2. **Find positive constants c_0 and c_1 such that, for all positive integers n,** $c_0 n^3 < 27n^3 + 13n^2 + 873(\lg n)^3 < c_1 n^3$. Justify briefly.

   For c_0, divide the inequality by n^3, $27 + 13/n + 873(\lg n)^3/n^3 > c_0$, when $n \to \infty, 27 + 13/n + 873(\lg n)^3/n^3 \to 27$. So, $c_0 \in (0, 27]$.
   For c_1, $27n^3 + 13n^2 + 873(\lg n)^3 < 27n^3 + 13n^3 + 873n^3 < (27 + 13 + 873)\, n^3 = 913n^3$. So, $c_1 \in [913, \infty)$.

3. Define $\alpha = (1 + \sqrt{5})/2$. Define $f(0) = 0$, $f(1) = 1$, and $f(n) = f(n-1) + f(n-2)$ for all $n \geq 2$. Define $T(0) = T(1) = 4$ and $T(n) = T(n-1) + T(n-2) + 7$ for all $n \geq 2$.

   (i) **Prove, for all integers $n \geq 3$, $f(n) > \alpha^{n-2}$.**

   Let P(n) be the above predicate, i.e. $f(n) > \alpha^{n-2}$.Prove the claim by induction on n Base case. $Letn = 3$. Then $f(3) = f(2)+f(1) = f(1)+f(0)+f(1) = 2 = (1+\sqrt{9})/2 > (1+\sqrt{5})/2 = \alpha^{n-2}$. So P(3).
   Inductive case. Let k be any integer $\geq$ 3. Assume P(n) for all integers n in the set $\{1, 2, 3, ..., k-1\}$. Then, we want to show P(k), i.e. $f(k) > \alpha^{k-2}$.
   Now, f(k) = f(k-1) + f(k-2) (by the definition of f(k), since $k \geq 3$), and P(k-1) and P(k-2) by the inductive assumption, i.e. $f(k-1) > \alpha^{k-3}$ and $f(k-2) > \alpha^{k-4}$, so we are done if $\alpha^{k-3} + \alpha^{k-4} \geq \alpha^{k-2}$.
   Since $\alpha + 1 = (3 + \sqrt{5})/2 = \alpha^2$, $\alpha^{k-3} + \alpha^{k-4} = \alpha^{k-4}(\alpha + 1) = \alpha^{k-4}\alpha^2 = \alpha^{k-2}$, it follows that $f(k) > \alpha^{k-2}$, so P(k). So, by the principal of mathematical induction, P(n) holds for all integers $n \geq 3$.

   (ii) **Prove, for all integers $n \geq 1$, $T(n) < 18f(n)$.** Hint. use some results from the seminar (see version revised today).

   Let Q(n) be the above predicate, i.e. $T(n) < 18f(n)$.
   $T(1) = 4 < 18f(1) = 18$. So Q(1).
   $T(2) = T(1) + T(0) + 7 = 11 < 18f(2) = 18(f(1) + f(0)) = 18$. So Q(2).
   Now, $T(n) = 11f(n+1) - 7$ (from the seminar, since $n \geq 3$), and $f(n) < \alpha^n$ (from the seminar, since $n \geq 0$).
   So, for $n \geq 3$, $T(k) = 11f(k+1) - 7 < 11\alpha^{k+1} - 7 = (11 + 11\sqrt{5})\alpha^k/2 - 7 < 18\alpha^k$. So $Q(k)$, since$k \geq 3$. So $T(n) = 11f(n+1) - 7$ holds for all integers $n \geq 1$.

4. (i) Show the output from the call `ff(4)`.

   L = [1, 6, 7, 13, 20], so the value returned is 20.

```
def ff(n):
  L = [1, 6]
  for j in range(2,n+1):    # j ranges from 2 to n
    L.append( L[j-2]+L[j-1] )
    # (*) invariant:  j is the index of the last element of L
  print L[n]
```

   (ii) Finish the proof of the claim.

   Claim: each time execution reaches `(*)`, the invariant holds.

   Proof. By induction on the variable `j`.

Base case. In Python, list indices start at 0, so the first time execution reaches the for loop, L has its initial 2 elements, so the first time execution reaches line `(*)`, L has had exactly 1 element appended (its value is `1+6=7`), so L has exactly 3 elements, so the index of the last element is 2. Also, the first time execution reaches `(*)`, `j` is 2. So the invariant holds when `j` is 2.

Inductive case. Let t be any integer ≥ 2. Assume that the invariant holds when execution reaches line `(*)` and `j=t`. We want to show that that the invariant then holds when execution reaches line `(*)` and `j=t+1`.

So, assume execution reaches line `(*)` with `j=t+1`. Now since we assume that the invariant holds when execution reaches line `(*)` and `j=t`, then before the iteration starts with $j = t+1$, t is the last index of L, which means the length of L is $t+1$. When the execution reaches line `(*)` with `j=t+1`, L has another element appended (its value is L[t-1]+L[t]). Then the length of L by now should be $t+2$, which means the last index of L would be $t+1$. So the invariant holds when $j = t+1$.

5. (i) Trace the execution of the algorithm below with input `x=29` and `y=11`.

   (ii) Give the runtime as a function of k, assuming x and y each have k bits.

```
def mr(x,y): #
  if (x==0):
    return 0
  z = mr(x/2,y)
  if (1==x%2):
    return z + z + y
  return z + z
```

Solution:
(i) We trace the execution of the algorithm as follows:
mr(29,11): x≠0 ⇒ z=mr(14,11)
mr(14,11): x≠0 ⇒ z=mr(7,11)
mr(7,11): x≠0 ⇒ z=mr(3,11)
mr(3,11): x≠0 ⇒ z=mr(1,11)
mr(1,11): x≠0 ⇒ z=mr(0,11)
mr(0,11): x=0 ⇒ Returned value = 0
mr(1,11): z=0 and x%2=1 ⇒ returned value = 0+0+11 = 11
mr(3,11): z=11 and x%2=1 ⇒ returned value =11+11+11 = 33
mr(7,11): z=33 and x%2=1 ⇒ returned value =33+33+11 = 77
mr(14,11): z=77 and x%2=0 ⇒ returned value =77+77 = 154
mr(29,11): z=154 and x%2=1 ⇒ returned value =154+154+11 = 319

(ii) With each recursive call, the number of bits of x is reduced by exactly 1, so the total number of recursive calls is in $\Theta(k)$. Within one call, there are a constant number of operations, including if-check, comparison with 0, return, integer division by 2 (and remainder), and addition. Each of these operations takes $O(k)$ time, and division by 2 with remainder takes $\Theta(k)$ time, so the time for each recursive call (except possible the last) takes $\Theta(k)$ time. So the runtime is in $\Theta(k^2)$.