A quantum computer can be simulated by applying rotations to a unit vector $u \in \mathbb{C}^{2^n}$ where $\mathbb{C}$ is the set of complex numbers and n is the number of qubits. The dimension is 2^n because a register with n qubits has 2^n eigenstates. Quantum operations are "rotations" because they preserve $|u| = 1$. Mathematically, a rotation of u is equivalent to the product Ru where R is a $2^n \times 2^n$ matrix.

The Eigenmath function $rotate(u, s, k, \ldots)$ rotates vector u and returns the result. Vector u is required to have 2^n elements where n is an integer from 1 to 15. Arguments $s, k, \ldots$ are a sequence of rotation codes where s is an upper case letter and k is a qubit number from 0 to $n-1$. Rotations are evaluated from left to right. The available rotation codes are

C, k	Control prefix
H, k	Hadamard
P, k, ϕ	Phase modifier (use $\phi = \frac{1}{4}\pi$ for T rotation)
Q, k	Quantum Fourier transform
V, k	Inverse quantum Fourier transform
W, k, j	Swap bits
X, k	Pauli X
Y, k	Pauli Y
Z, k	Pauli Z

Control prefix C, k modifies the next rotation code so that it is a controlled rotation with k as the control qubit. Use two or more prefixes to specify multiple control qubits. For example, C, k, C, j, X, m is a Toffoli rotation. Fourier rotations Q, k and V, k are applied to qubits 0 through k. (Q and V ignore any control prefix.)

Error codes

1 Argument u is not a vector or does not have 2^n elements where $n = 1, 2, \ldots, 15$.

2 Unexpected end of argument list (i.e., missing argument).

3 Bit number format error or range error.

4 Unknown rotation code.

Eigenstates $|j\rangle$ are represented by the following vectors. (Each vector has 2^n elements.)

$$
\begin{aligned}
|0\rangle &= (1, 0, 0, \ldots, 0) \\
|1\rangle &= (0, 1, 0, \ldots, 0) \\
|2\rangle &= (0, 0, 1, \ldots, 0) \\
&\vdots \\
|2^n - 1\rangle &= (0, 0, 0, \ldots, 1)
\end{aligned}
$$

A quantum computer algorithm is a sequence of rotations applied to the initial state $|0\rangle$. (The sequence could be combined into a single rotation by associativity of matrix multiplication.) Let

ψ_f be the final state of the quantum computer after all the rotations have been applied. Like any other state, ψ_f is a linear combination of eigenstates.

$$\psi_f = \sum_{j=0}^{2^n-1} c_j|j\rangle, \quad |\psi_f| = 1$$

The last step is to measure ψ_f and get a result. Measurement rotates ψ_f to an eigenstate $|j\rangle$. The measurement result is $|j\rangle$. The probability P_j of getting a specific result $|j\rangle$ is

$$P_j = |c_j|^2 = c_j c_j^*$$

Note that if ψ_f is already an eigenstate then no rotation occurs. (The probability of rotating to a different eigenstate is zero.) Since the measurement result is always an eigenstate, the coefficients c_j cannot be observed. However, the same calculation can be run multiple times to obtain a probability distribution of results. The probability distribution is an estimate of $|c_j|^2$ for each $|j\rangle$ in ψ_f.

Unlike a real quantum computer, in a simulation the final state ψ_f, or any other state, is available for inspection. Hence there is no need to simulate the measurement process. The probability distribution of the result can be computed directly.

Here are some useful Eigenmath code snippets for setting up a simulation and observing the result.

1. Initialize $\psi = |0\rangle$.

```
n = 4           -- number of qubits (example)
N = 2^n         -- number of eigenstates
psi = zero(N)
psi[1] = 1
```

2. Compute the probability distribution for state ψ.

```
P = psi conj(psi)
```

Hence

$$\begin{aligned}
\texttt{P[1]} &= \text{probability that } |0\rangle \text{ will be the result} \\
\texttt{P[2]} &= \text{probability that } |1\rangle \text{ will be the result} \\
\texttt{P[3]} &= \text{probability that } |2\rangle \text{ will be the result} \\
&\vdots \\
\texttt{P[N]} &= \text{probability that } |N-1\rangle \text{ will be the result}
\end{aligned}$$

3. Draw a probability distribution.

```
xrange = (0,N)
yrange = (0,1)
draw(P[ceiling(x)],x)
```

4. Compute an expectation value.

```
sum(k,1,N, (k - 1) P[k])
```

5. Make the high order qubit "don't care."

```
for(k,1,N/2, P[k] = P[k] + P[k + N/2])
```

Hence for $N = 16$

$$
\begin{aligned}
\texttt{P[1]} &= \text{probability that the result will be } |0\rangle \text{ or } |8\rangle \\
\texttt{P[2]} &= \text{probability that the result will be } |1\rangle \text{ or } |9\rangle \\
\texttt{P[3]} &= \text{probability that the result will be } |2\rangle \text{ or } |10\rangle \\
&\vdots \\
\texttt{P[8]} &= \text{probability that the result will be } |7\rangle \text{ or } |15\rangle
\end{aligned}
$$

Example.

```
-- Verify the following truth table for cnot

-- Input Output
--    00     00
--    01     11
--    10     10
--    11     01

U(psi) = rotate(psi,C,0,X,1) -- cnot

ket00 = (1,0,0,0)
ket01 = (0,1,0,0)
ket10 = (0,0,1,0)
ket11 = (0,0,0,1)

U(ket00) == ket00
U(ket01) == ket11
U(ket10) == ket10
U(ket11) == ket01
```