```
> k = ket(1,0)
> print(k)
     [,1]
[1,] 1+0i
[2,] 0+0i
> dirac(k)
[1] "1|0>"
> k = ket(1,1)
> print(probs(k))
     [,1]
[1,]  0.5
[2,]  0.5
>
```

1.

```
> X(ket(1,0))
     [,1]
[1,] 0+0i
[2,] 1+0i
> dirac(X(ket(1,0)))
[1] "1|1>"
> probs(X(ket(1,0)))
     [,1]
[1,]    0
[2,]    1
>
```

2.

```
> X(ket(1,1))
                [,1]
[1,] 0.7071068+0i
[2,] 0.7071068+0i
> dirac(X(ket(1,1)))
[1] "0.707|0> + 0.707|1>"
> probs(X(ket(1,1)))
     [,1]
[1,]  0.5
[2,]  0.5
```

3.
    a. The directions said to, "Apply an X gate to the qbit" and I didn't know if you meant the |0> qbit or the |0> + |1> so I did it for both and posted them on 2 and 3. In both cases these outputs make sense because of their direct derivations defined in class. The X gate command in both cases output the correct values and the probabilities are very clearly correct as the probability for a qbit to be either |0> or |1> is 50% while the chance for a qbit that has the value of |1> to be |1> is 100%.

```
> k = ket(sqrt(1/3), sqrt(2/3))
> X(k)
             [,1]
[1,] 0.8164966+0i
[2,] 0.5773503+0i
> dirac(X(k))
[1] "0.816|0> + 0.577|1>"
> probs(X(k))
          [,1]
[1,] 0.6666667
[2,] 0.3333333
> probs(k)
          [,1]
[1,] 0.3333333
[2,] 0.6666667
```

4.
   a. As we can see here, the probabilities for the qbit before the gate is ⅓,⅔ which makes sense because of the values we inputted. Then after we put the qbit into the gate, we see that the probability has swapped from ⅓,⅔ to ⅔, ⅓. When putting a qbit through the X gate, it is similar to swapping the qbits, so swapping the probabilities make sense and the values we obtained are accurate to the values we would get by manually doing the calculation as taught in class.