Hw1 Q3.
Spencer Tran
CSCI 104

a)

x	1	2	3	4
n	2	2^2	2^4	2^8

For the function to loop *x* times, n would need to be at least 2^(2^*x*)
This is because n needs to be greater than i, which multiplies with itself.
So, in order to run another time, the requirement for n also multiplies with itself:

$$2^{2^{(x+1)}} = 2^{2(2^x)} = 2^{2^x+2^x} = \left(2^{2^x}\right)\left(2^{2^x}\right)$$

Let us solve for x in terms of n:

$$n = 2^{2^x}$$
$$\log(n) = 2^x(\log 2)$$
$$\log(\log(n)) = x\log(2) + \log(\log(2))$$
$$x = \log(\log(n))$$

This means that a function with an input of n, would run the O(1) task *log(log(n))* times.
<u>The function is Θ(log(log(n))</u>

b) The if statement checks if i is a multiple of √n. It will return true √n times because there are √*n* multiples of √n from 1 to n. The first time it returns true, i will be the first multiple, √n. On the second time, i will be the second multiple, 2(√n). On the third time, i will be the third multiple, 3(√n), and so on. It goes on until i becomes the (√n)-th multiple, which is ((√n)(√n)) or n. For each time, the if statement's condition is true, the function will run an O(1) task for i^3 times.
So the total amount of times this task runs for is the summation of all √n multiples from 1 to n, cubed:

$$(1\sqrt{n})^3 + (2\sqrt{n})^3 + (3\sqrt{n})^3 + \dots + (\sqrt{n}\sqrt{n})^3$$

$$= \sqrt{n}^3 (1^3 + 2^3 + \dots + \sqrt{n}^3)$$

The sum of $(1^3 + 2^3 + \dots + \sqrt{n}^3)$ is $\left(\frac{\sqrt{n}(\sqrt{n}+1)}{2}\right)^2$ and approaches $(\sqrt{n})^4$ as n approaches ∞.

$$f2(n) = \Theta((\sqrt{n}^3)(\sqrt{n}^4))$$
$$= \Theta(n^{(\frac{7}{2})})$$

Because the if statement runs for n times, the runtime is n+n^(7/2). The function approaches n^(7/2) as n goes toward infinity, so the n is ignored.
Therefore, <u>the function is Θ(n^(7/2))</u>

c) The if statement is inside 2 for loops that both run n times for Θ(1). So, it contributes a runtime of Θ(n^2).
Now let us analyze the run time inside each if statement. The if statement's condition can be true at most n times because the contents of the array do not change. For every time the condition is true, the value of k is unique. Because there are only n unique values of k, the if statement's condition can be true at most n times.
The for loop that runs below the if statement will run for log(n) times. It runs for however many times m can double before exceeding n. So, m would double (log(n)) times before exceeding n.
So, the if statement's condition is true at most n times and inside the if statement, a for loop runs the O(1) task for log(n) times. The function is Θ(n*(log(n) + n^2). The n*log(n) is ignored because it grows slower than n^2, so the function approaches n^2 as n goes toward infinity. Therefore, <u>the function is Θ(n^2)</u>

d) Since the "a[i] = i*i;" statement is in the for loop and runs regardless of the if statement's condition, it will run at least n times. The if statement is Θ(1) and runs n times, so we can just combine its runtime with the "a[i] = i*i;" statement and calculate both as Θ(n). We must also consider the amount of times the “b[j] = a[j];” statement runs whenever the array needs to resize:

$$f(n) = \Theta(n + (\text{resizing time}))$$

Whenever i becomes the size of the array, it must run the assignment statement for “size” times. The summation for all resize statements is given by the equation below. Let us assume the log(n)’s without a base are in base 3/2.
The array resizes for a log(n) amount of times because it is equal to however many times the size can be multiplied by 3/2 before surpassing n.

Now, we find the sum of the geometric series, ($(3/2)^0 + (3/2)^1 + (3/2)^2 \ldots$):

$$1 + \left(\frac{3}{2}\right) + \left(\frac{3}{2}\right)^2 + \ldots + \left(\frac{3}{2}\right)^{\log(n)} = \text{sum}$$

$$\left(\frac{3}{2}\right) + \left(\frac{3}{2}\right)^2 \ldots \left(\frac{3}{2}\right)^{\log(n)+1} = \left(\frac{3}{2}\right)\text{sum}$$

$$(\text{sum} - 1) + \left(\frac{3}{2}\right)^{\log(n)+1} = \left(\frac{3}{2}\right)\text{sum}$$

$$-1 + \left(\frac{3}{2}\right)^{\log(n)+1} = \left(\frac{3}{2}\right)\text{sum} - \text{sum}$$

$$-1 + \left(\frac{3}{\cancel{2}}\right)^{\log(n)+1} = \left(\frac{1}{\cancel{2}}\right)\text{sum}$$

$$-2 + 3^{\log(n)+1} = \text{sum}$$

We can use this to find the resize time:

$$\text{resizing time} = 10\left(\tfrac{3}{2}\right)^0 + 10\left(\tfrac{3}{2}\right)^1 + \ldots + 10\left(\tfrac{3}{2}\right)^{\log(n)}$$

$$= 10\left(1 + \left(\tfrac{3}{2}\right) + \ldots \left(\tfrac{3}{2}\right)^{\log(n)}\right)$$

$$= 10\left(-2 + 3^{\log_{\frac{3}{2}}(n)+1}\right)$$

$$= 10\left(-2 + 3^{\frac{\log_3(n)}{\log_3\left(\frac{3}{2}\right)}+1}\right)$$

$$= 10\left(-2 + 3^{\log_3\left(n - \frac{3}{2}\right)+1}\right)$$

$$= 10\left(-2 + 3\left(n - \tfrac{5}{2}\right)\right)$$

$$\text{resize time} = 30n - 65$$

Now, we go back to the original equation and find the runtime:

$$f(n) = \Theta(n + 30n - 65)$$

$$= \theta(31n - 65)$$

$$= \Theta(n)$$

Therefore, the function is Θ(n).