

CS4FN

Computer Science for Fun

Summer 2005

*'Clever' lumps
of silicon*

*The Compute-Ability
Competition*

*Computer Science
at the movies*

Sodarace Special

Create artificial life forms



Welcome to the first issue of *cs4fn*: The magazine about the fun side of computer science.

We are passionate about all things to do with computer science – and it is about a lot more than simply computers. Our aim is to fill *cs4fn* with lots of stuff that you will find both fun and interesting. Hopefully along the way our passion will rub off on you. If you enjoy this issue then check out our website, which has lots more stories, puzzles and links to other interesting places – with more to come. We would love to know what you think of *cs4fn*. If you have any ideas or comments about other things that we could do, please email us at: cs4fn@dcs.qmul.ac.uk.

What is computer science about? We think it's about fun, more than anything...and lots of computer scientists would be doing it for

a hobby if they weren't being paid for it!

Come to that, an article in *Popular Electronics* magazine inspired Bill Gates and Paul Allen's first product: Microsoft started out as a hobby in their garage.

This issue features Sodarace – an online 'Olympics', where you can compete against creatures created by Artificial Intelligence (AI). It has a massive cult following – take care, it is highly addictive! We can't be held responsible for the fact that you won't get much sleep once you start creating Sodarace creatures or Sodaplay art.

Linus Torvalds was a student in Finland, when he posted the newsgroup message: 'Hello everybody out there ... I'm doing a (free) operating system (just a hobby, won't be big and professional ...).' Linux, the operating system he created, is part of a \$14 billion industry that Microsoft now view as their number one threat.

In this issue:

Sodarace:

An in-depth look at how to create and race your own artificial creature

6

'Clever' lumps of silicon:

How do computers become clever enough to beat humans at games like chess?

11

Codebreaking:

How a broken code cost Mary Queen of Scots her life

12

Computer Science at the movies:

Has *Minority Report* seen the future?

15

Are you a codebreaker?

Try one of our puzzles and find out!

18

BrainAcademy:

The Compute-Ability Competition

19

Why computer science?

'You can do things that can't be done in reality – anything is possible.'

Ossama

'It was my hobby and I was heavily into gaming.'

Lukasz

'Programmers work whatever you want them to be.'

Tim

'It's fun writing software that you then see other people using.'

Andrew

'It's just fun and [you can use your] imagination – you can make things that never existed before.'

Nick

'...to grow intellectually and to have loads of fun at the same time.'

Naresh

'...the diversity. It doesn't close any doors.'

Hayley

'When leaving college I was sure of one thing, I loved all things computer science.'

Adam



Enter the maze...

Most websites are designed to be easy to navigate – after all you want people to be able to find things. But what if you really want people to explore your website by wandering from page to page, finding interesting things that they weren't really looking for? What happens if you ignore the normal rules about making things easy to find?

This is what the *cs4fn* website does. Computer systems are often designed around metaphors – like the Microsoft Office desktop on your PC. It's not really the top of a desk but by designing it to look a bit like one, it makes it easier for people to use, as they can guess what they might be able to do, based on what they do on a real desktop. In a real office you move 'documents' around, open them, put

them in 'files' or the 'recycling bin'; the metaphor suggests that you can do the same on your PC. For the *cs4fn* website, we wanted people to 'explore' it, so we have used the metaphor of a 'maze of rooms'. Just like a real maze, you wander from place to place, down dead-ends, passing through rooms with interesting things in, where you might stop for a while, never knowing quite where you are or what is coming up next, but having fun just trying to find the centre of the maze.

Explore the *cs4fn* website (www.dcs.qmul.ac.uk/cs4fn/) through the maze...

'The maze is a great idea' – Adam, Computer Science undergraduate



Is that my equation ringing or yours?

'That ringtone sounds like Dr Who'

Mathematical equations make some people shudder just to look at, whereas mathematicians claim to see beauty in equations. Most people agree music can be beautiful (though you may argue between Mylo and Miles Davis, Beyoncé and Beethoven or Dr Dre versus Debussy).

Beauty, maths and music are closely related. People think maths is just about numbers but it is really about patterns. An equation is just a precise way of describing a pattern – and computer science is partly about actually using those patterns to do something useful. Music is made from patterns of sound – different kinds of patterns lead to different musical styles.

If maths and music are both about patterns, we wondered if it would be possible to listen to mathematical formulae.

Does that make you wonder what an equation sounds like? It made us think and, being computer scientists, we also wondered what we could do with it... so not only have we made it possible for you to hear what a piece of maths actually sounds like, now you can download mathematical ringtones for your mobile – for free – from the *cs4fn* website www.dcs.qmul.ac.uk/cs4fn/





Brain the size of a planet

The War of the Worlds? Could it happen? Are there Extraterrestrials out there, watching and waiting, planning to invade Earth? Your computer could help the world find out.

When you go to bed at night your computer doesn't need to sleep, and neither do hundreds of millions of other computers all over the world. All that computing power going to waste while you snooze, seems a shame. And that's where a new technology for the 21st century comes into play.

It's called GRID computing and it's heading to be the next big thing, now that the World Wide Web is such a part of our everyday lives.

You might have already played with some of the ideas behind grid computing, if, for example, you've been running the SETI (Search for Extra Terrestrial Intelligence) screen saver (<http://setiathome.ssl.berkeley.edu/>). The SETI group is based at Berkley, in the USA. They are trying to find signs of

alien broadcasts amongst the stars. The problem is that there are a lot of stars and heaps of data from radio telescopes to work through to find those elusive signals. The solution? A screen-saver that pops up when you pop off to have a cup of tea, and uses your computer's unused computer power to search some of the radio telescope data. Like a normal screen-saver, it stops when you come back refreshed but perhaps that cuppa could go down in history when **your** computer finally finds ET.

So that's the idea behind GRID computing: all over the world, the Internet harnesses spare computer power to do big, difficult data processing jobs. GRID software allows you to turn computers in different parts of the world into 'computer clusters', where the software decides on the best way to distribute and schedule the work over the member computers, based on the clusters' power and availability. There are even plans to be able to charge companies for this service – you could let your computer join the GRID cluster and the company whose

data you were searching would pay you for the 'work' done by your PC.

There are still lots of interesting issues left to solve, such as writing even faster, better GRID software, as well as dealing with security problems. Suppose a bank wants to process lots of sensitive financial data on a GRID cluster, which means that part of that data will be pumped into your PC. What's to stop less scrupulous folk taking a peek at the data?

As with all great software projects, there are tough technical problems to be overcome. But add to this further problems associated with different laws in different countries about what kind of data can be processed and transferred, combined with a need to make it all secure from criminals, and you can see that the development of GRID software will prove to be an interesting worldwide challenge. It will keep computer scientists busy for years. Perhaps in the future you will be the one who helps to make the GRID happen.

Past, present, future

Past

The first instance of 'spam' (unsolicited email) is believed to have been an announcement of a product presentation sent on 3 May 1978 by a Digital Equipment Corporation salesman to several hundred scientists and researchers on the ARPAnet – the original version of the Internet.

Present

80 per cent of text messages currently received in Japan are unsolicited junk texts ('spim').

Future

Watch out for 'spouch' or 'touch spam'. 'Haptic' interfaces – Interacting with computers by touch rather than sight – is a hot area of research right now. Mobile phones that vibrate silently in your pocket are a very simple version of a haptic interface. Sending sensations over the Internet is an obvious thing to do with it – sending people digital hugs rather than texts – and the spammers won't take long to catch on.

The future may also hold lots of 'spit' in store (spam on Internet telephones). It won't be long before Internet phones are commonplace: phones that use Wi-Fi networks to allow people to talk over the Internet rather than by using the expensive telephone networks. At that point the spammers are likely to start sending junk voicemails...

When do you think the first database was built?

5 years ago?

75 years ago?

50?

In actual fact, one possible contender for the first database is a book created by Saint Isidore of Seville. His 20-volume book *Etymologiae* aimed to be an encyclopedia of all knowledge 1,400 years ago, covering subjects like grammar, geometry, law, military history, agriculture, public games and furniture.

Etymologiae was structured in a way very similar to a modern database, hence the claim to be the creator of the first database. He drew his information from a vast number of sources, and accepted all the 'facts' collected unquestioningly. *Etymologiae* was very much like the web in that readers have to make their own judgements – he included both reliable and unreliable information for his readers to choose from, as a search engine might for you.

Computers follow rules – so do our brains

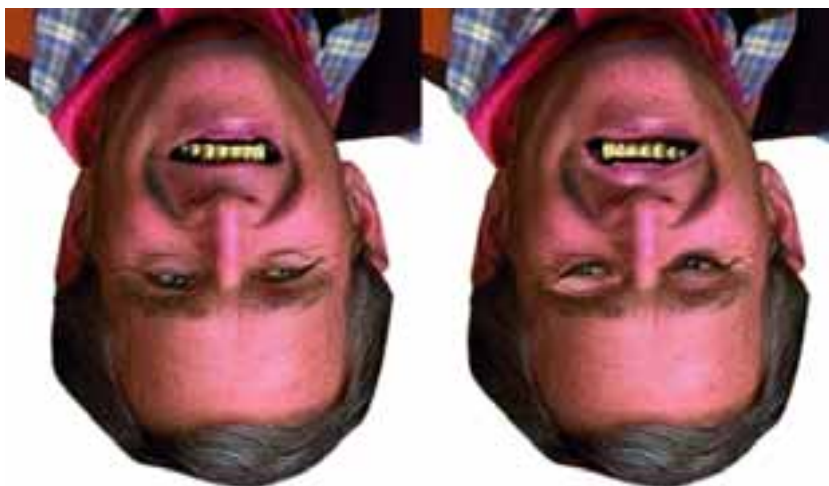
Like a computer, our brain uses 'rules' to help us understand the world. Often we don't know that our brain uses these rules, and it's only when the rules break down that we notice this. This is where optical illusions come in.

In an optical illusion, the human brain makes a mistake in understanding the world. In the example here, you can see two normal upside-down faces. Not such a strange situation. Your brain has rules for recognising faces, which are very useful. But because it's not used to seeing faces upside down, the rules your mind uses to 'put a face together' are not as easy to apply. So, if you turn the pictures upside down, you suddenly see that your brain

was confused in checking the rules. You'd never have made that mistake if the faces were upright!

That means that if you want to become a computer scientist, it helps if you understand the way that people's brains work, so that you can write program rules that let the computer cope with people. One ongoing challenge is to write rules that make computers not only behave as cleverly as people when involved in solitary activities like chess but that can also behave like good team players – working to our strengths and avoiding the weaknesses that result from our brains applying the wrong rules.

Computer science is about people too!



Do you think that the pictures have just turned George Bush upside down? Turn the image around to see!

The Matrix Reloaded – sorry to bug you

In *The Matrix Reloaded* (2003) Neo, Morpheus, Trinity, and crew continue their battle with the machines that have enslaved the human race in the virtual reality of the Matrix.

To find the Oracle, who can explain what's going on (which, given the twisty plot in the *Matrix* films, is always a good idea), Trinity needs to break into a power station and switch off some power nodes so the others can enter the secret floor. The computer terminal displays that she is disabling 27 power nodes, numbers 21 to 48, but that's actually 28 nodes: a computer that can't count and shows the wrong message.

Sadly there are far too many programs with mistakes in them. These mistakes are known as bugs because back in 1945 Lieutenant Grace Hopper, one of the women pioneers of computer science, found an error caused by a moth trapped between the points at Relay 70, Panel F, of the Mark II Aiken Relay Calculator being tested at Harvard University. She removed the

moth, and attached it to her test logbook, writing 'First actual case of bug being found', and so started the term 'debugging a computer program'. As the Oracle would no doubt say 'Check for moths Trinity, check for moths'.



Sodarace: Racing artificial creatures



Humans vs. machine intelligence has been the stuff of many a good Hollywood movie. Sodarace gives you the chance to play with the ideas for yourself, along with thousands of others. Will you be able to create a creature that outruns those designed by others on the net? And if you can, how about beating those created by machines using Artificial Intelligence (AI)?

Sodarace is a joint effort between Queen Mary's Department of Computer Science and the London-based digital arts company Soda Creative Ltd. It allows people worldwide to pit their wits against machine intelligence in an online 'Olympics'. Humans, and artificial intelligence

computer programs, use the free BAFTA-winning online Sodaplay constructor kit (available at www.sodarace.net) to create lifelike virtual racers out of masses and springs, then race them over 2D terrains.

Using the tutorial websites and forum pages you can join the worldwide Sodarace community and learn how to build racers yourself, or try your hand at writing artificial intelligence programs to beat your friends.

Enjoy a day at the Sodaraces and play with the limitless possibilities in creative art and science that Sodarace allows. To start creating your own racers, follow the instructions on www.sodarace.net or follow the 'Humans and AIs Compete' links from the *cs4fn* website (www.dcs.qmul.ac.uk/cs4fn/).

It walks!

'The first few Sodacreatures you make will probably collapse in a heap or wiggle and jiggle themselves exactly nowhere. Mine did, at any rate. Making my first creature, which actually managed to lurch its way from one side of the screen to the other, was a real buzz. It walks! It walks!

That is when the fun really starts – when you start to experiment and gradually discover there are actually lots of ways you can make your creatures move – just like real-life creatures, or even babies: some crawl, some bum-shuffle, some roll their way around before they learn to walk – just like your first Sodacreatures. To get you off to a flying start (well maybe a lurching one), see the step-by-step instructions on how to make one of the simplest creatures (ones that can move from one side of the screen to the other) on the *cs4fn* website: (<http://www.dcs.qmul.ac.uk/cs4fn/alife/sodaindex.html>)'

– Paul



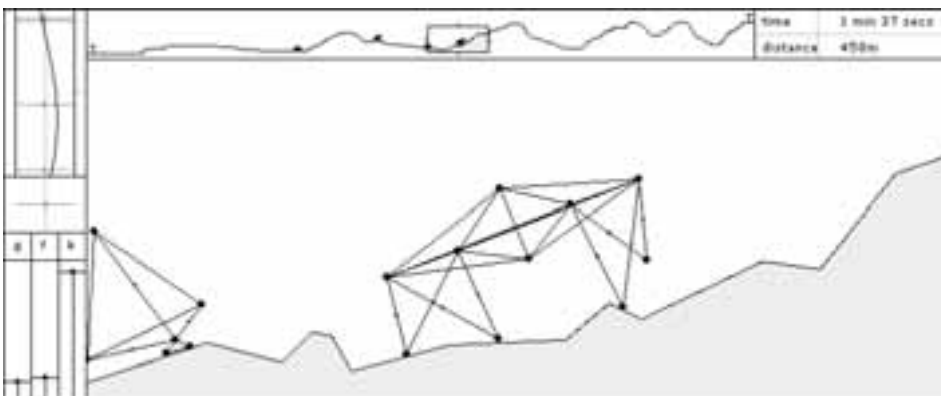
'It's life Jim, but not as we know it!'

Pop down to your chemist and buy some oxygen, carbon, hydrogen, nitrogen, calcium, and phosphorus. OK, now put it all together to make a human being. What's the problem? All the necessary chemicals you need are there, it's just a question of knowing how you put them together.

Life as we know it comes from the right combinations of chemicals, but what do we mean by life? It may seem odd that computer scientists ask this question; surely it's a question for biologists or philosophers? Well, computer scientists can be both, and in the fascinating field of artificial life (A-life) they explore these very questions.

Artificial life asks questions about how 'life as we know it' and – possibly more interestingly – 'life as it could be' come about. Through writing computer programs or building robots to simulate (to try and duplicate or model) real living biological systems we can start to try and understand how our biology works. We can also model the process of evolution in computers, so that we can build our own alien worlds and populate them with our own creatures, and see how they change and adapt to survive. We could, for example, throw in asteroid strikes to change the climate and see what happens next.

Some A-life computer scientists work in the field of xenobiology, trying to work out how aliens on distant planets might look, or use some of the A-life techniques to build better robots to help us work and play. We can even put these life-like qualities into software 'agents', small programs that wander around the web collecting data – that's where the idea for Agent Smith came from in *The Matrix*.



A-life helps us to get a better understanding of living processes. Some scientists, who believe in so-called 'strong A-life', even argue that they are not just simulating life in a computer, but actually creating it – and that life doesn't just belong to us carbon-based life forms. Lots of deep philosophical questions to argue about there!

We also use A-life in movie special effects, computer games and TV. It's used when thousands of Orcs are animated, or to show passengers wandering on the deck of the Titanic. It is used to clone warriors in battle, and, in the current series of Doctor Who, to show mechanical spider creatures crawling over Platform 1, chasing the Doctor. The computer animator doesn't tell each creature what to do; they interact and behave by themselves using A-life techniques, giving us very life-like performances (in some cases better than the human actors!).



With all of this going for it, A-life (not surprisingly) is always a popular topic for student projects. The picture shown is of one creature evolved on a simulated world, where a whole range of different creature shapes occurred, depending on climate and the environment. The student who created this creature didn't claim that it was 'alive', but this is a good example of 'life as it could be' in action.

So the next time you watch The Matrix or see hordes of computer-generated monsters rampaging on the screen, spare a thought for the computer scientists behind the scenes. A-life is their life.

Film facts

The first film shot entirely on digital cameras was 'Star Wars: Attack Of The Clones', directed by George Lucas in 2001.



NASA TETwalkers

It's quite easy to get a triangle to lurch along using Sodaconstructor, and many other Sodacreatures are based around flexing triangles. If you think Sodaracing is just for fun though, think again. NASA is working on the concept of TETwalkers: (http://www.space.com/imageoftheday/image_of_day_050404.html), robots that have a lot of similarities to a 3D Sodatriangle. The trouble with walking your robot on Mars, is that if it falls over there is no one there to pick it up. TETwalkers get round that – they move around by falling over.

How should robots walk? Walking upright on two legs is not as easy as it seems to you. Watch a baby try! The animal world has come up with many other ways of doing it, and science fiction films are catching up. Early film robots tended to look like humans, though now they are

just as likely to scuttle as in Minority Report or the new Doctor Who.

Moving by falling over is a fairly novel way to do it, though. TETwalkers are pyramid shaped frames. Just like in Sodarace, the edges can flex. If you flex the TETwalker edges in the right way, the centre-of-gravity of the top of the pyramid makes the whole thing fall over – the walker is now a bit further along and ready to move again, with a different point at its apex. TETwalkers have already been tested in Antarctica.

NASA is planning to create swarms of TETwalkers that are connected together and move like a snake or an amoeba – and that really does sound like Sodarace. Maybe you can out-invent NASA and build a Sodacreature that could be the basis of a Mars Explorer?

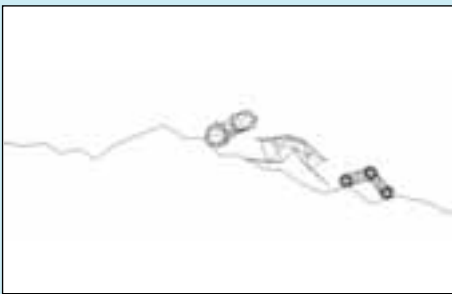


Sodarace – Your starter kit for ten



1 Look at the Sodarace website

Go to www.sodarace.net and have a look around. In particular, look at the 'How do I start?' sections on the homepage, and also the links section which includes several useful tutorial websites. Sodarace provides you with a set of powerful tools to explore your creativity, and also to compete with AI-designed racers. There are also loads of ways to use Sodarace in school: there are even lesson plans in the forums, so get your teachers interested and you could be Sodaracing in lessons!



2 Explore the Sodaconstructor website:

To build Sodarace creatures you use Sodaconstructor, so you need to learn how to use this online tool set. It's really quite simple, but it will take time to become a skilled Sodaconstructor. The cs4fn site has step-by-step instructions with screen dumps to get you started on building a very simple creature that walks. Have a look at the Sodaplay 'How do I start?' section too. It can be found at:
www.sodaplay.com/constructor/

3 The building blocks

All the structures built in Sodaconstructor are made of three basic components: masses, springs and muscles. In Sodarace:

- **Masses** are single points that have weight and obey the laws of gravity.
- **Springs** connect the masses.
- **Muscles** do the moving.

You have two 'modes':

- the '**construct**' mode where you can draw your models, and
- the '**simulate**' mode where the muscles do their stuff and start to move.

You go back and forward between these modes using the dropdown boxes at the top of the screen.



4 Change the world

The Sodaconstructor program allows you to build a creature by pointing and clicking your mouse on the screen. At the left-hand side of the screen is the control panel, and at the bottom there are three slide controllers – these let you change the laws of the world your creature lives and moves in. You can change gravity, friction and the stiffness of the springs.

- **Gravity** (g): If you turn the gravity up, your models will be squashed by their own weight. Turn it down low and your model will float. You can even turn gravity upside down using the popup menu.
- **Friction** (f): Friction slows moving masses. Apply lots of friction to your model and it will look like it's moving in treacle. Reduce friction and your creature can move fast but it might wobble out of control.
- **Spring stiffness** (k): This sets the springiness of your model's springs: weak springs make your model go floppy, very stiff springs are strong, but can make your model too jittery.

5 Making muscles move

You can turn any spring into a muscle by attaching it to the energy wave that repeatedly gives it pulses of energy. Remember that only muscles can power your creations to make them move.

When you click on the spring you want to 'muscle-ise' you will notice that a small circle appears around the centre point of the spring, and also that on the left hand side of the screen, in the control panel, there will be a short line below the wave that also now has a circle. Using your mouse, move this line (it will control the muscle you've selected) to some point on the wave.

When you switch from *create* to *simulate* mode (using the drop-down boxes) the wave will start to move, and as the wave passes through the muscle control line you have added, the muscle will expand and contract in time with the wave passing through.

You can build graceful moving creations by having muscles work in a synchronised way. You can make the wave move faster and change its strength (ie the amount that the muscles expand and contract by) using the sliders on the control panel. (See 'It walks!' for tips on how to build a really simple creature, to get you started.)

6 Saving and sending models

You've started to create, so you will want to save your work. You need to register with the site to allow this, but that's free and easy to do. The first time you use it, press the 'file' button (above the muscle wave in the top left of the Sodaconstructor window) followed by the 'Sodaplay login' and 'new users click here' buttons and register. Sodaplay will email your account details to you, and then you follow the instructions to activate your account. You must be logged into your Sodaplay account to save your models. To save the model, press the 'file' button (above the muscle wave in the top left of the Sodaconstructor window), then press the 'save current model' button and choose a name for your creation. You can also send saved models to your friends (see the Sodaconstructor site for details).

7 Loading models

To load a model, you press the 'file' button (above the muscle wave in the top left of the Sodaconstructor window) to switch to the browser interface. This will give you a list of available models in the middle of the screen, click on a model name and you will see a preview of the model on the left. To get a model, double click on a model name or select a model and press the 'get selected model' button. Remember that when you load a model it replaces the current model you have on screen, so save the current model first if you want to play with it again later.

8 Read the tutorial sites and ask for help

You now have the Sodaconstructor basics, the rest depends on your skill and a little help from the worldwide Sodarace community. There are forums where you can ask other constructors for help. Please use this facility sensibly. It's a brilliant resource for help, so don't post silly messages – the forums will not like it! The tutorial sites are full of expert information about basics and advanced constructing. You can also see some of the wonderful creations others have made, which should inspire you to create more Sodacreatures.

9 Know your forums

There are currently four user forums at the centre of Sodarace, each with a different theme.

- The **sodarace league** forum is used to upload and share your races, contenders and terrains and to discuss issues about the race software and results.
- The **artificial intelligence** forum discusses topics to do with artificial intelligence and how it can be used in Sodarace. It's full of interesting tips.

- The **model maker** forum is the place to share your model-making tips and interact with the Sodaracing community.
- Finally the **community discussion** forum, this is the forum for general discussion about community issues.

For all these forums, make sure that you read the rules regarding appropriate behaviour before posting.

10 Ready to race

Once you have a racer (a moving soda creation), you might want to learn to race it. The race software requires you to download the correct Java plug-in for your browser. The standard Microsoft version of Java doesn't have all the abilities needed, so follow the instructions on the Sodarace home page to install the correct version of Java. Look in the Sodarace league forum for hints and tips, or ask for help if you are having problems.

Once the new Java is installed, check by going to the Sodarace league forum and opening a race (a race is a link to a file ending '.jnlp'). This is how races are stored to be rerun whenever you wish.

Follow the race page instructions on the main Sodarace website. This will give you instructions on how to add your racer to races that have already been written, or write your own race to challenge others. You can even add your racer to a race with an AI-racer and see how your creations can compete with machine-learning.

For the races to work, you need to have an extra tool on the constructor, the fixed-bar-springs. These look like blue springs, but other objects can't pass through them, so they can be used to build terrains to race over.

Getting to grips with the race application tool takes a little practice, but the forums and help pages are there to get you going. It will be worth the effort. Why not share your experience with others in the forum or set up your own tutorial website? That's the Sodarace philosophy: 'Tools not Rules'. It's up to you to explore and push the races to the limit!

Hooke versus Newton

The laws of Physics that form the basis of Sodarace are so simple that they fit on a Post-It™ note. That is the beauty of Physics. The springs in Sodaconstructor work using rules discovered by British scientist Robert Hooke (1635-1703), who found that the force a spring exerts is directly proportional to how much it extends. This is called Hooke's law (Discovering a law of nature is a great way to be immortalised). Hooke also invented the iris diaphragm in cameras, the universal joint used in motor vehicles, the balance wheel in a watch, was the originator of the word 'cell' in biology, helped develop the microscope and worked with Sir Christopher Wren in rebuilding London after the Great Fire of 1666. Not bad going really. He also had a famous feud with Sir Isaac Newton (1642-1727), as Hooke felt that Newton had taken his ideas without giving him due credit. Interestingly, Sodaconstructor uses the laws of motion discovered by Sir Isaac, so Hooke and Newton work together now in Sodaconstructor at least.



Newton's work on light and gravity are well known, but he was also an alchemist, a member of parliament and Warden of the Royal Mint. He successfully foiled the coin counterfeiters, who were running riot at the time, by exchanging all the coins in the country for better designed ones: a task that required his enormous attention to detail and ability to improve the way people did tasks: both skills that would have made him a great programmer. He also set up a spy network worthy of Walsingham (see page 12) saying of his foes 'Criminals, like dogs, always return to their vomit'. Newton also invented the cat flap... the result of kittens ruining his optics experiments.

Locked-In syndrome

One of the worst medical conditions must surely be Locked-In syndrome. It leaves you with all your mental abilities intact but totally paralysed, except perhaps for the blink of an eye. A perfect, working mind is locked inside a useless body: the sufferer can sense everything around but is unable to communicate with anyone. Despite this, one of the most uplifting books I have read is *The Diving Bell and the Butterfly*. It is the autobiography of Jean-Dominique Bauby, written after he woke up in a hospital bed with Locked-In syndrome. In the book, he describes a life with Locked-In syndrome, including how he communicated, not only with medical staff, friends and family, but also how he came to write the book without any technological help.

The book was written using a heroic form of face-to-face interaction. Put yourself in his position, waking up in a hospital bed. What would be the best way for you to write a whole book? You have only a helper with a pen and paper to write down your 'words'. The only movement you can make is to blink your left eyelid.

How did Bauby do it?

Bauby's helper read the alphabet aloud ('A, B, C...') When the letter he was thinking of was spoken, Bauby blinked. The helper would write that letter down and then start again, letter after letter. Try it with a friend – communicate your initials to them that way. Now imagine that that is the only way you can talk to anyone. I hope your name isn't Zebedee Zacharius Zog or Zara Zootle.

Bauby realised that the 'A,B,C' method could be improved upon. He had been the Editor-in-chief of the French women's magazine *Elle* before he became ill, so he knew about language. He knew that some letters are more common than others in natural language, so he got the helper to read out the letters in order of frequency in French, 'E...S...A...R...' That way the helper got to the common letters more quickly. A similar trick has been used through the ages to crack secret codes – (See the Beheading story on page 12) and for doing the crossword-like puzzles called cross-references (try one on page 18).

Now, as a computer scientist I immediately start to think that I could have made his life so much better (even without replacing the human helper with blink detection gadgets and the like). In the worst case, perhaps dictating a story where someone snores 'Zzzzz' would take 26 questions per letter.

On average, in the course of dictating the whole book, roughly 13 letters will be said per letter dictated. Bauby's modification improves things but the worst case is still 26. Thinking as a computer scientist, the problem is a search problem (searching for one letter in 26) and the solution he used is known as linear search. Other search algorithms are far better. From some simple computer science that I learnt as an undergraduate, I know that a search through 26 things only needs at most five true/false or blink/no blink questions – not 26.

Learning from a children's game

How do we do it? By using the same strategy as is used in the children's game of 20 questions. It is a search problem too – a search to find the name of a famous person out of thousands – and yet it does not take thousands of questions to win. Played well, you do not ask as the first question 'Is it Nelson Mandela?', the equivalent to 'Is it E?' Rather you first ask: 'Are they female?' and so rule out half the possibilities whatever the answer. The equivalent question for the alphabet is 'Is it before N?' Try it – start with 1 million and see how many times you have to halve it before you get down to one. 1,000,000 ... 500,000 ... 250 000...


A similar trick has been used through the ages to crack secret codes

Keep asking questions like that about letters rather than famous people and you get down to a single letter in no more than five questions. Tweak it based on letter frequencies and you can do even better for the common letters.

Bauby should have got the helper to ask such halving questions. Think about it. Five questions at worst rather than 26, multiplied up by all the letters in his book. If only he had known some computer science, how much easier his life would have been.

Now we have worked out a method we can think how we could automate it with suitable technology. How wonderfully computer science can improve lives.

But wait a minute. Perhaps the computer scientist would have ensured his book was



never completed and his life was even more a hell. Perhaps we should have started with the person rather than our bright ideas. What if blinking is a great effort for him? His solution involved him blinking only once per letter. Ours requires him to blink five times. Multiply *that* by a whole book. Furthermore, his solution is easy for anyone to walk in and understand. Ours is complex and might need some explaining before the visitor understands and Bauby is not going to be the one to do the explaining.

It worked for him!

One thing is certain about Bauby's solution – it worked for him. He wrote a whole book that way, after all. Perhaps the helper did more than just write down his words. Perhaps they opened the curtains, talked to him about the outside world or just provided some daily human warmth. Perhaps the whole point of writing the book was that it gave him an excuse to have a person there to 'talk' to all the time.

Replace the person and perhaps you have replaced the one thing that was actually keeping him alive. In an extreme 'usability situation' such as this, the important thing is that the user really is involved throughout the process. They are the ones who ultimately have to make it work for them, not only technically but also emotionally and socially. Otherwise we may devise a 'solution' that is in theory wonderful but in practice hell on earth for the user.

As you can see, computer scientists have to think about so much more than just computers.

Find out about Jean-Dominique Bauby and what life is like with Locked-In syndrome by reading: *The Diving Bell and the Butterfly* by J-D Bauby, Fourth Estate.

How do computers become so clever?

Computers do some miraculous things. A computer can beat the world champion at chess, even fly a plane more skillfully than a human. How do they do it? How can a lump of silicon and wire appear to be cleverer than a human? Everything that a computer does that is intelligent is ultimately down to a person – a computer scientist in fact – writing clever instructions: rules to be followed. Everything you have ever seen a computer do, was just the result of it obeying the rules written by a computer programmer years earlier. Even a piece of paper can play these games as well as humans if it contains such rules. In fact:

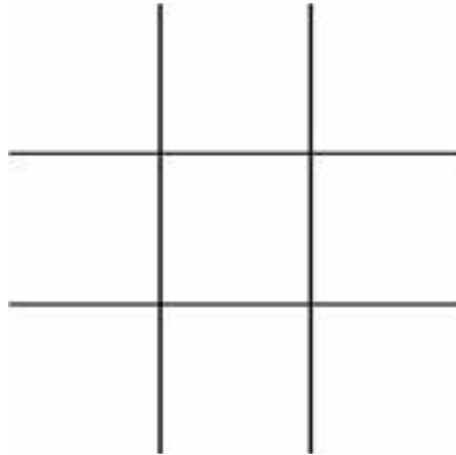
This piece of paper (let's call it Pete) can play better noughts and crosses than you can!

You will struggle to beat Pete at any rate! Try.

The instructions below give you a list of the moves that Pete the paper makes. If you are playing against Pete, you make whatever move you want to make after each one of Pete's moves. Pete gets to go first and is X. Follow his instructions as they are written.

Quick facts

By the beginning of the 1990s, a Hallmark greeting card embedded with a microchip that allowed the card to play 'Happy Birthday' contained more computing power than existed on the entire planet in the early 1950s. The one I got for my last birthday played 'Agadoo'...Arrghhhh!



Pete's Move 1:
Draw an X in a corner for me.

Your Turn:
Go where you like.

Pete's Move 2:
If no-one went there already then draw an X in the opposite corner to my move 1.

Otherwise put an X in a free corner for me.

Your Turn:
Go where you like.

Pete's Move 3:
If there are two Xs and a space in a line (in any order) then put an X in that space. I win!

Otherwise if there are two Os and a space in a line then put an X in that space. Ha!

Otherwise put an X in a free corner for me.

Your Turn: Go where you like.

Pete's Move 4: If there are two Xs and a space in a line (in any order) then put an X in that space. Gotcha! I win!

Otherwise if there are two Os and a space in a line then put an X in that space for me.

Otherwise put an X in a free corner.

Your Turn:
Go where you like.

Pete's Move 5:
Put an X in the free space for me.

Why not use Pete's moves above when you next have a game with your friends.

Obviously, if you're playing against friends, Pete's moves are the moves you make and your friend will make the moves labelled 'Your turn'. You will be invincible.

This is all a computer program is – a list of instructions that the computer can follow. The instructions for the computer have to be written very precisely in special languages so that the computer can follow them without understanding them, but the idea is the same.

Computers can only do things that the programmer has thought of – if things aren't as expected it won't seem so clever. I wrote the above rules expecting the paper to go first but what if it has to play second? Does it still seem so clever? That is the skill of the programmer: writing rules for every eventuality. Have a go at writing some better instructions for player 2 at noughts and crosses.

A puzzle, spies ... and a beheading



A puzzle about secrets

Sisters Amy and Elinor live together. Their cousin Emma wants to send them secret messages. She doesn't want Amy to read her messages to Elinor and vice versa. Amy buys them all small lockable notebooks for Christmas. They are normal notebooks except that they have a lock that can be locked shut using a small in-built padlock. The padlock can be opened with a single key. Amy suggests that they write messages in their notebook and post it and the key separately to the person that they want to send the message to. After reading the message that person tears the page out and destroys it, then returns the notebook and key. They try this and it appears to work, apparently preventing the others from reading the message. They exchange lots of secrets...until one day Amy gets a letter from Emma that includes a note added on the end from Elinor: 'I can read your messages. I know all your secrets'. She has been reading Emma's messages to Amy all along. She now wants them to know how clever she has been.

How did she do it and what does it have to do with the beheading of Mary, Queen of Scots?

Breaking the system

Elinor has of course been getting to the post first, steaming open the envelopes, getting the key and notebook, reading the message (and for the last one adding her own note). She then seals them back in the envelopes and leaves them for Amy. A similar thing happened to betray Mary,

Queen of Scots to her cousin Queen Elizabeth I. To find out how, read on.

A better way?

Emma suggests a solution to the problem of intercepted messages using the notebooks and keys in a similar way, but in which no keys are posted anywhere. To prove her method works, she sends a secret message to Amy, which Elinor fails to read. How does she do it? See if you can work it out before reading on...and what is the link to computer science?

Mary, Queen of Scots

The girls face a similar problem to that faced by Mary, Queen of Scots and countless spies and businesses with secrets to exchange before and since – how to stop people intercepting and reading your messages.

There are two ways to make messages secret – hide them so that no one realises that there is a message to read or disguise the message so that only people in the know can read it, or both. Hiding the message is called *Steganography*.



Disguising a message so that it cannot be read is called *encryption*. The girls in the puzzle discovered, just like Mary, that weak encryption is worse than no encryption, as it creates a false sense of confidence that the messages are secret.

Mary, Queen of Scots ultimately lost her life because her encryption was easy to crack.

She believed the encryption would protect her. It had given her the confidence to write what she otherwise would not have written.

House arrest

Mary had been locked up – under house arrest – for 19 years by Queen Elizabeth I, despite being captured only because she came to England seeking refuge from her cousin Elizabeth after losing her Scottish crown. Elizabeth was worried that Mary and her allies would try to overthrow her and crown Mary Queen of England if given the chance. Elizabeth thought that it would be better to lock her up before she even thought of treason. Towards the end of her imprisonment, in 1586, some of Mary's supporters plotted to free her and assassinate Elizabeth. Unfortunately they had no way of contacting Mary, as letters were not allowed either in or out by her jailors.

Mary, Queen of Scots lost her life because her encryption was easy to crack

Then the plotters had a stroke of good luck. A young priest called Gilbert Gifford turned up claiming he had worked out a way to smuggle messages to and from Mary. He wrapped the messages in a leather package and hid them in the hollow bungs of barrels of beer. The brewer delivered the beer to Chartley Hall where Mary was held and the packages retrieved by one of Mary's servants. This (a form of steganography) was really successful, allowing Mary to exchange a long series of letters with her supporters. Eventually the plotters decided they needed to get Mary's agreement to the full plot. The leader of the coup, Anthony Babington, wrote a letter to Mary, outlining all the details of the plot. To be absolutely safe, he also encrypted the message using a cipher that Mary could read (or 'decipher'). He soon received a reply in Mary's handwriting, also encrypted that showed that Mary agreed to the plot but also asked for the names of all the others involved. Babington responded with all the names. Unfortunately, unknown to Babington and Mary the spies of Elizabeth were reading everything they wrote – and the request for names did not even come from Mary.

Spies

Unfortunately for Mary and Babington, all their messages were being read by Sir Francis Walsingham, the ruthless Principal Secretary to Elizabeth and one of the most successful spymasters ever. Gifford was his double agent – the method of exchanging messages had been Walsingham's idea all along. Each time he had a message to deliver, Gifford took it to Walsingham first, whose team of spies carefully opened the seal, copied the contents, redid the seal and sent it on its way. The encrypted messages were a little more of a problem, but Walsingham's code breaker managed to break the cipher. The approach, called *frequency analysis*, which works for simple ciphers, involves using the frequency of letters in a message to guess which is which. For example the most common letter in English is E so the most common letter in an encrypted message is likely to be E. It is actually the way people nowadays solve crossword like code-puzzles known as cross-references that can be found in puzzle books (try the one on page 18). The trick can also be used to help people who are paralysed (see the story about Locked-In syndrome on page 10). Walsingham now had the key that allowed him to read even encrypted messages.

Can you break the code on page 18?

A beheading

When Walsingham read Babington's letter, he knew that he had the evidence to hang him, but he let the letters continue so that when Mary replied, Walsingham and Elizabeth finally had the excuse to try her too. Up to that point (for the 19 years of her house arrest) Elizabeth had not had strong enough evidence to convict Mary – just worries that Mary would be a magnet for plotters. Walsingham wanted more evidence though, so he forged the note asking for the names of other plotters and added it to the end of one of Mary's letters, encrypted in the same code. Babington fell for it, and all the plotters were arrested. Mary was tried and convicted. She was beheaded on 8 February 1587.

Private keys... public keys

Let's go back to our secret-swapping cousins. How does Emma's method get round the problem of her messages being intercepted and read? Her main weakness is that she has to send Amy the key, as well as the locked message – if the key is intercepted then the lock is worthless. An alternative way means that she doesn't have to keep sending the key to Amy. Suppose Emma wants to send a message to Amy. She first asks Amy to post her notebook (without the key but left open). Emma writes the message in Amy's book then snaps it locked shut and posts it back. Amy, who has kept the key safe all along, opens it secure in the knowledge that the

key has never left her possession. This is essentially the same as a method known by computer scientists as *public key encryption* – the method used on the Internet to protect the exchange of messages and that allows the Internet to be secure. In this scheme, keys come in two halves: a 'private key' and a 'public key'. Each person has a secret 'private key' of their own that they use to read all messages sent to them. They also have a 'public key' that is the equivalent to Amy's open padlock. If someone wants to send me a message, they first get my public key, which anyone who asks can have. It is used to encrypt the message (close the padlock), but is no use to decrypt it (reopen the padlock). Only the person with the private key (the key to the padlock) can get at the message. So messages can be exchanged without the important decryption key going anywhere. The message can't be intercepted.

Would this have helped Mary, Queen of Scots? No. Her problem was not that she exchanged keys but that she used a method of encryption that was easy to crack – in effect the lock itself was not very strong and could easily be picked. Walsingham's code-breakers were better at decryption than Babington was at encryption.

If you want to find out more about spies, encryption and even some of the computer science behind it then why not read *The Code Book* by Simon Singh, Fourth Estate.



Computers that work to our strengths?

One important thing about computer programs is the way that they present information to the people using them. In the early days of computing, people interacted with computers by typing commands and getting written results. Nowadays computers use 'GUIs' (Graphical User Interfaces) to represent information with graphics. Used well, GUIs make a big difference to the ease of a particular job. To see how much difference the way that information is organised and presented can make, let's play a game called Spit-Not-So. Write down the words:

SPIT NOT SO AS IF
IN PAN FAT FOP

- 1 The first player chooses a word that is on the list and crosses it out.
- 2 The first player writes the word down in front of them.
- 3 The second player then does the same thing choosing a different word.
- 4 The players take turns to do this until one person wins.

The winner is the first player to hold three words containing the same letter.

An example game might go:

Player 1 takes NOT Player 2 takes SPIT

Player 1 takes FAT Player 2 takes PAN

Player 1 takes FOP Player 2 takes IF

Player 1 takes SO ...and wins holding NOT, FOP and SO – 3 words with 'O'.

Play a few games to get the idea, then go to the [cs4fn website](#) for some sneaky tips and to see the link to GUIs!

Numbers game

2

The number of letters transmitted over the Internet before it crashed for the first time. The Internet was born on 20 October 1969 with the first transmission of data. The letters L and O were transmitted but the system crashed when the G of LOGIN was entered from a computer at the University of California and sent to another one at a research centre at Stanford, near San Francisco.



Mathemagic

Magic, maths and computer science

Pulling a rabbit out of a hat or making the Statue of Liberty vanish are impressive feats of magic. Magicians are in many ways like computer scientists: a magician must find a method to solve a problem, that problem being, say, making the rabbit appear or the Statue vanish, but without the audience realising how it's done. A good magic trick is a combination of method and presentation, in some ways like a computer program: the computer software must have a method to solve the problem (in computer science we call this method, or series of steps, an algorithm), but, unlike magic, software must present the results to the user so they can understand them.

A mind reading trick with a pocket calculator

Here is a mind reading trick to try with a pocket calculator. Remember that the method (the secret algorithm) and the calculator (the hardware) do the work for you, but you are the one who needs to provide the presentation (you're the 'user interface' here) to make it mysterious and magical.

- 1 Have someone secretly select a three-digit number and enter it twice into his or her pocket calculator. (For example: 123123) Have them concentrate on the display. You will try to discern their thoughts. (Magic Presentation User Interface needed here!)
- 2 From across the room (or even over the phone if you want), announce that you predict this number is exactly divisible by 11. Have them verify this by dividing by 11 to find a new whole number with no fractions. Magic!
- 3 Announce that you feel this new number now on the calculator display is exactly divisible by 13. Have them verify it. More Magic.
- 4 Now with the number left on the display have them divide by their original three-digit number.
- 5 Mysteriously announce that the final answer is 7. The Magical Finale.

The secret mathematical algorithm revealed

For your audience hopefully, if you presented magically, this will all look inexplicable, but as a computer scientist you should ask 'But why does this work?' When you look at the mathematics, the answer jumps out. Entering a three-digit number twice (123123) is equivalent to multiplying the three-digit number by 1001 ($123 \times 1001 = 123123$ – try it. It works for any number). Since $1001 = 7 \times 11 \times 13$, their original six-digit number will be divisible by 7, 11, 13, and their originally selected three-digit number.

Understanding how the trick works means that you can come up with your own variations, if rather than have the final prediction come out as 'lucky 7', you want it to be 'unlucky 13' what would you do?

Magic and computer software

It's not surprising that many mathematicians and computer scientists are interested in magic tricks. Working out ways to solve problems, whether predicting a chosen card in a trick or how to reduce the amount of digital data in an MP3 music file without the listener noticing, are very similar. The difference is that computer scientists want to tell the user how it's done. Magicians must keep the method a secret, never revealing it to the audience.



Computer science at the movies

Computer scientists develop the advanced digital technologies that make many film special effects possible. Their work and visions help shape the futuristic worlds that form the backdrop to many science fiction and fantasy films. Will criminals, for example, one day need to rip out their eyeballs to escape detection as in the film *Minority Report* or is biometric (biology)-based identification just a fantasy?

Stealing digits



In the future shown in the film *Minority Report*, biometrics – security systems based on biological features like eye scans, face recognition and fingerprints – are an everyday part of society. Other films regularly have biometric locks protecting vaults and top-secret control centres. Biometric security is not fiction though. It is already here, and not just in high-tech James Bond situations. The US is already using it at immigration desks and several makes of car have biometric ignitions. Biometrics is a popular technology, as it is so hard to forge. PINs can be stolen or forgotten and cards or keys can also be lost, but your fingers and eyes go everywhere with you – or do they?

While having eye transplants, as in *Minority Report*, to open a lock is still a long way in the future (let alone the even more extreme situation seen in *Face-Off*, where Nicholas Cage and John Travolta steal each others identities by having their faces transplanted), a grisly biometric theft has already happened. Carjackers in Malaysia, on finding the car they were stealing had a fingerprint recognition system to start the engine, cut off the tip of the owner's finger and stole that too. Which makes you think twice about using biometrics that don't need a living body attached!

Sodarace (see page 6) was featured on the official movie website for *Terminator 3* under the headline: **'Before robots can rule the world, they have to learn to walk.'**

Computing with the code of life

Computers and DNA: 'That's an odd combination', you might say, but in the near future it may be that the fastest computers on the planet are built, not from electronic bits, but from biological ones.

Electronic computers will soon face problems. We can pack more and more computing power onto a silicon chip, but this means having to build smaller and smaller parts in the silicon, and that's technologically very tricky as there are real physical limits on how small we can ultimately go. This is bad enough when physics causes problems, but worse still, geometry is against us too. As we increase the area we use on the chip, the space along the edges of the chip grows much more slowly. This means there is no room on the perimeter to put the wires we need to get all the data in and out. So what's the solution?

One possible approach is to start to build computers from other materials. Step forward DNA. DeoxyriboNucleic Acid is the chemical stuff that our genes are made off, it's the 'code of life' that tells our cells how to work, our eyes to be brown or our hair to be curly.



DNA works because it is a long string of chemical instructions; these instructions are in an alphabet with just four letters A, T, C and G, which correspond to the names of the four chemical 'bases' that make up DNA.

In 1994, computer scientist Leonard Adleman had the brilliant idea of using DNA to solve mathematical problems. He chose to begin with the famous 'Travelling salesman problem'. In this problem, a salesman has to visit a number of towns (let's refer to this number as N) and use the shortest route to visit them all (he needed to keep his travel expenses as low as possible). This sounds simple, and it is with just a few towns, but try it when N is 100 or 1,000. This is one of those classic problems that suffers when you scale it up: there is no known way to calculate the best solution quickly for a large N value.

Adleman coded up this problem on bits of DNA for just seven cities. Different strands of DNA were chosen to represent each city, in a particularly clever way, so that when Adleman actually mixed the solutions together in a test tube, the way the DNA chemical bonds joined up solved the problem, with the chains of resulting DNA representing routes. In fact, the different, joined strands in the tube gave all the possible solutions to the problem. The only difficulty Adleman faced was going through the goo to find the answer. But he got it in the end.

Three years after Adleman's experiment, the University of Rochester developed logic gates made of DNA. Logic gates are the fundamental electronic parts normally used to build a computer. They are the parts that allow the calculations to take place. They act like tiny switches with rules that say what to output when certain signals are input and have names like AND gates OR gates or the exotic sounding XOR (exclusive OR) gates. The researchers found that they could build DNA structures that followed the rules of logic. For these logic gates, the inputs were bits of DNA rather than electronic signals, and the gate then chemically spliced these fragments together to get the single required output – a very clever bit of biology. The researchers believe that these logic gates might be combined with larger DNA microchips to create a breakthrough in DNA computing.

DNA computations are fast and accurate, and the materials used are biodegradable and cheap. There is DNA in every cell. DNA also has the ability to contain a massive amount of information. If you take one-pound weight of DNA, it could store more information than all the electronic computers ever built. It's been suggested that the computing power of a teardrop-sized DNA computer, using the DNA logic gates, will be more powerful than the world's most powerful supercomputer and, unlike conventional computers, DNA computers will be able to perform all their calculations at the same time. We call this 'parallel processing', a very interesting way to build computers.

'In the future, computers could be build from biological, rather than electronic, material'

It will be fascinating to see how this new technology develops and is applied. Exciting times lie ahead for computer scientists and biologists, as they work out new ways to build us even faster computers.

Who wants to be the weakest millionaire?

TV game shows like Who Wants To Be A Millionaire? and The Weakest Link are very popular. Part of their popularity lies in the fact that they have interesting rules for the contestants to play against. These rules give the shows their tension, but someone has to make the rules up to begin with. So how would you go about designing the format for a good game show?

Cash in a box

Well, we need some prize money – the amount of cash that the player will walk off with in their pocket. Lets add a prop, a shiny box that our host puts the money into, and lets label this box with big letters 'PRIZE'. So, a very simple game show would ask the contestant a question. If they get it right, they win £100, for example, which our host puts into the 'PRIZE' box. At the end, the player gets what is in the box, so 'PRIZE' can refer to the box but also, more importantly, the value of money in the box.

Simply answer the question

How could we write the instructions for this simple quiz? Well how about:

```
If (Answer is correct)
    {PRIZE = £100}
```

We have used $PRIZE = £100$ to mean put £100 in the prize box. So, as we wanted, if the answer's right, the value in PRIZE goes up to £100.

Double your money!

Simple enough, now suppose that we make the game a little more challenging, so what will happen is that each time the player gets the answer correct they double their money. If they get the answer wrong then they lose the lot (a bit like Who Wants To Be A Millionaire?). So, how would we write that?

Your starter for 100?

We have a problem to begin with. If there is no money in PRIZE to start with, then doubling it when you get the answer right will give you twice as much nothing. Hardly fun to watch. So we have to put some cash in the box to begin, (or we give them a simple question to start with). Let's be kind. Let's put £100 in the box to start.

```
PRIZE = £100
```

If they get the question right we can then say that $PRIZE = 2 * PRIZE$. What this means is that the new money in the box will be twice the money that was in the box beforehand (we've used * for multiplication).

Twice in the box

We have our starting situation (we need something to double remember)

```
PRIZE = £100
```

And if the player gets the question right

```
If (Answer is correct)
    {PRIZE = 2*PRIZE}
```

But we were being nasty. What happens when they get the question wrong? They need to lose all the cash! That is we want to set $PRIZE=£0$: no cash in the box. How could we write this?

We could simply write

```
If (Answer is wrong)
    {PRIZE = £0}
```

After all the answer is either right or else it's wrong, and here's an idea about how to write this:

```
If (Answer is correct)
    {PRIZE = 2*PRIZE}
else
    {PRIZE = £0}
```

We have a single line. 'If the answer is correct, the cash doubles or else (if the answer is wrong, which is the only other option) the cash is lost.'

So for our Who Wants To Be A Millionaire?-type quiz, we have the rules.

```
PRIZE = £100
```

```
If (Answer is correct)
    {PRIZE = 2*PRIZE}
else
    {PRIZE = £0}
```

Round and round again

But we want more than one round of the game. Each additional round should become more exciting, as the prize money grows with each correct answer. How do we write this? Well we need some way to say that we do the same thing time and again, as long as we are happy to do it, but that at some stage we want to stop. Hmm. How many rounds do we want to have?

Quick quiz question

Why did a 13-year old girl from Brittan Elementary School in the USA make headlines across the world for saying: 'Look at this. I'm a grocery item. I'm a piece of meat. I'm an orange.'

Answer

She was outraged that her school had introduced electronic tags to keep a constant track of student movements within the school. As a result, the school suspended its use of the system, which just goes to show there is more to computer science than technical brilliance. To be a successful innovator, you have to understand people too.



Count up

Let's do our quiz round eight times:

```
Do 8 times {a round}
```

So we have a way to have eight rounds, and we know how to do each of the rounds, and how to start the prize fund in the box, so, let's put them together:

```
PRIZE = £100
```

```
Do 8 times
```

```
{  
  If (Answer is correct)  
  {PRIZE = 2*PRIZE}  
  else  
  {PRIZE = £0}  
}
```

Does this make sense? We start with setting the contents in the prize box to £100 before we do any rounds, so that first 'PRIZE = £100' will go outside the 'Do it eight times' bit. For each round if the answer is correct the prize money doubles, and as it's the same PRIZE box we use in each round, the money will continue to double for our lucky player.

That's the wrong answer!

What happens if the contestant gets the answer wrong? Well, looking at our 'else' rule, if in any round the answer is wrong, the prize in the box goes to zero. And that's a problem for our player because even if they get the next question right it would just double nothing! Unkind perhaps but would watching the contestants play for, say, five rounds, answering the questions for nothing, make good television?

Therefore, if a contestant gives a wrong answer, we want to stop the quiz and take away the prize money. The game will be over. So, let's add that to our set of rules. Put in the word 'Break' to mean just that: that we jump out of the rounds and end the game.

```
PRIZE = £100
```

```
Do 8 times
```

```
{  
  If (Answer is correct)  
  {PRIZE = 2*PRIZE}  
  else  
  {PRIZE = £0, Break}  
}
```

Roll the credits

Well done. So there we have the rules for a 'Millionaire'-type game, explained more-or-less in English and fairly easy to write down. But if you've followed this through, you've actually understood your first computer program. Computer programs look just like this, a series of rules to control the way numbers (or, in our case, cash) are moved around. We call this set of rules an algorithm, the way that we write the instructions is called syntax. And the rest of computer programming? That's just practice.



Code-breaking

How good are you at code-breaking? Here is a code crossword. No clues, just the code to break. The key is at the top. All of the letters of the alphabet appear in the key and grid but which letter is which? Work out the code to reveal one of the greatest movie messages of all time. And just to make it a bit more fun, you also have to answer the following questions about the grid:

- 1 Name a caffeine-loaded programming language.
- 2 What animal has a crush on you?
- 3 Name the lump of clay that ran away to have fun.
- 4 Name an explosive Blondie song?

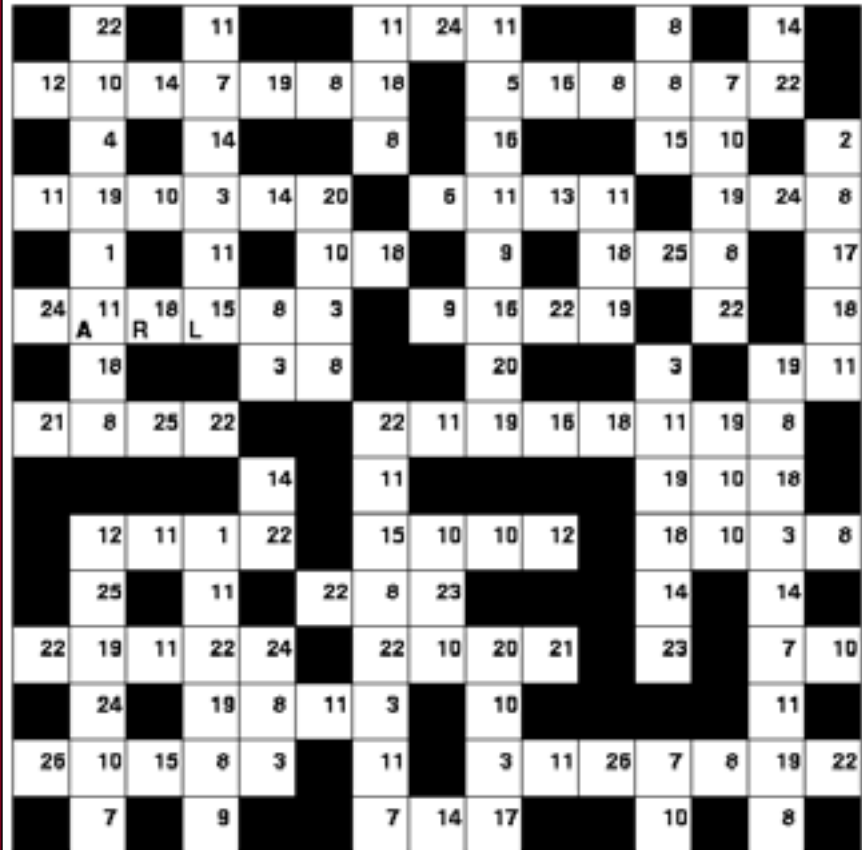
The answer will be posted on the cs4fn website at www.dcs.qmul.ac.uk/cs4fn/, where there are also some hints about code-breaking.

Cracking Codes

Cryptanalysis, the art of reading secret messages, was invented by Muslim scholars.

The earliest known description of the method needed for this puzzle and used by Walsingham to crack the messages of Mary, Queen of Scots (see page 12) was written by the great Arab philosopher, Al-Kindi in the ninth century.

1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26



14	7	22	12	11	20	8	7	10	-	10	7	8
20	11	7	24	8	11	18	25	10	16			
			22	20	18	8	11	3				

BrainAcademy, the Compute-Ability competition, is back! Enter NOW at www.brainacademy.qmul.ac.uk



BrainAcademy, the online talent spotting competition that gives the winners the chance of a big break into a top Computer Science career, is back. There are more prizes than ever, with the winner getting a prize tailored to their career plans.

Up for grabs are a place on an undergraduate computer science degree course in the Department of Computer Science at Queen Mary, University of London, starting in 2006 with all top-up fees paid; places on the Department's MSc Information Technology conversion course and on one of the Advanced MSc programmes (starting in 2006 with cash bursaries); and other unique career opportunities. The undergraduate winner will be guaranteed a fast-track interview for the popular Microsoft Student Intern Programme and, upon successful graduation, a fast-track interview for the coveted Microsoft Graduate Programme.

Further prizes include a summer work experience placement with top internet publishers ZDNet where you might have a go at evaluating the latest computer gadgets and gizmos, work experience with Omarketing, or a place on a student exchange programme at another top overseas University.

There is also a range of useful add-on and runner-up prizes, including a laptop, cash, books and software. The main prize will be tailored to the winner's interests and their career plans.

The first round involves taking part in a web hunt, answering questions related to Computer Science and IT. You will need to be able to solve problems and have good research skills. This year's theme is **Diversity**. The hunt will take you on a virtual round-the-world tour through different countries and cultures, learning about the men and women who have played a part in the history of computing. You will explore the way the arts, humanities, sciences and engineering all played a part in making Computer Science the diverse subject it is today. Get all the questions right and you will gain access to the programming level. Do you have the programming skills and creativity to take it further? Will you be able to write the best, most creative program and win?

BrainAcademy was first launched in 2003. The fun 'life-changing-prizes-game-show' caught the public imagination. The web hunt amassed 80,000 hits with 120 hopefuls making it through to the programming challenge, and a handful making it as far as the interview stage. The competition also won a Queen Mary's Drapers' prize. The 2003 winner, Adam Kramer, from North London, has just finished his first year at Queen Mary and is one of the Department of Computer Science's top undergraduate students. Adam was 17 when he entered the competition and a self-taught programmer: 'I wasn't expecting to win the competition, but I'm really pleased to be studying at Queen Mary'.

Do you know someone who has what it takes, but just needs a lucky break? BrainAcademy could be their chance.



Back (page) to the future.

What does the future hold for computer science? It's always a tricky question to try and answer, and many people in the past got their predictions splendidly wrong. Here are some of the best bloopers (allegedly).

1876

'This "telephone" has too many shortcomings to be seriously considered as a means of communication. The device is inherently of no value to us.' said a Western Union internal memo. Western Union is now one of the USA's largest telecom companies.

1949

'Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons.' wrote *Popular Mechanics* magazine. If they had been right you'd need a crane with every laptop.

1943

'I think there is a world market for maybe five computers.' said Thomas Watson, chairman of IBM, a company that later went on to revolutionise the home PC market.



1968

'But what ... is it good for?' said an engineer at the Advanced Computing Systems Division of IBM, commenting on the microchip. Now, of course, microchips run all the billions of computing devices on planet Earth.

1977

'There is no reason anyone would want a computer in their home.' said Ken Olson, president, chairman and founder of Digital Equipment Corp. If Ken had been right there would be a lot more table-top room in houses for dusting.

So what are your predictions for the future? Why not email them to us at: cs4fn@dcs.qmul.ac.uk ? We will feature the best on the cs4fn website and, who knows, you may even turn out to be right!



cs4fn is written by Paul Curzon, Peter McOwan and Gabriella Kazai of the Department of Computer Science, Queen Mary, University of London email: cs4fn@dcs.qmul.ac.uk

EPSRC Engineering and Physical Sciences Research Council

SodaRace is supported by EPSRC

 Queen Mary
University of London

Passionate about computer science?
www.dcs.qmul.ac.uk/cs4fn/