my timeline: take notes, write down thoughts: 15 min
write code: 1 hr 30 min
tests/final thoughts: 15 min

Assumptions

→ Assume: there will be an error if someone inputs invalid inventory, products needed, or set of product chains dictionary
→ for product chains, there are multiple possible product replacements and it doesn't matter how we break the tie.
→ products needed will be input as a dictionary
→ how do we optimize for fulfillment? I will use the product with highest inventory levels instead of product "most similar"
→ each product ID is unique
→ there will always be enough product (never run out)

Thoughts for Implementation

→ lots of dictionaries

→ Inventory: maps id integer to quantity in stock will use dictionary

→ products needed: write in assumptions this will be inputted as dictionary

→ set of product chains: maps id integer to list of id's that can be used as backup (no order in tie-breaking)

→ need helper functions
  → insert product into inventory dictionary / adds to it
  → determine product with highest inventory levels based on list

→ quantity of products added will map product id to quantity added, will use dictionary

→ need to make sure inventory doesn't drop below 0.

→ helper function to check inventory value won't drop below 0.

→ while loop to keep updating until our order is complete for each item

Thoughts for future development

→ searching product chain then searching Inventory is inefficient, so maybe use a different data structure that only has product chain id in product chain dictionary if it is in quantity

→ think of edge cases, write tests

→ prints should be asserts to throw error if not in dictionary