Part a)

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

i^2

k	i	2^{2^k}
1	2	2^{2^1}
2	4	$= 2^{2^2}$
3	16	$= 2^{2^3}$
4	256	$= 2^{2^4}$
5	65,536	$= 2^{2^5}$

for(int i=2; i<n; i=i^2)

$$\sum_{i=2} \Theta(1)$$

$$n = 2^{2^k}$$

where k = number of iterations

$$\log_2 n = 2^k$$

$$\log_2(\log_2 n) = k$$

$$\boxed{\Theta\left(\log_2(\log_2 n)\right)}$$

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 */
            }
        }
    }
}
```

int part

$\frac{j}{\sqrt{n}} \leq 1 \quad j \leq \sqrt{n}$

i: 1 2 3 4 5 j

i % √n

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

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

$m = \sqrt{n}$

$$\theta(n) + \sum_{j=1}^{m} i^3$$

$$\theta(n) + \theta(m^4)$$

$$\theta(n) + \theta((\sqrt{n})^4)$$

$$\theta(n) + \theta(n^2) = \boxed{\theta(n^2)}$$

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
      }
    }
  }
}
```

1, 2, 4, 8,

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

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

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

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

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

$$\boxed{\theta(n^2\log n)}$$

Part d)

```
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;
     }
}
```

Outer loop happens n times

each time i reaches size

array is resized which take O(size) time. size is changed to 1.5 * size

sizes will be rounded

10, 15, 22, 33, 50

summation is

$$\sum_{K=1}^{\log_{3/2}(n/10)} \theta(1) = \theta(\log n)$$

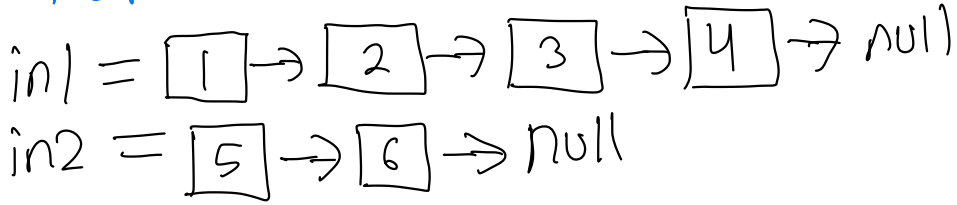
$$\theta(n) + \theta(\log n) = \boxed{\theta(n)}$$

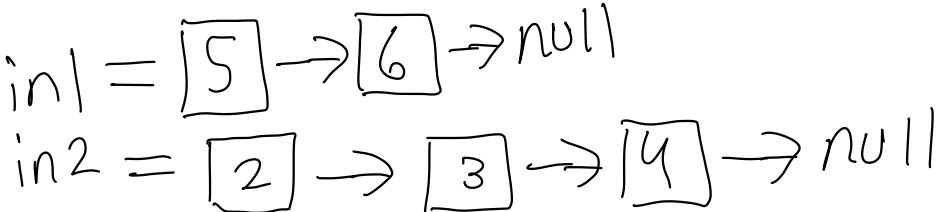
outer inner

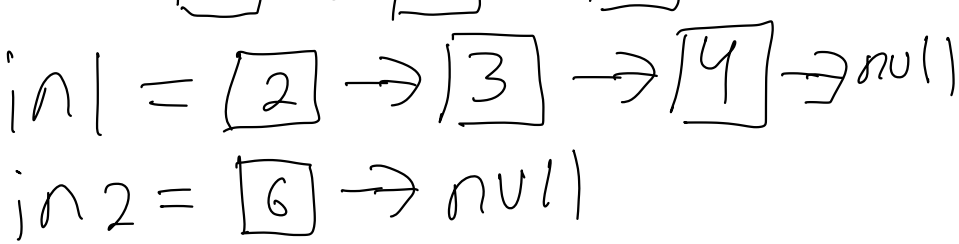
```
struct Node {
    int val;
    Node*  next;
};

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;
    }
}
```

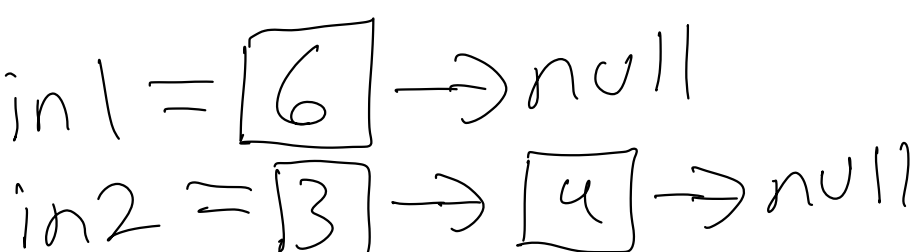
(next page for results)

a) Function calls

1st


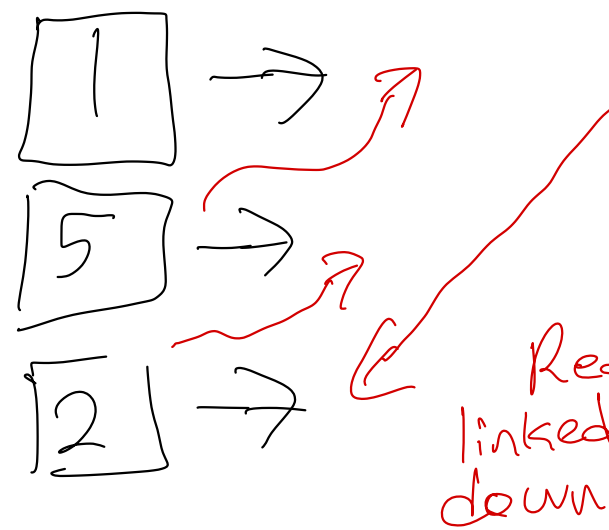
2nd


3rd


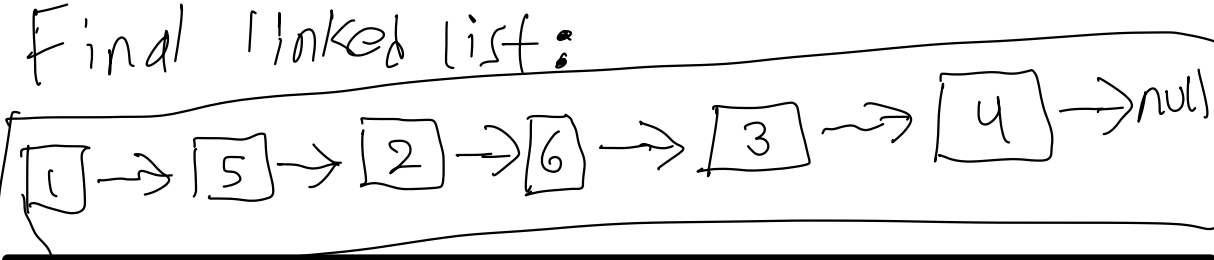
4th


5th


Base case reached



Final linked list:



b) 1st function calls

in1 = null

in2 = 2 → null

returns in2

so 2 → null

Since in1 = null Base case is reached right away and final linked list is