Activity 4

Card flip magic—*Error detection and correction*

Age group	Middle elementary and up.
Abilities assumed	Requires counting and recognition of (small) odd and even numbers. Children will get more out of it if they have learned binary number representation (see Activity 1, Count the Dots).
Time	About 30 minutes.
Size of group	From individuals to the whole class.

Focus

Odd and even numbers.

Patterns.

Magic tricks.

Summary

When data is transmitted from one computer to another, we usually assume that it gets through correctly. But sometimes things go wrong and the data is changed accidentally. This activity uses a magic trick to show how to detect when data has been corrupted, and to correct it.

Technical terms

Error detecting codes, error correcting codes, parity.

Materials

Each pair of children will need:

a pile of approximately 25 identical cards, as described below.

To demonstrate to larger groups you will need:

a set of about 40 cards with magnets on them, and a metal board for the demonstration.

What to do

This activity is presented in the form of teaching the children a "magic" trick. Their interest is easily gained by first performing the trick, and then offering to show them how to do it. As with any magic trick, a certain amount of drama is helpful in making the presentation effective. The children will need to be sitting where they can see you.

The trick requires a pile of identical, two-sided cards. For example, the cards could be red on one side and white on the other. An easy way to make them is to cut up a large sheet of cardboard that is colored on one side only. A pack of playing cards is also suitable.

For the demonstration it is easiest if you have a set of cards that have magnets on them. It is possible to buy strips of magnetic material with a peel-off sticky back, and these are ideal. Make about forty cards, half with a magnet on the front and half with one on the back. Alternatively, fridge magnets can be used; glue them back to back in pairs, and paint one side. You can then lay the cards out vertically on a metal board (e.g. a whiteboard), which is easy for the class to see. When the children come to do the trick they can lay the cards out on the floor in front of them.

1. Have one or two children lay out the cards for you on the magnetic board, in a rectangular shape. Any size rectangle is suitable, but about five by five cards is good. (The larger the layout, the more impressive the trick.) The children can decide randomly which way up to place each card. Figure 4.1 shows an example of a five by five random layout.

2. Casually add another row and column to the layout "just to make it a bit harder" (Figure 4.2). Of course, these cards are the key to the trick. The strategy is to choose the extra cards to ensure that there is an even number of colored cards in each row and column (this is explained in more detail below).

3. Select a child, and while you cover your eyes and look away, have the child swap a card—just one card—for one of the opposite color (they only need to flip it over if it is on the floor). For example, in Figure 4.3, the third card in the fourth row has been flipped. You

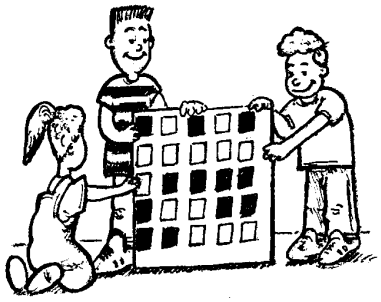

Figure 4.1: An initial random five by five layout of cards

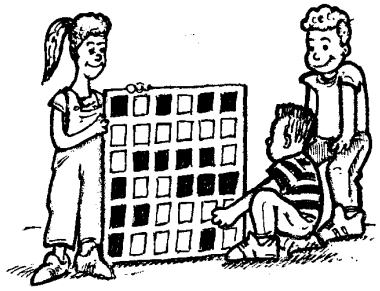

Figure 4.2: The cards with an extra row and column added

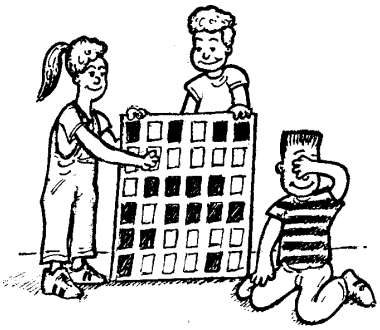

Figure 4.3: One card has been flipped

then uncover your eyes, study the cards, and identify which one was flipped. Because of the way the cards were set up, the row and column containing the changed card will now have an odd number of colored cards, which quickly identifies the offending card. Flip the card back, and have a couple more children choose a card to flip. Children will likely be quite impressed with your ability to repeatedly find the flipped card. The trick works with any number of cards, and is particularly impressive if a lot of cards are used, although it is important to rehearse the trick in this case.

4. Have the children try to guess how the trick is done. This is a good exercise in reasoning, and also helps to establish that the method is not obvious.

5. At this point you offer to teach the trick to the children. We have yet to see this offer refused!

   It is helpful to have the children work in pairs. Get each pair to lay out their own cards in a four by four square, choosing randomly whether each card is showing colored or white.

   Check that children can remember the concept of odd and even numbers, and that they appreciate that zero is an even number. Then get them to add a fifth card to each row and column, making sure that the number of colored cards is even (if it is already even then the extra card should be white, otherwise it should be colored). The technical name for the extra card is a *parity* card, and it can be helpful to teach the children the term at this point.

   Point out what happens if a card is flipped—the row and column of the flipped card will have an odd number of colored cards, and so the flipped card is the one where the offending row and column intersect. Each member of a pair can now take turns at doing the "trick."

   Once they are comfortable with the trick they may wish to get together with another pair and make up a larger rectangle of cards. You might even try pooling all the children's cards together to make one huge square.

Variations and extensions

Instead of cards, you can use just about any object at hand that has two "states." For example, you could use coins (heads or tails), sticks (pointing left-right or up-down), or cups (upside-down or right-way-up). If the activity is to be related to binary representation (Activity 1), the cards could have a zero on one side and a one on the other. This makes it easier to explain the wider significance of the exercise—that the cards represent a message in binary, to which parity bits are added to protect the message from errors.

There are lots of other things that children can discover by experimenting with the cards. Get them to think about what happens if two cards are flipped. In this case it is not possible to determine exactly which two cards were flipped, although it is always possible to tell that something has been changed. Depending on where the cards are in relation to each other, it can be possible to narrow it down to one of two pairs of cards. A similar thing happens with three card flips: you can always tell that the pattern has been changed, but it's not possible to determine exactly which cards have been flipped. With four flips it is possible that all the parity bits will be correct afterwards, and so the error could go undetected.

Another interesting exercise is to consider the lower right-hand card. If you choose it to be the correct one for the column above, then will it be correct for the row to its left? (The answer is yes, which is fortunate as it saves having to remember whether it applies to the row or the column. Children can probably "prove" this to themselves by trying to find a counter-example.)

The description above uses *even parity*—it requires an even number of colored cards. It is also possible to use *odd parity*, where each row and column has an odd number of colored cards. However, the lower right-hand card only works out the same for its row and column if the number of rows and columns of the layout are either both odd or both even. For example, a 5 by 9 layout or a 12 by 4 layout will work out fine, but a 3 by 4 layout won't. Children could be asked to experiment with odd parity and see if they can discover what is happening to the corner bit.

An everyday use of a related kind of error checking occurs in the International Standard Book Number (ISBN) given to published books. This is a ten-digit code, usually found on the back cover of a book, that uniquely identifies it. The final (tenth) digit is not part of the book identification, but is a check digit, like the parity bits in the exercise. It can be used to determine if a mistake has been made in the number. For example, if you order a book using its ISBN and get one of the digits wrong, this can be determined using only the checksum, so that you don't end up waiting for the wrong book.

The checksum is calculated using some straightforward arithmetic. You multiply the first digit by ten, the second by nine, the third by eight, and so on, down to the ninth digit multiplied by two. Each of these values is then added together to form a value which we will call s. For example, the ISBN 0-13-911991-4 gives a value

$$s = (0\times 10)+(1\times 9)+(3\times 8)+(9\times 7)+(1\times 6)+(1\times 5)+(9\times 4)+(9\times 3)+(1\times 2) = 172.$$

You then take the remainder after dividing s by 11, which is 7 for the example. If the remainder is zero then the checksum is zero, otherwise subtract the remainder from 11 to get the checksum. For the example, the checksum (at the end of the ISBN), is therefore $11-7=4$.

Figure 4.4: A bar code (UPC) from a grocery item

If the last digit of the ISBN wasn't a four then we would know that a mistake had been made somewhere.

With this formula for a checksum it is possible to come up with the value 10, which requires more than one digit. This is solved in an ISBN by using the character X for a checksum of 10. It isn't too hard to find a book with an X for the checksum—one in every 11 should have it.

Have the children try to check some real ISBN checksums. Now get them to see if they can detect whether one of the following common errors has been made in a number:

- a digit has its value changed;
- two adjacent digits are swapped with each other;
- a digit is inserted in the number; and
- a digit is removed from the number.

Have the children think about what sort of errors could occur without being detected. For example, if one digit increases and another decreases then the sum might still be the same.

A similar kind of check digit (using a different formula) is used on bar codes (universal product codes or UPCs) such as those found on grocery items (Figure 4.4). If a bar code is misread, the final digit is likely to be different from its calculated value, in which case the scanner beeps and the checkout operator must re-scan the code.

What's it all about?

Imagine that you are depositing $10 cash into your bank account. The teller types in the amount of the deposit, and it is sent to a central computer to be added to your balance. But suppose some interference occurs on the line while the amount is being sent, and the code for $10 is changed to $1,000. No problem if you are the customer, but clearly it is in the bank's interest to make sure that the message gets through correctly!

This is just one example of where it is important to detect errors in transmitted data. Just about anywhere that data is transmitted between computers, it is crucial to make sure that a receiving computer can check that the incoming data has not been corrupted by some sort of electrical interference on the line. The transmission might be the details of a financial transaction, a fax of a document, some electronic mail, or a file of information. And it's not just transmitted data that we are concerned about: another situation where it is important to ensure

that data has not been corrupted is when it is read back from a disk or tape. Data stored on this sort of medium can be changed by exposure to magnetic or electrical radiation, by heat, or by physical damage. In this situation, not only do we want to know if an error has occurred, but if possible we want to be able to reconstruct the original data—unlike the transmission situation, we can't ask for corrupted disk data to be re-sent! Another situation where retransmission is not feasible is when data is received from a deep space probe. It can take many minutes, even hours, for data to come in from a remote probe, and it would be very tedious to wait for retransmission if an error occurred. Often it is not even possible to retransmit the data, since space probes generally don't have enough memory to store images for long periods.

People have come to expect that when they store a document on their word processor, or send a message by electronic mail, they won't find the occasional character changed. Sometimes major errors happen, as when a whole disk gets wiped out, but very minor errors just don't seem to occur. The reason is that computer storage and transmission systems use techniques that ensure that data can be retrieved accurately.

Being able to recognize when the data has been corrupted is called *error detection*. Being able to reconstruct the original data is called *error correction*. A simple way to achieve error detection and correction is to transmit the same data three times. If an error occurs then one copy will be different from the other two, and so we know to discard it . . . or do we? What if, by chance, two of the copies happen to be corrupted in the same way? What's more, the system is very inefficient because we have to store or send three times as much data as before. All error control systems require some sort of additional data to be added to our original data, but we can do better than the crude system just described.

All computer data is stored and transmitted as sequences of zeros and ones, called *bits*, so the crux of the problem is to make sure that we can deliver these sequences of bits reliably. The "card flip" game uses the two sides of a card to represent a zero or one bit respectively, and adds parity bits to detect errors. The same technique is used on computers. Adding a single parity bit to a row of bits will allow us to detect whether a single bit has been changed. By putting the bits into imaginary rows and columns (as in the game), and adding parity bits to each row and column to ensure that it has an even number of ones, we can not only detect if an error has occurred, but *where* it has occurred. The offending bit is changed back, and so we have performed error correction.

In practice computers often use more complex error control systems that are able to detect and correct multiple errors. Nevertheless, they are closely related to the parity scheme. Even with the best error control systems there will always be a tiny chance that an error goes undetected, but it can be made so remote that it is less likely than the chance of a monkey hitting random keys on a typewriter producing the complete works of Shakespeare.

And to finish, a joke that is better appreciated after doing this activity:

Q: What do you call this: "Pieces of nine, pieces of nine"?

A: A parroty error.

Further reading

The parity method of error detection is relatively crude, and there are many other methods around that have better properties. Some of the more important ideas in error coding are *Hamming distances, CRC (cyclic redundancy check), BCH (Bose-Chaudhuri-Hocquenghem) codes, Reed-Solomon Codes,* and convolutional codes. Details about these sorts of codes can be found in Richard Hamming's book *Coding and Information Theory*, and Benjamin Arazi's *A commonsense approach to the theory of error correcting codes.* The ISBN of Hamming's book is 0-13-139139-9, an interesting number for a book about coding. Less comprehensive, but more accessible, information about error correction can be found in *The Turing Omnibus* by Dewdney, and in *Computer Science: An Overview* by Brookshear.