



Unplugged CS Activity Cards

Free Library · Teach CS with no devices

Real computer science with zero screens. Six full activities teach the ideas behind coding using paper, string, and each other: algorithms, sorting, binary, functions, debugging, and networks. This teacher guide has a mini-guide and answer key for every activity; the student worksheets and a binary reference sheet are the student pages. Edit any of it in the Word version.

HOW TO RUN THESE ACTIVITIES

Each activity is self-contained and takes one class period or less. Run them in any order, or use them as warm-ups across a unit. The pattern is always the same: read the short teacher guide, gather the everyday materials, run the activity, then have students complete the worksheet so the thinking gets recorded.

THE SIX ACTIVITIES AT A GLANCE

TEACHER

Activity	CS concept it teaches	Materials	Time
1. Algorithm as a recipe	Sequencing and precise instructions	Paper, pencil	20 min
2. Human sorting network	Sorting algorithms and comparisons	Number cards, floor tape	25 min
3. Binary bracelets	Binary numbers and encoding	Beads or bead template, string	30 min
4. Function machine	Functions, inputs, and outputs	Paper machine, slips	20 min
5. Debugging maze	Debugging and tracing code	Grid maze handout, pencil	25 min
6. Be the network	Packets, routing, and protocols	Index cards, room space	30 min

HOW TO USE IT

Assign one activity per day for a two-week no-device rotation, or drop a single activity into any lesson as an unplugged break. Each teacher mini-guide names the concept and the directions so you can run it cold. The matching student worksheet is where they record and reflect.

PRINT TIP

Print the Student Pages PDF single-sided so a student never gets a teacher guide or answer key on the back of their worksheet. This teacher guide carries the store name; the student worksheets do not.

Concept it teaches

An algorithm is a precise, ordered list of steps. Computers do exactly what you say, in order, with no guessing. Students learn that a vague step breaks the whole program.

Directions

- 1** Pick a simple task everyone knows: making a peanut butter sandwich, brushing teeth, or drawing a smiley face.
- 2** Students write the algorithm as numbered steps. Tell them to assume the computer knows nothing.
- 3** Trade papers. Each student becomes a literal robot and acts out **ONLY** what is written, exactly as written.
- 4** When a robot gets stuck (the jar is closed, the pencil never lifts), that is a bug. The author fixes the step.
- 5** Debrief: which missing detail caused the funniest failure? That is why precision matters in code.

Timing

5 min setup, 10 min writing and acting out, 5 min debrief. Total about 20 minutes.

EXTENSION

Add a loop: 'repeat step 4 until the glass is full.' Add a condition: 'if the bread is torn, get a new slice.' Now students are writing loops and if-statements without knowing it.

Concept it teaches

Sorting is one of the most common jobs a computer does, and there are many ways to do it. Students feel how an algorithm makes decisions by comparing two things at a time and swapping.

Directions

- 1 Give six students a number card each (use the answer key for a clean set). They line up in random order.
- 2 Run a bubble sort: compare the first two students. If the left number is bigger, they swap places.
- 3 Slide down one spot and compare the next pair. Keep going to the end of the line. That is one pass.
- 4 Repeat passes until a full pass happens with zero swaps. The line is now sorted.
- 5 Count the swaps and passes as a class and record them on the worksheet.

Timing

5 min setup, 15 min running two or three rounds with different start orders, 5 min debrief. About 25 minutes.

EXTENSION

Try a sorting network: set students in fixed positions and only allow specific pairs to compare at the same time. Compare how many rounds it takes versus bubble sort.

ACTIVITY 2 · ANSWER KEY

ANSWER KEY · TEACHER

Use this clean set of number cards so the sort has a known answer. The starting order below sorts in exactly the swaps and passes shown.

Number cards to hand out	Sorted result	Bubble sort passes	Total swaps
3, 6, 1, 5, 2, 4	1, 2, 3, 4, 5, 6	4 passes	8 swaps

Bigger lists need many more swaps because bubble sort can move a value only one spot per pass. This is why students feel it get slow as the list grows.

Concept it teaches

Computers store everything as ones and zeros. Binary is just counting with two symbols. Students encode the first letter of their name (A=1, B=2, ... Z=26) as a 5-bit pattern of beads.

Directions

- 1 Show the place values from right to left: 1, 2, 4, 8, 16. Each bead is either ON (dark) or OFF (light).
- 2 Students find their letter number using A=1 through Z=26 (a light letter chart is on the worksheet).
- 3 They build that number from the place values. Example: M is $13 = 8 + 4 + 1$, so the pattern is 01101.
- 4 String five beads: dark bead means 1, light bead means 0. Read left to right as the 16-8-4-2-1 columns.
- 5 Trade bracelets and decode a partner's letter using the student reference sheet.

Timing

5 min intro, 20 min encoding and beading, 5 min decode a partner. About 30 minutes.

NO BEADS? NO PROBLEM

Fill-in circles work just as well: shade a bubble for 1, leave it blank for 0. The worksheet has a bead template so the activity runs with only paper and pencil.

ACTIVITY 3 · ANSWER KEY

ANSWER KEY · TEACHER

The first letter of the name encodes to these 5-bit patterns. Read left to right as 16-8-4-2-1.

Letter	Number	Binary	Letter	Number	Binary
A	1	00001	N	14	01110
E	5	00101	O	15	01111
H	8	01000	R	18	10010
I	9	01001	S	19	10011
M	13	01101	T	20	10100

Full A to Z: A=1 through Z=26. Numbers above 16 need the leftmost column, so patterns can start with a 1 in the 16 place. See the student reference sheet for the conversion method.

Concept it teaches

A function takes an input, follows a rule, and returns an output. Same input always gives the same output. Students run inputs through a secret rule and reverse-engineer it, just like reading code.

Directions

- 1 One student is the machine and picks a secret rule, such as 'multiply by 2' or 'add 3'.
- 2 Other students feed in input numbers on slips of paper. The machine returns the output for each one.
- 3 The class records input and output pairs in a table and hunts for the pattern.
- 4 When someone thinks they know the rule, they predict the output for a brand new input before it is run.
- 5 Reveal the rule. Write it as a function: output = (something done to) input.

Timing

5 min setup, 12 min running two or three secret rules, 3 min naming the pattern. About 20 minutes.

EXTENSION

Chain two machines so the output of the first becomes the input of the second. That is function composition, and it is exactly how real programs pipe data from one step to the next.

ACTIVITY 4 · ANSWER KEY

ANSWER KEY · TEACHER

Sample secret rules and how to check student answers.

Secret rule	Input 3	Input 5	Input 10
output = input x 2	6	10	20
output = input + 3	6	8	13
output = input x 2 + 1	7	11	21

Concept it teaches

Debugging is finding and fixing the step that makes code go wrong. Students trace a given list of moves through a grid, find where it crashes into a wall, and rewrite the broken command.

Directions

- 1 Students use the grid and the buggy program on the worksheet. Moves are U, D, L, R (up, down, left, right).
- 2 Starting at the S square, they trace each command one at a time, drawing the path in pencil.
- 3 Somewhere the path hits a wall or leaves the grid. That is the bug. Circle the exact command that fails.
- 4 Rewrite that one command (or add a missing one) so the path reaches the goal square G.
- 5 Trade and verify: a partner traces the corrected program to confirm it now reaches G.

Timing

5 min intro, 15 min tracing and fixing, 5 min partner check and debrief. About 25 minutes.

TEACHING MOVE

Insist students trace by hand and never guess. 'Reading the code line by line' is the whole skill. The answer key below shows the bug and the corrected program.

ACTIVITY 5 · ANSWER KEY

ANSWER KEY · TEACHER

The buggy program was: R, R, R, D, D, D, R, D. Start S is the top-left corner. Command 4 (the first D) drives into a wall. One correct fix that reaches G:

CORRECTED PROGRAM

D, D, R, R, D, D, R, R. This path steps around the shaded wall squares and lands on G in the bottom-right corner. Any move list that reaches G without crossing a wall is acceptable.

Concept it teaches

The internet breaks a message into small packets, sends them across nodes, and reassembles them at the destination. Students become routers and pass numbered packets across the room.

Directions

- 1 Spread students out as network nodes. Pick a sender and a receiver on opposite sides of the room.
- 2 The sender writes a short message and splits it into packets: one word per index card, numbered in order.
- 3 Each packet also gets the receiver's name (the address). Nodes pass packets hand to hand toward the receiver.
- 4 Deliberately shuffle the delivery order or hold one packet back to show packets can arrive out of order or late.
- 5 The receiver sorts packets by number and reassembles the message. Missing a packet? Request a resend.

Timing

5 min setup, 20 min running two message rounds with different disruptions, 5 min debrief. About 30 minutes.

EXTENSION

Block a node to force packets down a longer path. That is rerouting, and it is why the internet keeps working when one connection goes down.

ACTIVITY 6 · ANSWER KEY**ANSWER KEY · TEACHER**

There is no single answer; check that the receiver reassembled packets in number order and that students noticed packets can arrive late or out of order. A missing packet is fixed with a resend request, which is exactly how real network protocols recover lost data.

WANT THE FULL CS UNIT?

These cards are a taste. The editable computer science and coding curriculum in the shop builds the lessons, slides, and assessments around these ideas so you can teach an entire unplugged-to-plugged unit start to finish.