1. Show that \( n! = o(n^n) \) and \( \log(n!) = \Theta(n \log n) \). Is \( [\log n]! \) a polynomially bounded function?

2. Prove that for any real constants \( a \) and \( b \), where \( b > 0 \), holds \( (n+a)^b = \Theta(n^b) \).

3. What can you say about \( 2^{\Theta(\sqrt{\log n})} \)? (This class of functions appears quite often in the analysis of algorithms. See e.g. “More Applications of the Polynomial Method to Algorithm Design” by Abboud et al.)

4. Prove or disprove:
   
   (a) \( f(n) + g(n) = \Theta(\min(f(n), g(n))) \)
   
   (b) \( f(n) + o(f(n)) = \Theta(f(n)) \)
   
   (c) \( \Theta(f(n)) + O(f(n)) = \Theta(f(n)) \)

5. Explain how to implement two stacks using one array. Operations \texttt{Pop} and \texttt{Push} must take \( O(1) \) time. If the number of elements in the stacks exceeds the size of the array, you can return overflow.

6. Implement a stack using a linked list.

7. Develop two implementations of a stack using two queues. Analyse their performance.

8. Develop a (non-recursive) procedure with running time \( \Theta(n) \) that reverses the order of elements in a linked list. The procedure must use \( O(1) \) extra space.

9. Given a singly linked list of characters, your task is to decide whether the list is a palindrome. Design (a) a linear-space and linear-time algorithm; (b) a constant-space and linear-time algorithm.

10. A sequence of stack operations is performed on a stack whose size never exceeds \( k \). After every \( k \) operations, a copy of the entire stack is made for backup purposes. Show that the cost of \( n \) stack operations, including copying the stack, is \( O(n) \).

11. Let \( A \) be a binary array of length \( n \). Develop an algorithm that finds the longest segment of \( S \) such that the sum of elements in it equals exactly \( k \).

12. Describe how two stacks can be used to implement a queue in such a way that the amortized time per queue operation is \( O(1) \).