



Wiki

work will receive 0 credit.

### Part (a)

```
void f1(int n)
{
    int i=2;
    while (i < n)
    {
        i = i*i;
    }
}
```

Handwritten notes:

- Annotations for the while loop:  $2^2 \rightarrow 4$ ,  $2^{2^2} \rightarrow 5$ ,  $2^{2^2} \rightarrow 5$
- A box containing the time complexity:  $\Theta(\log(\log(n)))$
- Additional notes:  $2^{2^2} \rightarrow 5$ ,  $2^{2^2}$

### Part (b)

[illegible]

### Part (c)

[illegible]

### Part (d)



USC University of  
Southern California



GitHub



Ask on  
Ed

### 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;
    }
}

```

Handwritten analysis notes:

- $n < 10 \rightarrow O(n)$
- $n=11 \rightarrow$  runs 10 times to fill - new size - new b
- runs 15 times
- $n=16 \rightarrow$  does the 15 times at  $n=15 \rightarrow$  does  $15/2$
- $3.5/2$
- Creates array of new size
- puts all values of
- will run  $n$  times
- $O(n^2)$
- $a[i] = i*i;$
- $15 \rightarrow 22 \rightarrow 33 \rightarrow 44$
- $10 \times (3/2)^x = 11 \rightarrow \log_{3/2}(11/10) = x$
- $n \times n = O(n^2)$
- $10 \times \frac{1 - (3/2)^{-10}}{1 - 3/2} \rightarrow 10 \times \frac{1 - (2/3)^{10}}{1/2} \rightarrow 20 \times (1 - 1024/59049) \rightarrow 20 \times \frac{48825}{59049} \rightarrow \frac{1676000}{59049} \rightarrow 28.38$
- inner loop runs  $n$  times

### Problem 4 - Linked List Recursion Tracing (13%)

Consider the following C++ code. For the following questions, all of the points for this problem will be assigned based on your explanation, since we have full faith in your ability to run this program and copy down the answer.

To show work, you can draw a call tree or box diagram of the