

Part (a)

```
void f1(int n)
{
    int i=2;
    while(i < n){
        /* do something that takes O(1) time */
        i = i*i;
    }
}
```

a)

0	2	3	4	15	16	17	256	3
1								
k	0	1	1	2	2	3	3	4

we can see that the common formula is $\log \log n$

Part (b)

```
void f2(int n)
{
    for(int i=1; i <= n; i++){
        if( (i % (int)sqrt(n)) == 0){
            for(int k=0; k < pow(i,3); k++) {
                /* do something that takes O(1) time */
            }
        }
    }
}
```

b)

$$= \sum_{i=1}^n \left(\theta(1) + \sum_{k=0}^{i^3} \theta(1) \right)$$

$$= \sum_{i=1}^n \theta(1) + \sum_{i=1}^{\sqrt{n}} \sum_{k=0}^{i^3-1} \theta(1)$$

n	16	i = 1
<hr/>		
	2	
	3	
	4	0
	6	
	7	
	8	0
	9	
	10	
	12	0
	13	
	14	
	15	
	16	0

$$\pm \sqrt{n} = ?$$

$$= \sum_{i=1}^n \theta(1) + \sum_{k=1}^{\sqrt{n}} \theta(i^3)$$

$$= n + \sum_{k=1}^{\sqrt{n}} (k\sqrt{n})^3$$

$$= n + (\sqrt{n})^3 \sum_{k=1}^{\sqrt{n}} k^3$$

$$= n + n^{3/2} \left(\frac{n}{4} (\sqrt{n}+1)^2 \right)$$

$$= n + n^{3/2} (n^2)$$

$$= n + n^{7/2}$$

$$= \theta(n^{7/2})$$

Part (c)

```

for(int i=1; i <= n; i++){
    for(int k=1; k <= n; k++){
        if( A[k] == i){
            for(int m=1; m <= n; m=m+m){
                // do something that takes O(1) time
                // Assume the contents of the A[] array are not changed
            }
        }
    }
}

```

c)

$$\sum_{i=1}^n \sum_{k=1}^n \left(\theta(1) + \sum_{m=1}^{\log n} \theta(1) \right)$$

$$= \sum_{i=1}^n \left(\sum_{k=1}^n \theta(1) \right) + \sum_{k=1}^n \sum_{m=1}^{\log n} \theta(1)$$

$$= \sum_{i=1}^n (n) + \sum_{k=1}^n \theta(\log n)$$

$$= \theta(n^2) + \theta(n \log n)$$

$$= \theta(n^2) + \theta(n \log n)$$

$$= \theta(n^2)$$

Part (d)

Notice that this code is very similar to what will happen if you keep inserting into an ArrayList (e.g. vector). Notice that this is NOT an example of amortized analysis because you are only analyzing 1 call to the function $f()$. If you have discussed amortized analysis, realize that does NOT apply here since amortized analysis applies to multiple calls to a function. But you may use similar ideas/approaches as amortized analysis to analyze this runtime. If you have NOT discussed amortized analysis, simply ignore it's mention.

```
int f (int n)
{
    int *a = new int [10];
    int size = 10;
    for (int i = 0; i < n; i++)
    {
        if (i == size)
        {
            int newsize = 3*size/2;
            int *b = new int [newsize];
            for (int j = 0; j < size; j++) b[j] = a[j];
            delete [] a;
            a = b;
            size = newsize;
        }
        a[i] = i*i;
    }
}
```

i | 10 15 22 33

general form $i = 10 \left(\frac{3}{2}\right)^t$

$i = n$

$$10 \left(\frac{3}{2}\right)^t = n$$

$$\left(\frac{3}{2}\right)^t = \frac{n}{10}$$

$$t = \log_{3/2} \left(\frac{n}{10}\right)$$

$$\log_{3/2} \left(\frac{n}{10}\right)$$

$$\sum_{t=0}^{\log_{3/2} \left(\frac{n}{10}\right)} \sum_{j=0}^{\text{size}}$$

$$0 = 10 \left(\frac{3}{2}\right)^0 + 10 \left(\frac{3}{2}\right)^1 + 10 \left(\frac{3}{2}\right)^2 + \dots + 10 \left(\frac{3}{2}\right)^t$$

$$\sum_{t=0}^{\log_{3/2} \left(\frac{n}{10}\right)} \left(10 \left(\frac{3}{2}\right)^t\right) =$$

$$10 \sum_{t=0}^{\log_{3/2} \left(\frac{n}{10}\right)} \left(\frac{3}{2}\right)^t$$

$n = 17$ 0 1 new

1

2

3

4

5

6

7

8

9

10

$\frac{3^* \text{size}}{2}$

11

12

13

14

15

$\frac{3^* 15}{2}$

$$\text{let } \log_{\frac{3}{2}}\left(\frac{n}{10}\right) = m$$

$$10 \sum_{t=0}^m \left(\frac{3}{2}\right)^t = 10 \left(\frac{\left(\frac{3}{2}\right)^{m+1} - 1}{\frac{1}{2}} \right)$$

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

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

$$= 10 \left(\frac{n}{10} \right)$$

$$= \underline{\underline{f(n)}}$$