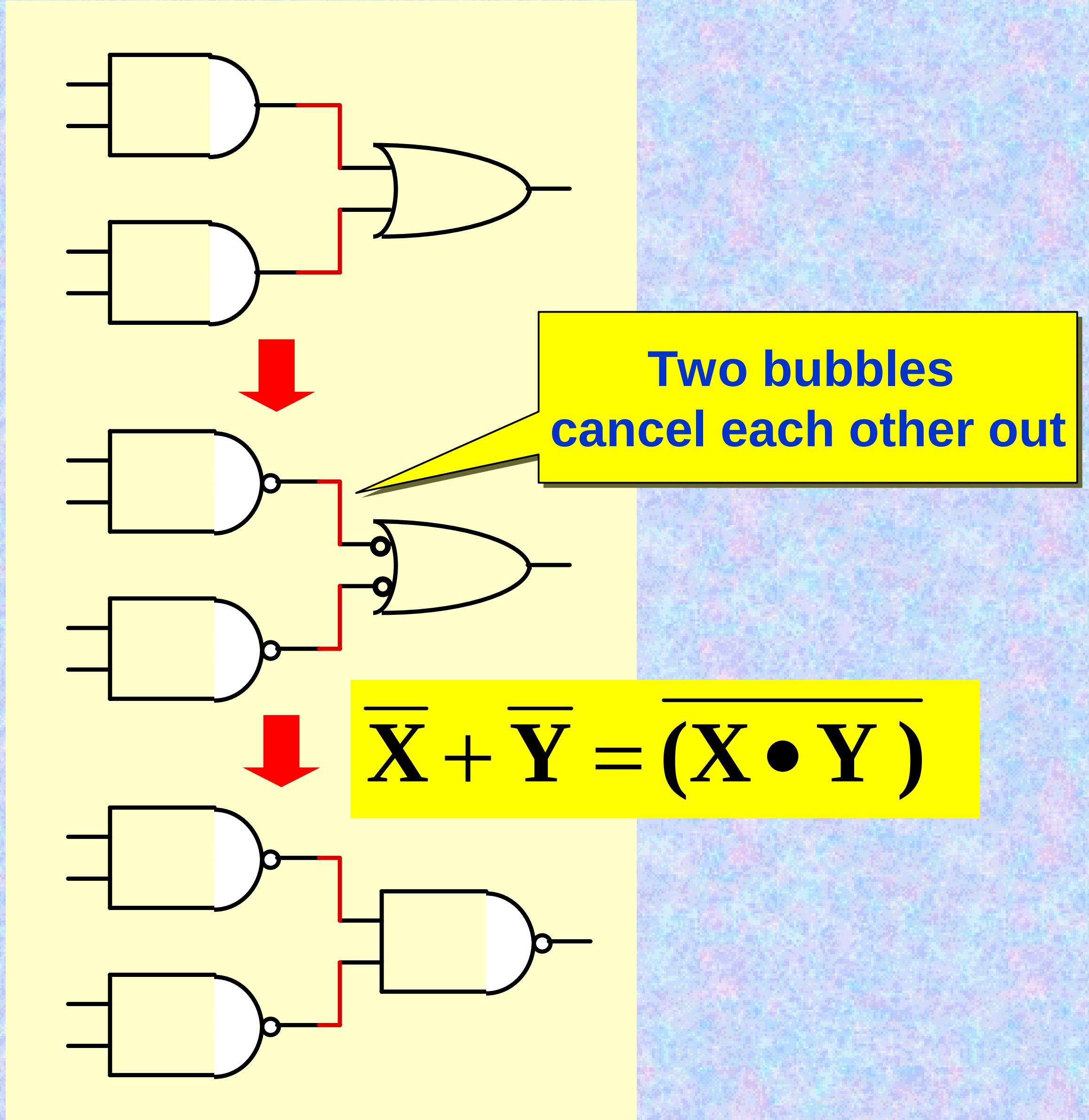

**In general…**

12. $\overline{(X + Y + Z + ...)} = \overline{X} \bullet \overline{Y} \bullet \overline{Z} \bullet ...$

12D. $\overline{(X \bullet Y \bullet Z \bullet ...)} = \overline{X} + \overline{Y} + \overline{Z} + ...$

13. $\overline{F(X_1, X_2, ..., X_n, 0, 1, +, \bullet)}$
$= F(\overline{X_1}, \overline{X_2}, ..., \overline{X_n}, 1, 0, \bullet, +)$

# DeMorgan's with Bubbles




$$\overline{X} + \overline{Y} = \overline{(X \bullet Y)}$$

**"All NAND" circuit has the same topology as the AND-OR**

# DeMorgan's Law

**Use to convert AND/OR expressions to OR/AND expressions**

**Examples:**

$$Z = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

$$\overline{Z} = \overline{(\overline{A}\,\overline{B}C)} \bullet \overline{(\overline{A}BC)} \bullet \overline{(A\overline{B}C)} \bullet \overline{(AB\overline{C})}$$

$$\overline{Z} = (A + B + \overline{C}) \bullet (A + \overline{B} + C) \bullet (\overline{A} + B + \overline{C}) \bullet (\overline{A} + \overline{B} + C)$$

# Theorems - Set 4

**Duality**

**14.** $(X + Y + Z + \ldots)^D = X \bullet Y \bullet Z \bullet \ldots$

**14D.** $(X \bullet Y \bullet Z \bullet \ldots)^D = X + Y + Z + \ldots$

**15.** $\{F(X_1, X_2, \ldots, X_n, 0, 1, +, \bullet)\}^D = \{F(X_1, X_2, \ldots, X_n, 1, 0, \bullet, +)\}$

**Multiplying and Factoring Theorems**

**16.** $$(X + Y) \bullet (\overline{X} + Z) = X \bullet Z + \overline{X} \bullet Y$$

**16D.** $$X \bullet Y + \overline{X} \bullet Z) = (X + Z) \bullet (\overline{X} + Y)$$

**Consensus Theorem**

**17.** $$XY + YZ + \overline{X}Z = XY + \overline{X}Z$$

**17D.** $$(X + Y)(Y + Z)(\overline{X} + Z) = (X + Y)(\overline{X} + Z)$$

**Watch out! Cancellation does not work in Boolean algebra!**

**Qn: How do we know if these theorems work?**

# Proving Theorems

**Boolean Algebra:**

| | |
|---|---|
| **E.g., prove the theorem:** | $X \cdot Y + X \cdot \overline{Y} = X$ |
| **distributive law (8)** | $X \cdot Y + X \cdot \overline{Y} = X \cdot (Y + \overline{Y})$ |
| **complementary law (5)** | $X \cdot (Y + \overline{Y}) = X \cdot (1)$ |
| **identity (1D)** | $X \cdot (1) = X$ |

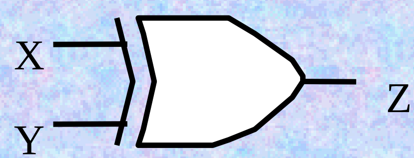
| | |
|---|---|
| **E.g., prove the theorem:** | $X + X \cdot Y = X$ |
| **identity (1D)** | $X + X \cdot Y = X \cdot 1 + X \cdot Y$ |
| **distributive law (8)** | $X \cdot 1 + X \cdot Y = X \cdot (1 + Y)$ |
| **identity (2)** | $X \cdot (1 + Y) = X \cdot (1)$ |
| **identity (1)** | $X \cdot (1) = X$ |

# Other Useful Functions

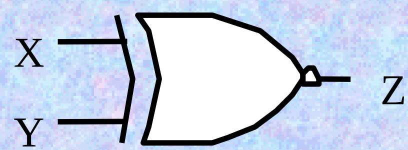
**There are 16 possible unique functions of 2 variables**

| X | Y | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | 0 | $X \cdot Y$ | | $X$ | | $Y$ | XOR | $X + Y$ | NOR | XNOR | $\overline{Y}$ | | $\overline{X}$ | | NAND | 1 |

**XOR** $X \oplus Y = X\overline{Y} + \overline{X}Y$




**XNOR** $\overline{X \oplus Y} = XY + \overline{X}\,\overline{Y}$




**New and useful gates**

# Do Activity #2 Now

**Due: End of Class Today**

**RETAIN THE LAST PAGE (#3)!!**

# For Next Class:

- **Bring Randy Katz Textbook**
  - **Electronic copy on website will disappear Friday.**
- **Required Reading:**
  - **Sec 2.2 & 2.3 of Katz**
- **This reading is necessary for getting points in the Studio Activity!**