Assignment 3

Due by class time Tuesday, September 20

Note: in the exercises below, natural (base e) logarithms are written as $\log(x)$, whereas logarithms using a different base b are written as $\log_b(x)$.

1. Calculate the following natural logarithms. Give *three* distinct answers for each, including a real-valued answer if one exists. Express your answers in Cartesian form, and show your work.

   (a) $\log(e^5)$

   (b) $\log(-e)$

   (c) $\log(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\,i)$

2. Calculate the common (base 10) logarithm of -100. Show your work. Hint: $\log_b(x) = \log(x)/\log(b)$ for any base b.

3. We can even use negative or imaginary bases for logarithms. Perhaps we could call base -10 logarithms "uncommon", base $-e$ logarithms "unnatural", and base $-i$ logarithms "unreal".

   (a) Calculate the unnatural logarithm $\log_{-e}(e)$. Show your work.

   (b) Calculate the uncommon logarithm $\log_{-10}(-100)$. Show your work.

   (c) Calculate the unreal logarithm $\log_{-i}(-1)$. Show your work.

4. Calculate the two square roots of i, and express them in both exponential and Cartesian form, showing your work. Draw a picture of i and its roots as vectors (arrows) in the complex plane, as accurately as you can. Hint: first rewrite i in exponential form.

5. Figure 1.6 on page 19 of our textbook *Quantum Computing for Computer Scientists* isn't quite right. Briefly describe why the figure is inaccurate, and explain how to fix it.

6. Figure 1.10 on page 24 is also not right. Briefly describe why the figure is inaccurate, being as specific as you can, and draw the figure as it should appear.

7. Calculate the sixth-roots of unity (that is, $\sqrt[6]{1}$), expressing each root in exponential form (Cartesian form is not required), and draw them as vectors (arrows) in the complex plane, as accurately as you can.

Extra Credit (optional)

8. Write a Python program to compute the phase θ of an arbitrary complex number $a + bi$. Your program should accept a and b as input parameters, and return the phase as an angle in radians from 0 to 2π. You may assume that a and b will never both be 0, but one or the other might be. Your program will need to check for several special cases to ensure that the result is in the range $0 \leq \theta < 2\pi$, and to avoid division by zero. Note: you are not allowed to use Python's `atan2` function for this exercise; but you may use standard `atan`. Test your program thoroughly on *at least* the following examples:

```
phase(1, 0)  -> 0.0              phase(-1, 0)  -> 3.141592
phase(2, 2)  -> 0.785398         phase(-2, -2) -> 3.926990
phase(0, 1)  -> 1.570796         phase(0, -1)  -> 4.712388
phase(-2, 2) -> 2.356194         phase(2, -2)  -> 5.497787
```