

INTERLUDE: PLAY

"It's got to stop sometime"

Scene: A classroom with a long, wide blackboard at the front. The professor is standing at the front, asking for volunteers.

Prof: Come on now, I need five volunteers to be "people programs". All you need to do is to hold up one of these cards and, when I say, you just perform the instructions on the card to whatever is written on the board.

Noel: I'll have a go but I'm not very good at this sort of thing. I'm sure I'll get it all back-to-front.

Prof: That's exactly what I want you to do. Your program is called **REVERSE**.

He hands Noel a card on which is written the words:

Reverse the data

Now whenever I call on you all you have to do is to rewrite whatever is on the board backwards.

Peter: If it's as easy as that then I'm your man and as my mum always says if you want someone to do a job properly and not give up half-way through then ask me because I'm your man and as my mum always says ...

Prof: I'm sure you are, Peter.

He hands him a second card bearing the instruction:

WHILE-EVER THE DATA ENDS IN "T" PUT ANOTHER AT THE END OF IT

The Bubble Twins: (*in chorus*) We'd like to help too, but only if we can do it together.

Prof: Oh, then you'll like your job.

He gives June Bubble a card on which is written:

COPY THE DATA

When I call on you, all you have to do is to make a second copy of whatever appears on the blackboard.

Jane: (*to her sister*) Ooh, I'll do the copying because I've got the steadier hand. You can hold up the instructions in case I forget them.

Prof: Right, let's practise those three programs.

Mary: What about me? I knew you'd forget met. It's just not fair!

Prof: You'll get your chance, Miss Contrary, I've got just the job for you. But we'll just practice these first three.

Now when I call out the name of your program you have to perform whatever instructions you have on whatever appears on the board. If I say **REVERSE** that's your cue, Leon.

Noel: Do you mean me?

Prof: Sorry, Noel, yes it's you I mean. And if I say **REPEAT** its over to you Peter. And your cue girls is **DOUBLE**.

*He writes the letters RAH on the board. OK it's **DOUBLE** first.*

*The Bubble sisters write a second **RAH** underneath the first.*

Now **REVERSE**.

*Noel rubs out the message **RAH RAH** and replaces it with **HAR HAR**.*

And **DOUBLE** again.

The message now becomes **HAR HAR HAR HAR**.

And finally **REPEAT**.

*Peter was about to start tacking a row of **T**'s on the end of the data but the Prof caught him just in time.*

No Pete. Your instructions are to add **T**'s while-ever it already ends in **T**. But it doesn't, so there's nothing for you to do.

Peter, somewhat disappointed, sits down again.

Now we'll try another one.

He cleans the board and replaces it by **EXIT**.

REVERSE

Noel changes **EXIT** into **TIXE**.

Peter: Isn't **ENTRANCE** the reverse of **EXIT**?

Prof: No Pete, Noel's right. I said **REVERSE**, not **OPPOSITE**. OK, now **DOUBLE**.

*Jane Bubble adds a second **TIXE**.*

REVERSE

*Noel replaces the **TIXE TIXE** with **EXIT EXIT**.*

And now **REPEAT**.

*Peter excitedly writes **T** after **T**, getting **EXIT EXITTTTTTTTTTTTTTTTTT.....** until he'd run out of blackboard. The Prof had to restrain him from continuing across the wall.*

Mary: That's stupid! Whenever Pete takes off nobody else can follow him.

Prof: No, Mary, its not stupid. It's just like when a computer program crashes because it gets into a loop.

Mary: Well it's stupid ever to get into a loop. The computer should be clever enough to know that it's being told to get into a never-ending loop and spit out the offending program.

Prof: But Mary, it's not always so easy to ensure that a program will go on forever.

Mary: 'Course it is! Any fool could see what was going to happen when Pete took over. A clever computer would be able to examine any programs it had to run and refuse any which would make it crash.

Prof: But that would need another program to work out what would happen.

Mary: So what! It might be a complicated program but I'm sure someone smart like Tim could come up with one. You just get Tim's program to look at the one you're going to run and if it's OK it rings a bell and if it would loop forever it rings a buzzer. Then you'd know not to let the computer run any program that sets off the buzzer.

Prof: But this program would have to be able to work on every possible program.

Mary: Sure, and what's wrong with that?

Prof: Well, it would even have to be able to work on itself.

Mary: Well any dum dum can see that Tim's program would always halt so if you ran it on itself you'd get the bell, of course. Now when are you going to give me my program, or had you forgotten?

Prof: OK Mary Contrary, I've got just the program for you. It's called **DISOBEY**.

He gives her a card with the following instructions:

**If data is HALT then REPEAT
else REVERSE**

Mary: But that's silly. If I'm told to **HALT** I go on forever writing **HALTTTTTTT.....** and if, for example, I'm told **LOOP**, I write **POOL** and then halt. I'll always be doing the opposite to what I'm told.

Prof: That's why it's called **DISOBEY**, Miss Contrary! Let's try it out.

*He writes **POTS** on the board. Now **REVERSE**.*

*Noel changes it to **STOP**.*

And now **DISOBEY**.

Mary: Well the data is not **HALT** so I do the else bit. That means getting **POTS** again. She picks up the duster but the professor gently restrains her. What's the matter, I've got to do a **REVERSE**, don't I?

Prof: Not you, your job is to activate Leon as a subroutine. He does the actual reversing.

Mary: Oh, all right then. Go on Noel. (I suppose that's who you meant.)

*Noel reverses **LOOP** and once again the word **POOL** is written on the board.*

Prof: Now again. He cleans the board and writes **HALT**. OK Mary **DISOBEY**.

*Mary gives Peter a hard thump and Peter starts writing dozens of **T**'s until the Professor gives Peter a nudge to break him out of his infinite loop.*

Now has it ever occurred to you that a program can be made to operate on itself?

Tim: Well I suppose I could write a program called COUNT which counts the number of words in a piece of text and I could run it on a copy of the COUNT program itself.

Prof: Exactly. So June, if DOUBLE acted upon itself, what would happen?

June: DOUBLE DOUBLE I suppose.

Prof: And, Leon if you REVERSE REVERSE?

Noel: You'd get ESREVER.

Prof: Pete, would you mind doing REPEAT REPEAT.

Peter: What do you mean?

Prof: I mean write REPEAT on the board as your data and carry out the REPEAT program on it.

Peter writes REPEAT on the board and then, after scratching his head for a minute, he turns it into REPEATTTTTTTTTTTTTTTTTTTTT.....

Prof: So if DOUBLE acts upon itself it will halt. The same is true of REVERSE. But if REPEAT acts on its own description as data it will never halt.

Jane: It's just like it gets indigestion. It can't digest a copy of itself.

Mary: Sounds like a cannibal. What a positively disgusting idea!

Prof: That's a good analogy. How about if we call a program a "cannibal" if it halts when it feeds on itself. So DOUBLE and REVERSE are cannibals. But REPEAT is not. As Jane says, it gets indigestion if it tries to eat a copy of itself. What about DISOBEY Mary?

Mary: DISOBEY isn't HALT so once again I do the "else". Go on Noel, REVERSE. And Noel proceeds to turn DISOBEY into YEBOSID.

Prof: So DISOBEY is a cannibal program.
Now Tim, the last program is yours. It's called PREDICT.

Tim: I knew you'd say something like that. You're going to tell me that my program predicts whether or not any program will halt, or whether it will go into an infinite loop.

Prof: Exactly, and because the answer will depend on what data it's given it needs to be given the program plus the data.

He hands Tim the last card with the program:

PREDICT

**If the program will halt when given the data, print out HALT
but if the program will loop, print out LOOP**

Noel: That's not very difficult. All Tim's program has to do is just run the given program and if it halts then it prints out HALT and if it doesn't halt ...

Prof: ... then you'd never be able to break into it to print out the message **LOOP**.

Peter: Well can't you just break it out of its loop if it seems to be going on a long time.

Prof: How long? A program might take a very long time and still halt. Even if you waited a hundred years you wouldn't know for certain that it's not going to halt some time in the future.

Noel: Well how's Tim going to do it?

Prof: He can't. It's impossible.

Mary: That's rubbish. Tim's a computer whiz. And even if Tim can't, someone will one day. It makes me mad when people say that something is impossible just because they're not clever enough to do it themselves!

Prof: Well, we're supposing for the sake of argument that Tim has done it and **PREDICT** is that program. Let's try it out.

He writes the word **TEST** followed by the word **DOUBLE**.
OK Tim, **PREDICT**.

Tim: Well it's obvious that if you ran the program **DOUBLE** on the **TEST** data you're just going to get **TEST TEST**.

Prof: So, carry out your program.

Tim: If I ran **DOUBLE** on **TEST** the program would halt so I write the word **HALT**.
*He erases **TEST DOUBLE** and replaces it by **HALT**. The Prof now writes the word **REPEAT** to the right of **HALT**.*

Right Tim, here's another example, go ahead and **PREDICT**.

Tim: Clearly I predict that **REPEAT** will loop in this case.
*He writes the word **LOOP** in place of **HALT REPEAT**.*

Prof: Well Tim, is **PREDICT** a cannibal, will it halt if it feeds upon its own description?

Tim: I guess so. It is supposed to print either **HALT** or **LOOP**, but in either case it, itself, has to halt so that you can read its answer.

Prof: Now if I was to attach **DOUBLE** to **PREDICT** you'd get a program which tells you whether or not any given program is a cannibal. But I want to give it a twist. Here is a program I've called **MONSTER**.

The professor holds up the last card displaying the four words:

MONSTER

**DOUBLE
PREDICT
DISOBEY**

Do you think **MONSTER** is a cannibal?

Peter: Well it sounds like a pretty uncivilised, pagan program so I guess it is.

Prof: Guessing isn't good enough. We must have certainty.

Jane: Well, one thing's for certain, either it is a cannibal or it isn't.

Mary: Stupid girl. Where do you think that inane remark will get us?

Prof: Further than you might think. Let's follow up each possibility in turn. Suppose Pete is right and it is a cannibal. Let's feed **MONSTER** its own description to digest. What happens first?

June: Well first we do **DOUBLE** and get **MONSTER MONSTER**.

Tim: Then **PREDICT** examines the structure of **MONSTER** and decides whether it will halt when it feeds on **MONSTER**.

Noel: And because we are at the moment assuming that it is a cannibal it will be able to digest its own description, so **PREDICT** will spit out **HALT**.

Mary: Then I come along and upset the applecart, because as soon as I see the word **HALT**, my instructions in **DISOBEY** tell me to turn this into **HALTTTTTTTTTTTTTTTTT...**

Peter: But that will give **MONSTER** indigestion. It'll never get to the end.

Mary: So **MONSTER** is not a cannibal after all. That's dumb. We assumed it was.

Prof: So all that means is that that assumption has to be rejected.

Tim: Oh, I see, that contradiction proves that **MONSTER** is not a cannibal.

Prof: Well, as that seems to be the only possibility remaining, let us assume that **MONSTER** will go on for ever if it feeds on a copy of itself.

Mary: We don't need to assume that, we know that.

Prof: So let's follow through **MONSTER** again as it attempts to digest **MONSTER**. First step gets us **MONSTER MONSTER**.

Tim: Then my **PREDICT** program interprets this as the program **MONSTER** acting on the data **MONSTER** and **PREDICT** must predict whether it will halt.

Noel: And since we know that **MONSTER** is not a cannibal, the answer **LOOP** will come out of the **PREDICT** part of **MONSTER**.

Mary: And then I come along and **DISOBEY**, which means that since I don't see the word **HALT** I simply turn the **LOOP** into a **POOL** and halt. But that's dumb too because that means that **MONSTER** is a cannibal. It fed upon itself and finished. Didn't you say that **MONSTER** couldn't be a cannibal?

Prof: Well we do appear to be in a bit of a fix. If we suppose that **MONSTER** is a cannibal we can prove he isn't and if he isn't we can prove he is.

Mary: That's the dumbest thing I ever heard. If he is, he isn't and if he isn't he is.

Prof: So we've reached a blank wall again. But remember, we're still making an assumption.

Noel: What's that?

Prof: Well Tim hasn't actually got a **PREDICT** program.

Peter: So ... ?

Prof: If ever he, or anyone else for that matter, ever came up with a **PREDICT** program that can decide in advance whether or not any given program will halt, the contradiction we reached must inevitably follow. So no such program could ever be written. The "Halting Problem" is insoluble!

Mary: My "Halting Problem" is the fact that this stupid lesson seems to be going on forever. Tim, do you predict it will ever HALT?

At that moment the end-of-lesson bell was heard.

Tim: Indeed I do.