Prove that $o(g(n)) \cap \omega(g(n))$ is the empty set. . Proof by Contradiction . I(n) \in $dg(n) \cap \omega(g(n))$. Then, for every positive constant (1, there exists a constant no such that . For every positive constant (2, thereexists a constant no such that . Colog(n)) \leq E(n) for all nZno . If $C_1 = 1$ and $C_2 = 5$, then $\frac{1}{2}(g(n)) \leq \frac{1}{2}(n) \leq \frac{1}{2}(n)$. This is impossible.

2. SPARC to Python

Consider the following SPARC code:

$$\begin{array}{ll} foo\ x=\\ &\mbox{if}\ x\leq 1\ \mbox{then}\\ &x\\ &\mbox{else}\\ &\mbox{let}\ (ra,rb)=(foo\ (x-1))\ ,\ (foo\ (x-2))\ \mbox{in}\\ &\ ra+rb\\ &\mbox{end}. \end{array}$$

- · 2a. Translate this to Python code fill in the def foo method in main.py
- · 2b. What does this function do, in your own words?

This function determines the Xth term in the Fibonacci sympence.

3. Parallelism and recursion

Consider the following function:

def longest_run(myarray, key)

Input:

'myarray': a list of ints

'key': an int

Return:

the longest continuous sequence of `key` in `myarray`

E.g., $longest_run([2,12,12,8,12,12,12,0,12,1], 12) == 3$

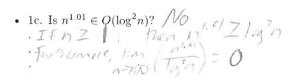
- · 3a. First, implement an iterative, sequential version of longest_run in main.py.
- 3b. What is the Work and Span of this implementation?

Spani O(n)

CMPS 2200 Assignment 1

:2" 5 n? 16.

CIVIL & 2200 Assignment 1	
Name: Naomi Stoner	
In this assignment, you will learn more about asymptotic notation, parallelism, functional langualgorithmic cost models. As in the recitation, some of your answer will go here and some will go in You are welcome to edit this assignment-01.md file directly, or print and fill in by hand. If you do please scan to a file assignment-01.pdf and push to your github repository.	n main.py
1. Asymptotic notation	
• 1a. Is $2^{n+1} \in O(2^n)$? Why or why not? $\cdot \checkmark \in S$ • If $C = 3$ and $P_0 = -1$, • $2^{n+1} \le 3 \cdot 2^n$	
1b. In $92^n \in O(9n)2$. When an unbrunet?	
• 1b. Is $2^{2^n} \in O(2^n)$? Why or why not? N_0 $ (2^n) = (2^n) \text{for } N_0 = 0$	



- 1d. Is $n^{1.01} \in \Omega(\log^2 n)$? Yes

 $n^{1.01} Z (\cdot \log^2 n)$ for C = 1• for C = 1• $\log^2 n$ and n = 1• $\log^2 n$ $\log^2 n$ •
- 1e. Is $\sqrt{n} \in O((\log n)^3)$? Yes • for C = 12 and $N_0 = 2$ • $\sqrt{n} \le 124(\alpha_0 n)^3$
- If. Is $\sqrt{n} \in \Omega((\log n)^3)$? N_0 There is no value for which C. Vin asymptotically deminates $((a_3h)^3)$
- 1g. Consider the definition of "Little o" notation:

 $g(n) \in o(f(n))$ means that for **every** positive constant c, there exists a constant n_0 such that $g(n) \le c \cdot f(n)$ for all $n \ge n_0$. There is an analogous definition for "little omega" $\omega(f(n))$. The distinction between o(f(n)) and O(f(n)) is that the former requires the condition to be met for **every** c, not just for some c. For example, $10x \in o(x^2)$, but $10x^2 \notin o(x^2)$.

- 3c. Next, implement a longest_run_recursive, a recursive, divide and conquer implementation. This
 is analogous to our implementation of sum_list_recursive. To do so, you will need to think about
 how to combine partial solutions from each recursive call. Make use of the provided class Result.
- 3d. What is the Work and Span of this sequential algorithm?

(J(n) 15 equal to the number of nodes, as each node has
(di) work.
Therefore, w(n) = nlogn, as the work; Sbalanced
S(n) = 2logz(n) & (logzh)

• 3e. Assume that we parallelize in a similar way we did with sum_list_recursive. That is, each recursive call spawns a new thread. What is the Work and Span of this algorithm?

longest-helper complets in a(1) time, So: W(n)= Zn E (Cn)

S(n) = Z(yz(n) & O(logzn)