**Date & Time / Business Logic**

**31. How can we add a holiday calendar so due dates skip weekends/holidays?**  
Use Python’s datetime and maintain a set of holiday dates. When calculating due dates, loop forward until the date is not Saturday, Sunday, or in the holiday set.

**32. How do we allow books to be reserved when already borrowed?**  
Maintain a **queue (collections.deque)** for each book. If a book is borrowed, the next member gets added to the queue. When the book is returned, the first member in queue is notified/assigned.

**33. How can we implement a renewal system where members can extend due dates only once?**  
Add a boolean flag renewed=False in borrow record. On renewal, check this flag; if False, extend due date and set renewed=True. If True, raise an exception.

**34. How can we track and print a monthly report of top borrowed books?**  
Keep a counter (collections.Counter) that increments every time a book is borrowed. At the end of each month, sort and display the top N books.

**Performance & Optimization**

**35. How do we use generators to lazily iterate through all books?**  
Instead of returning a full list, use yield to generate books one at a time:

def iter\_books(library):

for book in library:

yield book

**36. How do we profile the system using cProfile and identify bottlenecks?**  
Run:

python -m cProfile -s time library.py

This shows where the most time is spent (sorting, saving, searching, etc.).

**37. How do we cache frequently accessed books using functools.lru\_cache?**

from functools import lru\_cache

@lru\_cache(maxsize=128)

def get\_book(book\_id):

return library[book\_id]

**38. How can we simulate 100 members borrowing simultaneously using multiprocessing?**

from multiprocessing import Pool

def borrow\_task(member\_id):

library.borrow(member\_id, "B101")

with Pool(10) as p:

p.map(borrow\_task, range(100))

**39. Why should we replace normal dictionaries with defaultdict or OrderedDict?**

* defaultdict: avoids KeyError by providing default values (e.g., list for reservations).
* OrderedDict: maintains insertion order (useful for logs/reports).

**40. How do we benchmark file vs JSON persistence performance with 10,000+ books?**  
Use time module:

import time, json, pickle

start = time.time()

with open("books.json", "w") as f:

json.dump(big\_data, f)

print("JSON time:", time.time() - start)

start = time.time()

with open("books.pkl", "wb") as f:

pickle.dump(big\_data, f)

print("Pickle time:", time.time() - start)