Binary Sorting and Searching

Lesson Focus

In ”Binary for the Younger Set”, or other introductory Binary lessons (see resource attached) we learned what Binary is, how to represent numbers or letters as binary number patterns of 1 and 0. In this lesson we will see how we can use the simple on/off 1/0 property of binary to easily:

- Search for information, once it has been “coded” in binary for storage
- Perform simple computer program tasks such as sorting, but use binary

Lesson Synopsis

Age Levels

5-12

Objectives

This lesson reinforces students’ ability to write, read, and compare small binary numbers. It also shows how the simple properties of binary can be used to do useful tasks quickly. The lesson introduces beginning students to the simple properties of binary logic – “compare”, AND, OR, NOT.

Anticipated Learner Outcomes

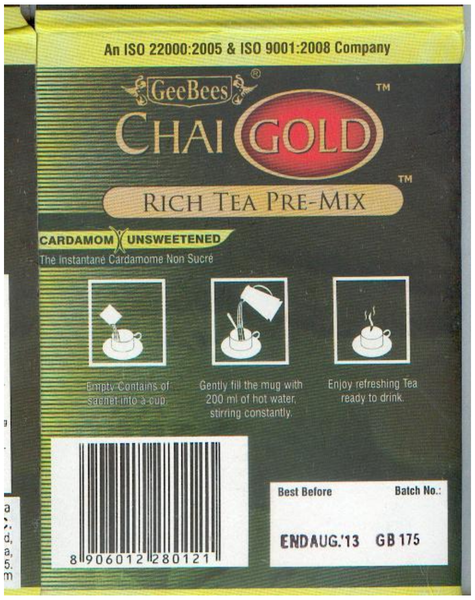
Students, after completing the lesson should be able to:

- Describe what “information storage” is
- Describe what encoding information involves and why it is useful (i.e. to look for information quickly)
- Describe how to express “queries” to look for information
- Describe what sorting information is and a simple ways to do it.
- Describe simple ways to “code “ information which can be retrieved using simple binary queries about the information
- Describe a simple way to sort information (bubble sort)
- Write the binary numbers from 0 to 9 (0000b to 1001b).
- Perform simple binary operations on numbers from 0 to 9 (NOT, AND, OR, Compare)

Lesson Activities

Sorting

Motivation. Many places in the world we want to sort things either numerically or alphabetically. Can students think of 7 good examples? Here is one to get started – the widespread "UPC" code on a box (or its invisible equivalent the "RFID" code) is used at grocery checkouts to lookup the price for that item, as well as in the manufacturing. This box of tea was routed around the factory getting filled up with packages of tea, but at the end of the production line, boxes that are all the same get put together into larger boxes to send to the store. This is the most simple example of sorting, which is "selection". More complicated examples might sort items like bolts and nuts by size.


An ISO 22000:2005 & ISO 9001:2008 Company
GeeBees
CHAI GOLD
RICH TEA PRE-MIX
CARDAMOM UNSWEETENED
The Instantane Cardamome Non Sucré
Empty Contains of Sachet into a cup.
Gently fill the mug with 200 ml of hot water, stirring constantly.
Enjoy refreshing Tea ready to drink.
8 906012 280121
Best Before
Batch No.:
END AUG.'13
GB 175

In this class exercise, we are going to sort a group of students standing in line at the front of the class by age, But , just for fun we will use their ages in Binary.

1. The class as a group reviews how to represent numbers from 0-16 in binary. For example "7" is 0111b = 0x8 + 1x4 + 1x2 + 1x1

    2. Each student takes a sheet of paper, and with a marker writes their age in Binary in large 1s and 0s (large enough for others to read from a distance). Note: if there is no "variation" in the ages, the students and teachers suggest changing some students ages so there is a good range of ages. Students should write their "fictitious" age in binary on the back of their sheet and cross out their actual age on the other side.
    3. Students (or up to 10 students at a minimum, if a large class would be too hard to fit at with their age (or "adjusted" age) in front of them.

4. Here is an example, with ages from 0 to 16:
   0011 0101 0101 0100 1001 0111 1011 0001 1000 0111 (binary)
   3 5 5 4 9 7 11 1 8 7 (decimal)
5. The fun here is that all of the numbers are written binary, so the class can have some fun reinforcing their understanding of binary.
   a. Who is the "youngest" student? With binary, we look for "leading" 0s – the more 0s on the left of the number, the smaller it is.
   b. Who is the "oldest" . With binary, we look for "leading" 1s - the more 1s in the leftmost binary positions, the bigger the number is
6. Now we do a "bubble sort" which is a simple way to get everybody sorted by age, with the youngest on the left when viewed from the front (class), oldest on the right.
   a. Leftmost student (viewed from the front of the line) checks with the student on THEIR left – if that student is younger, change places. How do we know which one is younger?
      i. Start the left of each binary number. If digits are the same (1=1 or 0=0) go to the next step. If one digit is 0 and the other is a 1, then that number is smaller (younger age).
      ii. Move one binary digit to the right. Repeat step 1 unless you are on the rightmost binary digit. In that case both numbers are identical, do not change places

It is worth noting that electronic circuits in computers do basically this, but using a set of "logic gates" that do the comparison of the 1s and 0s VERY QUICKLY..

   b. Student to the right repeats step (a), unless they are the last student, in which case go back to the leftmost student and repeat.
   c. The class will begin to see that "younger" students are slowly moved to the left as a result of this process, until eventually all the students are lined up with the youngest on the left, and the oldest on the right.
7. When we are all done with the example we would have gotten:
   0001 0011 0100 0101 0101 0111 0111 1000 1001 1011 (binary)
   1 3 4 5 5 7 7 8 9 11 (decimal)

8. This is actually a VERY SLOW way to do the sorting. If students are interested in how to speed things up, they can check the resources. HINT: the simplest way to get some improvement is to start checking backward from right to left after having done one "sweep" of comparisons from left to right (then sweeping back again from left to right etc.). This is called a SHAKER SORT.

So – what did we learn?

- We reviewed writing small binary numbers

- We learned how to COMPARE binary numbers using the "scan bits left to right" method. Just like learning Decimal numbers, this becomes easier once we recognize the "pattern" of the binary numbers, with more weight being given to 1s on the left
- We saw how to do a "bubble sort" to sort a group of items according to some CRITERION – in this case student age.
- We "worked" with binary numbers – i.e. the age of the students by doing comparisions and "executing" the "algorithm" of BUBBLE SORT on the row of students to get them sorted by age.

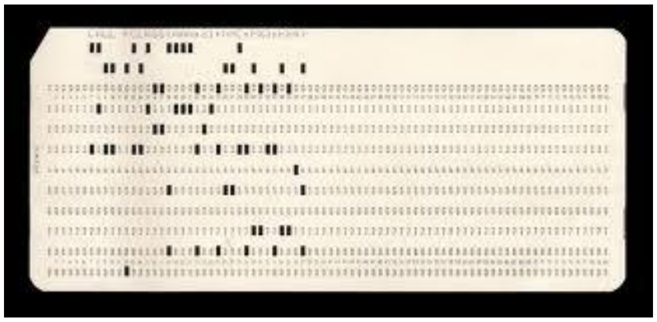
Questons

- Suppose you also wanted to sort your row of students by eye colour, such that: BLUE is on the left, BROWN is on the right, and everything else is in the middle? HINT: break the problem down using BINARY. Does a student have BLUE eyes (YES/NO). To get all the BROWN-eyed students at the right of the line, how should the BINARY number to represent BROWN eyes be coded?
- What are some examples of everyday items which use BINARY code? Is the code "visible" like the UPC bar-code or "invisible" ?
- Who invented BINARY code?
- Can you execute bubble sort using a pack of cards, and use just the numbered (Ace to 9 cards) in one suite – like SPADES laid out in random order in a row to start? You can also use the Jack, Queen, and King if you remember by convention Jack=10, Queen=11 and King = 12. If you don't have cards handy you can see it on YouTube.
- Try to use cards of 4 different suites and numbers, and sort so that all cards on one suite end up together. Obviously you can do this without going through the whole "bubble sort" (although if you try "bubble sort" by suite with say SPADE=1, DIAMOND=2, HEART=3, CLUB=4, you will find it does work). What kind of BINARY COMPARISON is going on inside your head to be able to quickly put them in proper suites? (ANS: likely 4 separate selection sorts, "Is this a SPADE?", "Is this a DIAMOND"?, etc.).
- Describe some better ways of sorting? HINT see http://www.sorting-algorithms.com, or other sites like YouTube where lines of folk dancers show the way to swap places and get things sorted faster
- Explore on the Web how binary "bits" can be stored in atoms, and used to make "Quantum Computers".

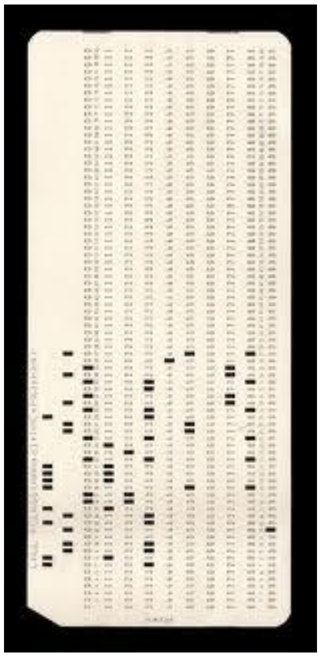
Searching

Motivation. Information is being stored and retrieved all the time. In order to retrieve or sort items (including information), we use KEYS which are the characteristics of items. In the last example, for the items=STUDENTS, we used KEYS of AGE (a binary number), and EYE COLOUR (also basically a binary number, but where each individual BIT might represent a colour like BLUE=0 or BROWN=1.

In the early days of “computing before computers”, a lot of information was stored on INDEX CARDS. In modern days, these INDEX CARDS are stored electronically as DATABASE RECORDS. Here is an example of such an INDEX CARD



Notice the data is coded in BINARY, where each HOLE is a 1, and the absence of a HOLE is a 0. Just to be confusing the MOST SIGNIFICANT binary digit (bit) is at the top of the card, and the least at the bottom. We can read it better this way, by turning it

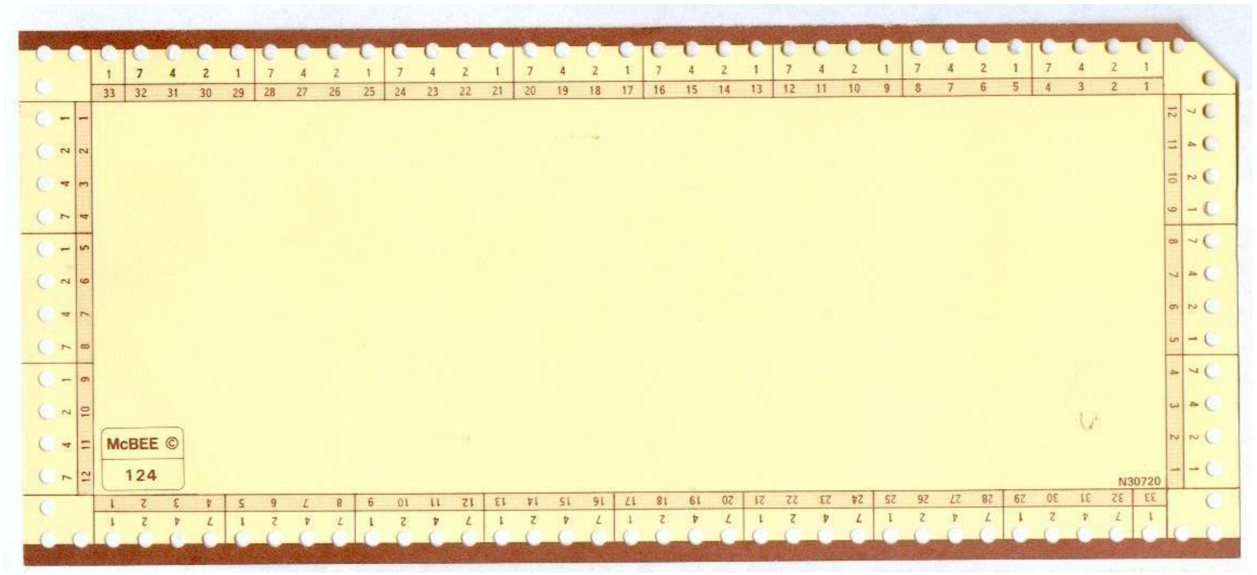


Then it looks more like the BINARY numbers we saw before. If you look very closely reading the top punched row from left to right you see:

010000010010

These cards are what made early computer makers like HOLLERITH, UNIVAC, IBM, HONEYWELL and so on famous, and were first used in the United States Census around 1900. The machines (early computers) to read these cards were very expensive, and used electricity to sense whenever there was a HOLE=1, or NO HOLE=0.

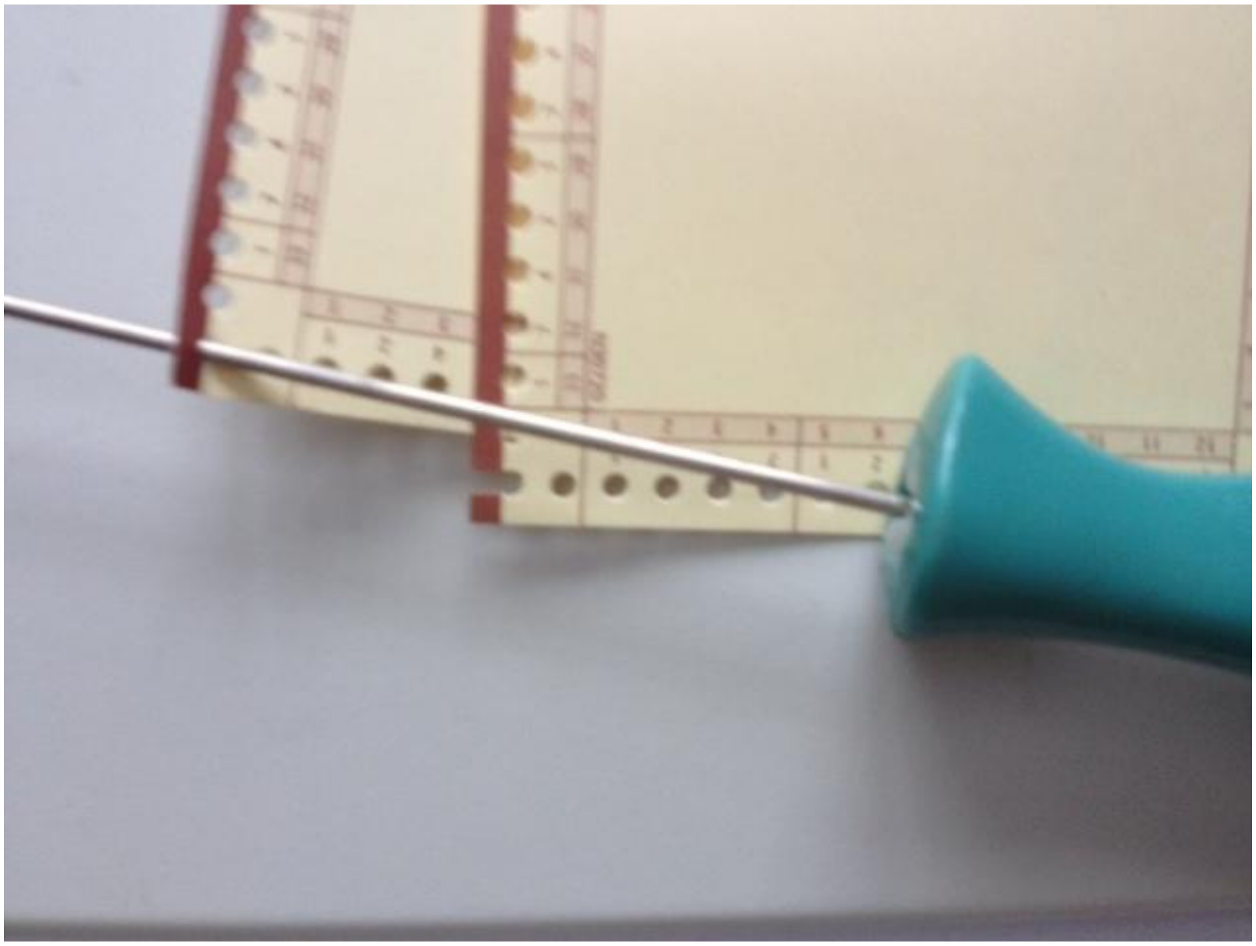
BUT we don't have to be so sophisticated to record and retrieve our information. Here is an example of another widely used INDEX CARD, with a difference



It is called a McBee card (that was the company that made them. Here is how it works:

1. Write (or type, or print on a computer or whatever) whatever you want. In some systems small photographs were even attached. For example, each card could represent a student.
2. Figure out how you want to identify this item. For example the little hole at the very top left might represent EYE COLOUR=BLUE (YES or NO).
3. If the student represented by this card actually has EYE COLOUR= BLUE (YES), then cut from the edge of the card down into the hole so there is a SLOT out to the edge of the card.
4. When we put these cards all together into a DECK (or DATABASE), we can pick out all of the BLLUE-EYED students by inserting a ROD (like this one):

5. The EYE COLOUR=BLUE cards fall out of the DECK, the others remain SKEWERED on the rod.



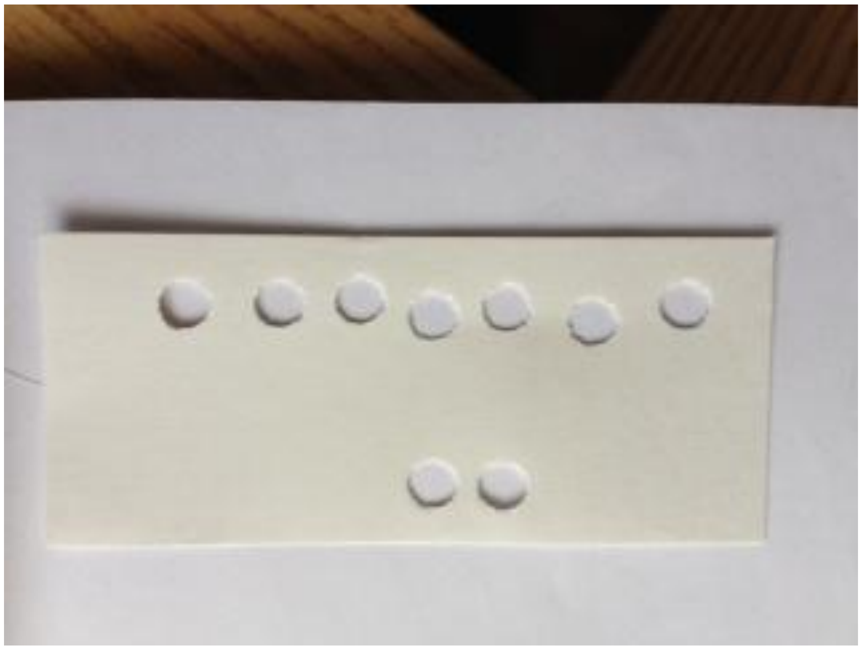
<u>WHAT YOU NEED</u>

1. Index cards. Small ones 9x4 cm. are good, but any work OK. You can use a standard piece of paper cut small rectangles in a pinch. Anything small than 9x4 cm. is problematic.
2. A paper punch. It needs to be a "clean" hole – just pushing a pencil or pen through the card doesn't work very well.
3. Scissors – used to cut from the edge of the card down into the hole to create a "1".
4. One or more SORTING RODS. Bamboo "skewers" for shish-ka-bobs available at grocery stores work. In a pinch you can bend a wire coat hanger or just straighten out the curved end of the hanger without breaking it up, to get a short piece of stiff wire. Use twist-ties in a pinch – but something more rigid/substantial works better.

In this individual or team exercise, we will create some INDEX CARDS that are coded in BINARY, to answer 2 questions about students:

1. What is the eye colour of the student?
2. Which other students in the class does each student like?

Let's start by "designing" our INDEX CARD. We can use any kind of card stock. Paper is not so good because it does not "shake out" from a DECK of records, but it could be used in a pinch.



a. Here is a sample of small index card where the BOTTOM ROW has two BINARY selectors – these could be BROWN EYES=YES/NO and BLUE EYES=YES/NO. And the TOP ROW is encoded for something else (as suggested below).
b. The holes are just punched with a hand paper-punch:

Note: if you use a "modern" punch like this it keeps the punch-out CHADs until they can be dumped in the waste-basket. Otherwise you end up with a lot of CONFETTI.

c. To make it easier to do the sorting, punch out as many cards as you can together (maybe 5-7). Try and get all the HOLES lined-up as closely as you can – a little variation (1/2 a hole-width) can be tolerated.
d. IMPORTANT. The examples as we continue, assume the INDEX CARDs are setup to select EYE COLOUR and who students LIKE (YES/NO). If you want to have more selection items (along another part of the long edges, or the short edges) – YOU NEED TO DECIDE THAT DESIGN NOW, and change things accordingly. The example here continues with EYE COLO|UR and who LIKES who. (Another possibility would be to encode the STREET students live on, their age, who their favourite movie star is etc. etc.). Having this discussion in class turns the thinking to how to do the DESIGN of the information, and how that can BE ENCODED in BINARY selectors. All the Selectors don't have to be YES/NO, several BITS as we saw before could be used to ENCODE a NUMBER which represents something, such as AGE).

- To keep the lesson short, it’s easier to just re-create the cards shown.

- It is important, however from a learning standpoint to actually create the cards and watch them fall out of the DECK. This really reinforces BINARY selection of items.

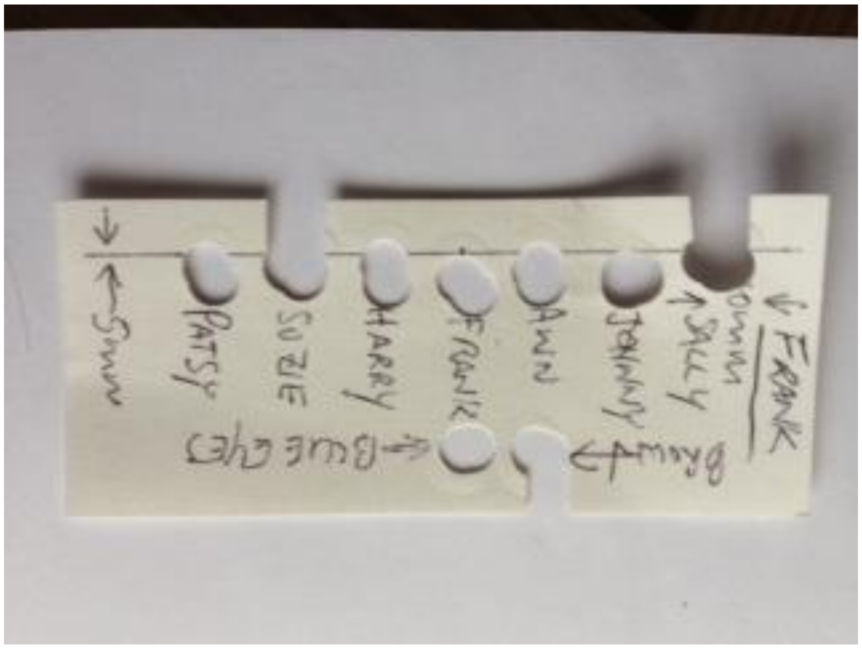
- To have more fun, and get more participation, it’s better to encode the information for the real class.

- If time allows, get class agreement on the DESIGN and BINARY selection times (or items encoded as BINARY NUMBERS using multiple HOLES) is very good.

- You can include other information on the INDEX CARD that is not a BINARY selection criteria. For example, students could include a Cartoon drawing of the student, write something else unique to them, etc.

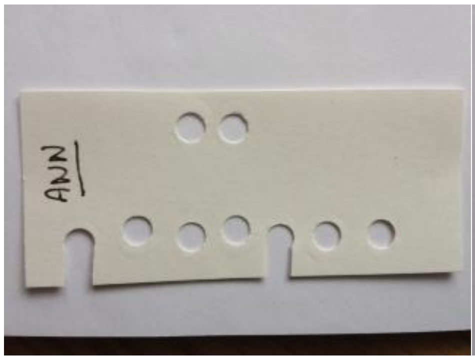
- Don’t aim for an OVERYLY COMPLEX design. The basic principles are demonstrated just fine with a couple of simple BINARY selection criteria like EYE COLOUR and who you LIKE.

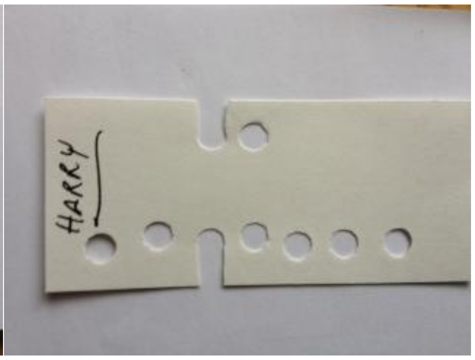
e. The bottom row can be used to ENCODE whether this student LIKES any of the other students represented by these BINARY DIGITS. For example if this card is for FRANK, we can punch out whether he likes PATSY or SUZIE, etc. (one for each HOLE). You need to decide whether FRANK should LIKE FRANK.
f. So, here is FRANK’s INDEX CARD

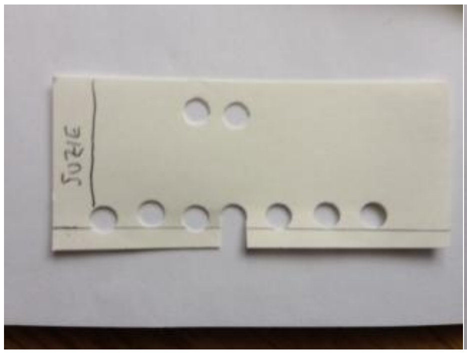


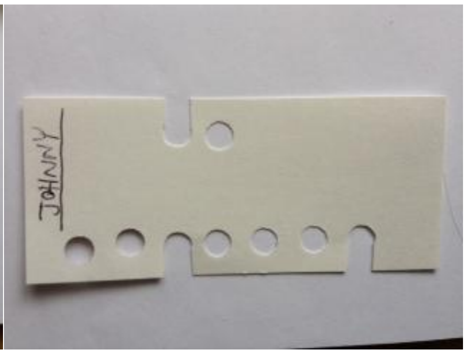
What is his EYE COLOUR? Who does he LIKE?

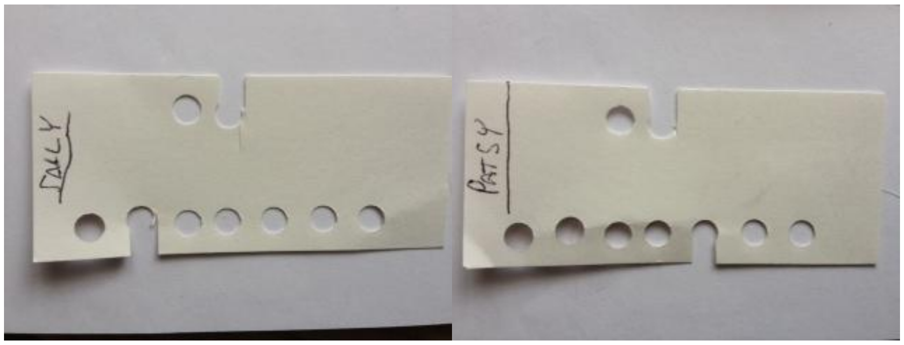
g. We can continue making INDEX CARDs for all of our students, and cutting out the hole to the edge, representing a 1 or TRUE for the various BINARY selection items. This raises some interesting points, such as "Should a person LIKE themselves" (probably not), and "What if a student's EYE COLOUR is not BLUE or BROWN?" (OK, shouldn't be a problem, except all we will ever be able to tell is that their EYE COLO|UR is NOT BLUE AND NOT BROWN.











h. We can also discuss what is means to LIKE Sally OR Patsy, or LIKE Sally AND Patsy, or maybe LIKE Sally but NOT Patsy etc. Does our ENCODING scheme work for these cases? Hopefully it does, and now we begin to see the power of BINARY and BINARY BOOLEAN operations such as NOT, AND, OR. Have the students formulate some QUERIES on our INDEX CARDS using questions using FIRST ORDER BINARY logic like this. Can they see how easy and powerful it is?

So what did we learn?

- It is quite possible to search for information using simple BINARY keys
- We can choose several interesting or important KEYS for an information record
- It is very important to DESIGN what information we will use to LOOK for selected information
- We can combine several BINARY items into a KEY such as number, by just using more BITS
- We can make combined queries using FIRST ORDER LOGIC like: NOT, AND, OR which combines simple BINARY items in order to select items. If time allows we can have some riddles which students can use to understand how LOGIC works. For example "If FRANK always lies, AND Sally never lies, OR FRANK never lies, do we know that FRANK LIKES SALLY"? That is, we need to make sure our data does not violate the rules of logic, or what people are telling you does not follow simple logic.
- Electronic versions of information and storage retrieval use all of the techniques we used in our exercise, just a lot faster and more conveniently

Questions

- Who does Frank LIKE ?
- Who does NOT have BLUE OR BROWN eyes?
- Who LIKES Sally but NOT Patsy?
- How do you select the whole class?
- Can you "SORT" these cards so all the BLUE-eyed people are at the front of the deck?

- Computer systems have a lot of trouble with “Duplicate records” (i.e. 2 cards for the same FRANK). What happens when this INDEX CARD #2 for FRANK is included? How can we prevent this in our experiment? (HINT: what if we SORT the cards somehow – does that help us find duplicates?).
- What other BINARY methods can you use to encode information (HINT: INDEX CARD Colour is one)
- Could we set this up so that cutting out to the edge represented a BINARY 0, and leaving the HOLE alone was a BINARY 1?

Resources/Materials

Alignment to Curriculum

Internet Connections

“Give Binary a Try” lesson plan on Tryengineering.org

“Binary for the Younger Set” Lesson Plan

Trycomputing.org “Fun with Sorting” Lesson

Optional Writing Activities

Student Resource

Binary Basics (Reprinted from TryEngineering.org "Give Binary a Try" lesson)

Binary Bytes and Computer Applications

The binary numeral system (base 2 numerals), or bin for short, represents numeric values using two symbols, typically 0 (off) and 1 (on). Because of its straightforward implementation in electronic circuitry, the binary system is used internally by virtually all modern computers. And, computers can be found in just about every product used in today's society - from cars, to phones, to refrigerators -- and also in most manufacturing processes. In almost all modern computers, each memory cell is set up to store binary numbers in groups of eight bits (called a byte).
Each byte is able to represent 256 different numbers; either from 0 to 255 or -128 to +127. To store larger numbers, several consecutive bytes may be used (typically, two, four or eight).
When negative numbers are required, they are usually stored in two's complement notation. Other arrangements are possible, but are usually not seen outside of specialized applications or historical contexts. A computer may store any kind of information in memory as long as it can be somehow represented in numerical form. Modern computers have billions or even trillions of bytes of memory.

How Does It Work?
One can think about binary by comparing it with our usual numbers. We use a base ten system. This means that the value of each position in a numerical value can be represented by one of ten possible symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. We are all familiar with these and how the decimal system works using these ten symbols. When we begin counting values, we should start with the symbol 0, and proceed to 9 when counting. We call this the "ones," or "units" place.

The "ones" place, with those digits, might be thought of as a multiplication problem. 5 can be thought of as 5 × 100 (10 to the zero power, which equals 5 × 1, since any number to the zero power is one). As we move to the left of the ones place, we increase the power of 10 by one. Thus, to represent 50 in this same manner, it can be thought of as 5 × 101 , or 5 × 10

500 = (5 x 102) + (0 x 101) + (0 x 100)

5834 = (5 x 103) + (8 x 102) + (3 x 101) + (4 x 100)

When we run out of symbols in the decimal numeral system, we "move to the left" one place and use a "1" to represent the "tens" place. Then we reset the symbol in the "ones" place back to the first symbol, zero. Give Binary a Try!

Developed by IEEE as part of TryEngineering : <u>www.tryengineering.org</u>