What Did Qubits Ever Do for Me? An Answer for CS2 Students

Robert Frohardt
Department of Computer Science
University of Colorado at Boulder
UCB 430
Boulder, CO 80309-0430
303-547-8023
frohardt@colorado.edu

Yingdan Huang
Department of Computer Science
University of Colorado at Boulder
UCB 430
Boulder, CO 80309-0430
303-489-8934
Yingdan.Huang@colorado.edu

Michael Main
Department of Computer Science
University of Colorado at Boulder
UCB 430
Boulder, CO 80309-0430
303-723-9527
main@colorado.edu

ABSTRACT
We show how to teach and motivate small quantum computer programs as a supplemental topic in a CS2 data structures class. A traditional example such as Shor's factorization [8] could be used, but we focus instead on the area of *quantum pseudo-telepathy games.* Examples in this area require less mathematics than factorization and are easy to motivate with short proofs that the problems solved have no solutions in a world of classical computing. A CS2 class is a good location to present this work because of the matrix storage and manipulation that's required.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *curriculum.*

I.m **[Computing Methodologies]:** Miscellaneous.

General Terms
Algorithms.

Keywords
CS2, quantum computing, quantum pseudo-telepathy.

1. INTRODUCTION
Undergraduate students are enticed by new developments in science and computing. For computer science students, one extraordinary area is quantum computing. The students' interest comes directly from the remarkable counterintuitive results—but this counterintuitive nature and some advanced mathematics makes it seem unmanageable to introduce the ideas to beginning students.

Nevertheless, there are aspects that are appropriate for a typical CS2 class. This paper describes the approach we took from the introduction of quantum computing to a comprehensive assignment that involves implementing a matrix class for simulating quantum computations. We include specific techniques for explaining the necessary concepts of qubits, superposition, interference and quantum entanglement using a particular problem in the area of quantum pseudo-telepathy. The CS2 students can understand the proof that the problem is unsolvable using classical computing. Next, they quickly gain a qualitative understanding of the quantum computing solution—particularly the aspect that Einstein objected to as a "spooky action at a distance." With an additional introduction of the matrix representation of quantum computer programs, the students can implement a simulation of the quantum computing solution using a Kronecker product.

ITiCSE 2010, June 26–30, 20010, Bilkent, Ankara, Turkey.

2. THE THREE-STONES PROBLEM
One way to motivate any new technique is to present a problem that simply cannot be solved without the technique. Many such problems in quantum computing are characterized as pseudo-telepathy games—a name that was chosen because it appears that the problems cannot be solved without instantaneous telepathy. (See, for example, the survey by Brassard, *et. al.* [2].) In our CS2 class, we begin by presenting one such game derived from an example of Greenberger, *et. al.* [3].

The game involves three astronauts and three aliens, so we pick three students from our class—Alice, Bob and Charlie—for the astronauts and three conveniently available teaching assistants for the aliens. The entire class is told the rules ahead of time:

1. Prior to the start of the game, the astronauts may meet to devise strategies and exchange whatever objects they may need.
2. The three aliens will then take the three astronauts to planet around different far-flung stars.
3. Once at the stars, the aliens will give one colored stone to each astronaut. There is also a guarantee: Either the stones are all blue stones (the *all-blue* case), or there is one blue stone and two red stones (the *1-2* case).
4. At this point, each astronaut must decide whether to keep his or her stone or to give it back—and they must do so quickly (without enough time to communicate at the speed of light).
5. In the all-blue case, the game is won if the astronauts keep an odd number of stones; in the 1-2 case, the game is won if the astronauts keep an even number of stones.

At this point, we have our students play the game a few times (although, because of time constraints, the students are taken to the far-flung corners of the classroom instead of to stars). Perhaps the students win sometimes, perhaps they lose—it doesn't matter because we're just trying to understand the rules. Then we pose a task to Alice, Bob and Charlie: Come up with a strategy that will *always* win, regardless of whether you're given all blue stones or one of the 1-2 cases. This challenge is presented at the end of a lecture, and the class is sent away to think.

At the start of the next lecture, we let them try a few ideas, but the classroom easily shoots them down.

"Let's always give back blue stones and keep red ones," the students might propose—but that won't work in the all blue case. A more complicated strategy might have Alice, Bob and Charlie each do something different with their stones, but each suggested strategy has at least one situation where it fails. This kind of strategy—where Alice, Bob and Charlie each decide ahead of time what to do with each kind of stone—is called a *deterministic strategy*. We ask the students to figure out the number of different deterministic strategies, and to justify their number. Some students might come up with a table like this, where a 0 means that an astronaut gives back the stone and a 1 means it's kept. Each row of the table represents a different possible deterministic strategy:

What does Alice do with…		**What does Bob do with…**		**What does Charlie do with…**	
a blue stone	**a red stone**	**a blue stone**	**a red stone**	**a blue stone**	**a red stone**
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
…and so on down to…					
1	1	1	1	1	1

We could go through the table row-by-row and find a situation where each strategy fails. For example, the first row (where every stone is always given back) fails if the aliens give the astronauts three blue stones. The row-by-row approach will successfully shoot down all 64 strategies, but it is time-consuming and there's a better, generalized proof that shoots down all strategies at once.

For the proof, we give a name to each of the six bits of any given row from the table of possible strategies. The bit in the first column is A_{BLUE}; the second column is A_{RED}; the third column is B_{BLUE};and so on. For the given strategy to work in the all blue case, we must have:

$$A_{BLUE} + B_{BLUE} + C_{BLUE} = \text{some odd number}$$

And for the three different 1-2 cases to all work, we must have:

$$A_{BLUE} + B_{RED} + C_{RED} = \text{some even number}$$
$$A_{RED} + B_{BLUE} + C_{RED} = \text{some even number}$$
$$A_{RED} + B_{RED} + C_{BLUE} = \text{some even number}$$

This gives us four equations. If one of the equations fails, then the strategy fails for the corresponding case. So, an always-successful strategy must have all four equations true. With this in mind, ask the students to add the left and right sides of all four equations and simplify things to come up with one total equation. They'll get something like this:

$$2A_{BLUE} + 2A_{RED} + 2B_{BLUE} + 2B_{RED} + 2C_{BLUE} + 2C_{RED}$$
$$= \text{some odd number}$$

This one equation must be valid if the given strategy is always successful. But the left side of the equation is even and the right side is odd—hence no deterministic strategy will always win the game.

The purpose of this little proof is to convince the students that there's no strategy that wins all the time. Some student may wonder about *nondeterminism.* "How about if we wait until we get the stones and then roll some dice to determine whether to give back the stones?" This is not a deterministic strategy, but it fares no better because the dice rolls just randomly select one of the flawed deterministic strategies. Sometimes the dice rolls might be lucky and win, but the dice don't always win.

Another possibility is for the astronauts to communicate with each other after they get their stones. "Hey, I've got a blue stone," Alice can shout to Bob. "What should I do with it?" If this were allowed, then a winning strategy is not hard to device, but such communication is forbidden. In the real test, the astronauts are at far-flung stars, and they will have to make their decisions quickly—much more quickly than speed-of-light messages could be sent and received.

And now the amazing part: It is possible to create some quantum computer programs that the astronauts can run after they get to their stars. The programs that they run depend on the colors of their stones, and the outputs of the programs direct their actions in a way that guarantees a win.

3. QUBITS

It's time to introduce quantum computing, starting with the basic data in a quantum program. Our students already know about ordinary bits. "A qubit is similar," we say. "It's a memory value, and when we examine it, it's value is either 0 or 1. But in the time before we examine a collection of qubits, they may be in a remarkable state called a *superposition of observable states."*

For example, suppose we have three qubits. The physicists represent one possible superposition this way:

$$0.7|111\rangle + 0.5|100\rangle - 0.5|011\rangle + 0.1|010\rangle$$

Each term in the expression has an amplitude (such as 0.7) and a triplet of bit values (such as $|111\rangle$). The square of the amplitude tells the probability of finding the triplet when the bits are examined. In this case, we have:

- 0.7^2 = 49% chance of finding three 1's ($|111\rangle$)
- 0.5^2 = 25% chance of finding $|100\rangle$
- $(-0.5)^2$ = 25% chance of finding $|011\rangle$
- 0.1^2 = 1% chance of finding $|010\rangle$

Physicists have proposed and built a number of different ways to implement small collections of qubits. The important point is that when a quantum computer program runs, the amplitudes attached to various triplets change. At the end of the program, we examine the qubits. The probability of finding any particular triplet of qubits depends on the amplitude attached to the triplet at that time.

How can this help with the three-stone problem? Before the astronauts leave the Earth, they'll set up a three-qubit quantum computer. Each astronaut takes one of the three qubits with him to the stars—perhaps carrying it in a contraption called an ion trap, which is one way to implement qubits. When each astronaut is shown the color of his stone, he uses that color to select one of two programs—the blue program or the red program—and he runs that program on his qubit. For example, if Alice is given a red stone, she runs the red program on her qubit and then looks at the qubit. A zero qubit tells her to give the stone back; a one means to keep her stone. The other two astronauts do the same.

Because of a property called *quantum entanglement,* when each astronaut runs a program it changes the amplitudes on all eight of the triplets in a mathematical way that we'll see in a moment. This is part of quantum mechanics that Einstein never fully accepted. He called it a "spooky action at a distance"[1] and spent much effort devising thought experiments to illustrate its fallacy—but experiments in the 1980s showed that entanglement occurs exactly as predicted by quantum mechanics.

We'll see our astronauts' blue and red programs in a moment, but for now, you should know this:

When the astronauts run their programs, the amplitude on any incorrect triplet becomes zero.

For example, if they were given all blue stones and they each run their blue program, then the amplitudes on $|000\rangle$, $|011\rangle$, $|101\rangle$ and $|110\rangle$ all become zero. (Those are the four combinations that lose in the all-blue case.) Hence, when they go to look at their three qubits, they always find a winning combination.

4. QUANTUM COMPUTER PROGRAMS

The presentation could be stopped here. But if you've used some previous programming examples involving complex numbers and matrices, you can continue to show exactly what the quantum computer programs look like.

The complex numbers and matrices are needed because the amplitudes that we've already seen may be any complex number, and these amplitudes are usually arranged into a single vector. We draw the vector as a column vector in which the columns' labels are the triplets, like this:

$$\begin{array}{c} \begin{bmatrix} 0.7 \\ 0 \\ 0 \\ 0.1+0.5i \\ 0 \\ 0 \\ -0.5 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} |000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle \end{array}$$

In this example, the triplet $|000\rangle$ has an amplitude of 0.7, meaning that the probability of finding that triplet is 0.7^2 = 49%. Notice that the triplet $|011\rangle$ has a complex number, $0.1 + 0.5i$, for its amplitude. For a complex amplitude, the Nobel laureate Max Born proposed the rule that the probability is the sum of the squares of the coefficients for the real and imaginary parts ($0.1^2 + 0.5^2 = 26\%$ for our example).

The advantage of arranging the amplitudes in a vector is that quantum computer programs can then be mathematically expressed as a certain kind of matrix.

A superposition of n qubits is represented as a column vector of height 2^n. And a quantum computer program to manipulate n qubits is always a unitary matrix of size $2^n \times 2^n$.

Our students haven't seen unitary matrices at this point, so we start by explaining the conjugate transpose of a matrix (obtained by transposing the matrix and then replacing each entry $a+bi$ with its complex conjugate $a-bi$). For any matrix U, this gives a new matrix $U^\dagger$—and the definition of a unitary matrix requires that $U^\dagger$ is the matrix inverse of U. We give a small example that will also arise later in the presentation:

$$U = \begin{bmatrix} 1/\sqrt{2} & i/\sqrt{2} \\ 1/\sqrt{2} & -i/\sqrt{2} \end{bmatrix} \quad U^\dagger = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -i/\sqrt{2} & i/\sqrt{2} \end{bmatrix}$$

$$UU^\dagger = U^\dagger U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

As a larger, but still simple, example, we ask our students to compute how the 3-qubit quantum system given above changes when we run this particular program (an 8×8 unitary matrix):

$$\begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \end{bmatrix}$$

The answer is obtained by a matrix multiplication:

$$\begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \end{bmatrix} \begin{bmatrix} 0.7 \\ 0 \\ 0 \\ 0.1+0.5i \\ 0 \\ 0 \\ -0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ 0.7 \\ 0 \\ 0 \\ 0.1+0.5i \end{bmatrix}$$

After running the program, the probabilities of finding various triplets have changed according to the rules of matrix multiplication.

The students inevitably want to know *why* quantum computer programs are unitary matrices. One honest answer is that physicists such as Born simply found that this mathematics provides the correct results for experiments with quantum systems. Moreover, modern-day physicists and computer scientists are now able to build small collections of qubits and the hardware that enacts any possible unitary matrix to manipulate the system. A variety of different physical mechanisms are used, such as the ion trap in 2008 that achieved eight qubits via calcium ions confined in electromagnetic fields. We provide some reading on this implementation [7], but the main emphasis now becomes a data structures assignment to simulate the action of the particular

[1] "*spukhafte Fernwirkung*"—from a letter that Einstein wrote to Max Born on March 3, 1947.

quantum computer programs that the astronauts can use to solve the three-stone problem.

5. THE ASTRONAUTS' PROGRAMS

For the three-stone problem, the starfaring astronauts begin (before leaving Earth) by creating a system of three qubits that is initially in this state:

$$\begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix} \begin{matrix} |000\rangle \\ |001\rangle \\ |010\rangle \\ |011\rangle \\ |100\rangle \\ |101\rangle \\ |110\rangle \\ |111\rangle \end{matrix}$$

If the astronauts were to measure the qubits right away, there would be a 50% probability of finding $|000\rangle$ and 50% probability of finding $|111\rangle$. But, of course, they don't measure the qubits right away. Instead, each astronaut takes one qubit to a far-off star. When each astronaut is given a stone, he or she chooses a program to run based on the color of the stone. There will be three programs in all (one run by each astronaut on his or her qubit).

The combined effect of the three astronauts running their three programs will change the amplitudes on the triplets according to some 8×8 matrix. For example, let's look at the case where they are all given blue stones. The effect of all astronauts running their blue programs is given by this quantum computer program:

$$\frac{1}{\sqrt{8}} \begin{bmatrix} -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The scalar, $1/\sqrt{8}$, simply means that each of the matrix entries is multiplied by this amount. When this program is applied to the starting state, the result is:

$$\frac{1}{\sqrt{8}} \begin{bmatrix} -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ 1/2 \\ 0 \\ 1/2 \\ 0 \\ 0 \\ 1/2 \end{bmatrix}$$

As promised for this situation of all blue stones, the final state has zero values for the triplets $|000\rangle$, $|011\rangle$, $|101\rangle$ and $|110\rangle$. Hence, the only triplets that the astronauts will ever see are ones where they keep an odd number of stones, and they are guaranteed to win the game!

The students will have questions now.

Why did the astronauts create that particular starting state? To some extent, the answer to this questions is contrived. The designers of this game explicitly created it so that it would have no solution in the world of classical physics, but that it would have a quantum computing solution that started from that one particular starting state.

Just how did you figure out that the combined effect of running three blue programs would be that matrix? For this question , we need to know about Kronecker products—a mathematical construct named after the 19th century German mathematician Leopold Kronecker, though he was not the first to use it [4].

6. THE SPOOKY MATHEMATICS OF KRONECKER PRODUCTS

To finish the presentation, we need to show how the 8×8 matrix is constructed from three separate programs that the astronauts run on their three separate qubits.

The three astronauts and their qubits are widely separated, which has a consequence: it is no longer possible to manipulate the qubits with an arbitrary unitary matrix. Instead, each astronaut must run a program for one qubit on his or her qubit. These matrices for manipulating a single qubit are 2×2 unitary matrix. Now, here's the entanglement part that Einstein found spooky:

When the three astronauts run their separate 2×2 matrices on their individual qubits, the result on the entire triplet is computed by an 8×8 matrix that is formed from the smaller matrices using the mathematical operation of Kronecker product.

Our students have not previously seen Kronecker products, but the mathematics can be explained with a small example. Suppose we want to compute the Kronecker product of these two matrices:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Start by writing a copy of the left-hand matrix with a big dot after each entry:

$$\begin{bmatrix} a_{11}\bullet & a_{12}\bullet & a_{13}\bullet \\ a_{21}\bullet & a_{22}\bullet & a_{23}\bullet \\ a_{31}\bullet & a_{32}\bullet & a_{33}\bullet \end{bmatrix}$$

Next, replace each dot with a copy of the right-hand matrix:

$$\begin{bmatrix} a_{11}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{12}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{13}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ a_{21}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{22}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{23}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ a_{31}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{32}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{33}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{bmatrix}$$

Do the multiplications that you have set up and get rid of all those inner brackets, giving one big matrix as your answer:

$$\begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} & a_{13}b_{11} & a_{13}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} & a_{13}b_{21} & a_{13}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} & a_{23}b_{11} & a_{23}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} & a_{23}b_{21} & a_{23}b_{22} \\ a_{31}b_{11} & a_{31}b_{12} & a_{32}b_{11} & a_{32}b_{12} & a_{33}b_{11} & a_{33}b_{12} \\ a_{31}b_{21} & a_{31}b_{22} & a_{32}b_{21} & a_{32}b_{22} & a_{33}b_{21} & a_{33}b_{22} \end{bmatrix}$$

Some questions you can ask the students: If A is an $n \times m$ matrix and B is a $q \times p$ matrix, then how big is $A \otimes B$? What's the formula for the entry in row i and column j? (As a hint for the second question, remind them of modular arithmetic.) Is the operation commutative? (No. In general $A \otimes B \neq B \otimes A$.) But it is associative. What algorithm would you use to compute the Kronecker product from two input matrices?

Our students designed two different algorithms, one of which used two nested loops to iterate through the rows and columns of the answer matrix. The other algorithm had four nested loops that iterated over the rows and columns of the two input matrices. Aferward, we asked the students to compare the time complexities of the two algorithms and to discuss which algorithm is easier to understand.

Once the students know how to form a Kronecker product, you can explain exactly how the astronauts 8×8 matrices are formed. Each astronaut chooses one of these 2×2 matrices to run on his or her qubit according to the color of their stone:

$$\text{BLUE} = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \text{RED} = \frac{1}{\sqrt{2}}\begin{bmatrix} -i & 1 \\ i & 1 \end{bmatrix}$$

Notice that the red matrix has two imaginary entries.

The combined effect of the three astronauts running their individual programs is the Kronecker products of their three matrices. For example, if all three are given blue stones, then the combined matrix is:

$$\text{BLUE} \otimes \text{BLUE} \otimes \text{BLUE} = \frac{1}{\sqrt{8}}\begin{bmatrix} -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

This is the matrix we have seen before for the all-blue case. Each of the 1-2 cases results in a different 8×8 matrix. For example, if Alice gets a blue stone and the other two get red stones, then the matrix is:

$$\text{BLUE} \otimes \text{RED} \otimes \text{RED} = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & i & i & -1 & -1 & -i & -i & 1 \\ -1 & i & -i & -1 & 1 & -i & i & 1 \\ -1 & -i & i & -1 & 1 & i & -i & 1 \\ 1 & -i & -i & -1 & -1 & i & i & 1 \\ -1 & -i & -i & 1 & -1 & -i & -i & 1 \\ 1 & -i & i & 1 & 1 & -i & i & 1 \\ 1 & i & -i & 1 & 1 & i & -i & 1 \\ -1 & i & i & 1 & -1 & i & i & 1 \end{bmatrix}$$

7. A CS2 PROGRAMMING ASSIGNMENT

With this background in place, we give a two-part assignment to our CS2 students :

1. Write a class for square matrices where the individual entries are complex numbers. The class must include operations for retrieving the size, setting and retrieving individual elements, matrix addition, matrix multiplication and the Kronecker product.
2. Use your matrix class in a program to verify that our three astronauts' quantum computer strategy will always win the three-stone game.

Some students ask for more information. Mermin [5][6] does a wonderful job of explaining these kinds of games (now called Mermin-GHZ games). With the background that we've provided, many students can also understand the paper by Brassard, *et. al.,* [2]. For amore general coverage of quantum computing, we point them to Aaronson's recent *Scientific American* article [1].

8. ACKNOWLEDGMENTS

Our thanks to our students for allowing us to try out these ideas.

9. REFERENCES

[1] Aaronson, S. "The limits of quantum computers", Scientific American (Mar. 2008), 62-69.

[2] Brassard, G., Broadbent, A. and Tapp, A. 2005. Quantum pseudo-telepathy. Foundations of Physics 35, 11 (Nov. 2005), 1877-1907. http://arxiv.org/abs/quant-ph/0407221

[3] Greenberger, D.M., Horne, M.A., Shimony, A., Zeilinger, A. "Bell's theorem without inequalities", Amer. J. of Physics 58, 12 (Dec. 1990), 1131-1143.

[4] Jemderson, H.V., Pukelsheim, F., and Searle, S.R. "On the history of the Kronecker product", Linear and Multilinear Algebra 14, 2 (Oct 1983), 113-120.

[5] Mermin, N.D. "Quantum mysteries revisited", Amer. J. of Physics 58, 8 (Aug. 1990), 731-734.

[6] Mermin, N.D. "What's wrong with these elements of reality?", Physics Today 43 (1990), 9-11.

[7] Monroe, C.R. and Wineland, D.J. "Quantum computing with ions", Scientific American (Aug. 2008), 64-79.

[8] Shor, P. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM J. Sci. Statist. Comput. 26 (1997), 1484.