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

$T(n) = \theta(1) + O\left(\sum^{n} \theta(1)\right)$

$= \log_2(\log_2 n)$

$n = 36$

$i=2$

$i = 2 \cdot 2 = 4 \qquad 2^2$

$i = 4 \cdot 4 = 16 \qquad 2^4$

$i = 16 \cdot 16 \; 256 \qquad 2^8$

1000

$\sum_{i=1}^{n} \left(2^{2^i}\right)$

$2^{2^i} = n$

$\log_2 n = 2^i$

$\log_2(\log_2 n) = i$

$\Rightarrow \sum_{k=1}^{\log_2(\log_2 n)} k$

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 */
            }
        }
    }
}
```

$\frac{n}{\sqrt{n}} = \sqrt{n}$

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

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

$= \theta(n) + \sqrt{n}\; \theta(n^3) = n^{\frac{7}{3}}$

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
      }
    }
  }
}
```

k = 1 2 3 4

| 1 | 2 | 3 | 4 |

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

$$\sum_{i=1}^{n}\sum_{k=1}^{n}\left(\theta(1) + O(\log_2 n+1)\right)$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{n}\theta(1) + \sum_{i=1}^{n}\sum_{k=1}^{n} O(\log_2 n+1)$$

max data set to pass if statement is n times

$$= \sum_{i=1}^{n}\theta(n) + n\log_2 n + n$$

$$= n^2 + n\log_2 n + n = n^2$$

$m=1$ $\quad n=50$

$m=1+1=2$

$m=2+2=4$

$m=4+4=8$

$m=8+8=16$

$m=16+16=32$

$m=32+32=64$

$$\sum_{m=1}^{n} 2^{m-1} \rightarrow \log_2 n = m-1$$

$$\log_2 n + 1 = m$$

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;
     }
}
```

k	1	2	3	4	5	6	7	8	9
size	10	15	22	33	49	73	109	163	244

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

$$=\theta(2)+\sum_{i=1}^{n}\theta(6)+\sum_{i=1}^{n}\sum_{j}\theta(1)$$

$$=\theta(6n)+\sum_{i}^{\log_{\frac{3}{2}}\left(\frac{n}{10}\right)}\sum_{j}^{i}\theta(1)$$

$$=\theta(n)+\sum_{i=0}^{\log_{\frac{3}{2}}\left(\frac{n}{10}\right)}\theta(i)$$

$$=\theta(n)+10\,\theta\left(\frac{3}{2}\right)^{k}$$

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

$$=\theta(n)$$

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

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

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

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

```
struct Node {
    int val;
    Node*  next;
};

Node* llrec(Node* in1, Node* in2)
{
    if(in1 == nullptr) {
        return in2;
    }
    else if(in2 == nullptr) {
        return in1;
    }
    else {
        in1->next = llrec(in2, in1->next);
        return in1;
    }
}
```

Question a: What linked list is returned if llrec is called with the input linked lists in1 = 1,2,3,4 and in2 = 5,6?

Question b: What linked list is return if llrec is called with the input linked lists in1 = nullptr and in2 = 2?

(a) in1 $\boxed{1}$ $\boxed{2}$ $\boxed{3}$ $\boxed{4}$

in2 $\boxed{5}$ $\boxed{6}$

(1, 5) → (5, 2) → (2, 6) → (6, 3) → (3, null)
→ return in1 = 3

(b) (null, 2) → return in 2 = 2