Northeastern University

CS 2100: Program Design and Implementation 1

Practice Quiz 4

Instructions

- Please put all of your answers on the answer sheet. Only the answer sheet will be graded.
- Do not begin the quiz until instructed to do so.
- You may use both sides of a sheet of paper up to 8.5”x11” for reference, but no other resources, including phones, computers, AI, headphones, and ear pods.
- You have until the end of the class period to complete the quiz.
- Students may not leave the classroom during the first 10 minutes of the quiz (except in case of emergency).
- Hand your completed answer sheet to an instructor before leaving the room.
- Talk to an instructor if you need to leave the room and reenter.

Stakeholder-value matrices (revisited from Quiz 2)

1. Consider a personalized video recommendation system, such as the one that YouTube uses to suggest videos to watch. Which of these is NOT a relevant stakeholder?
   - a. Video creators
   - b. Viewers who are unable to quickly click suggested videos
   - c. People who are not viewers, have never used a computer before, and are not in any videos
   - d. People who label the data which is used in the recommendation algorithm
2. Considering the personalized video recommendation system, which of these is NOT a relevant value?
   - a. Profitability
   - b. Smell
   - c. Fairness
   - d. Decreasing carbon emissions
3. Out of the following, which would NOT be appropriate to put in the Stakeholder-Value Matrix where the value is mental health, and the stakeholder is viewers who want to hear about world news?
   - a. The viewer should hear about all relevant news, even if it is detrimental to their mental health.
   - b. The viewer may have gone on a mental health walk before watching the videos.
   - c. The viewer should get to choose from a variety of news videos, some of which are positive, and some of which are negative.
   - d. The amount that the viewer cares about mental health may change while a video is being played.
4. Which of the following is true?
   -

- a. Stakeholder-value matrices must include all possible values to be useful.
- b. Stakeholder-value matrices should be part of the design process before writing a program.
- c. It is important to consider scammers, fraudsters, and other “antagonists” when selecting stakeholders for the matrix.
- d. Stakeholder-value matrices should be written by an AI, so that humans don’t need to think through the stakeholders and values.

Correlation (revisited from Quiz 2)

5. For videos, if the correlation coefficient for the number of likes and the number of views is 0.95, what can we conclude?
   - a. Having more likes causes the video to be recommended more, and therefore watched more.
   - b. Having more views causes people to be more likely to like the video.
   - c. A higher number of views is associated with a higher number of likes.
   - d. The number of views is likely unrelated to the number of likes.
6. Assuming that people who comment on videos are also likely to click the like button on videos, what would make sense as a correlation coefficient between comments and likes?
   - a. 80
   - b. 0.8
   - c. -0.8
   - d. -80
7. Let’s assume that whether someone likes watching cat videos is unrelated to whether they like watching elephant videos. What would make sense as a correlation coefficient between viewers’ view counts for cat videos and their view counts for elephant videos?
   - a. -0.8
   - b. -0.03
   - c. 0.5
   - d. 0.99
8. If the correlation coefficient between viewers’ view counts for dog videos and their view counts for fish videos is -0.99, what can we conclude?
   - a. If a viewer likes fish videos, we should recommend them a lot of dog videos, too.
   - b. If a viewer likes fish videos, we should not recommend them many dog videos.
   - c. If a viewer likes fish videos, we have no information about whether we should recommend them dog videos.
   - d. Videos with fish in them are likely to also have dogs in them.

The next questions will use the following code:

```python
from collections.abc import Iterable, Iterator


class Video:
    """A video episode in a streaming service."""

    def __init__(
        self, title: str, season: int, episode: int, allowed_locations: list[str]
    ) -> None:
        self.title = title
        self.season = season
        self.episode = episode
        self.allowed_locations = allowed_locations

    def __eq__(self, value: object) -> bool:
        if not isinstance(value, Video):
            return False
        return (
            self.title == value.title
            and self.season == value.season
            and self.episode == value.episode
        )

    def __lt__(self, other: "Video") -> bool:
        if self.season == other.season:
            return self.episode < other.episode
        return self.season < other.season


class Playlist(Iterable[Video]):
    """A playlist of videos."""

    def __init__(self, videos: list[Video], location: str) -> None:
        self.videos = videos
        self.location = location
        self.index = 0

    def __iter__(self) -> Iterator[Video]:
        return PlaylistIterator(sorted(self.videos), self.location)


class PlaylistIterator(Iterator[Video]):
    """Iterator for Playlist that yields videos allowed in a specific location."""

    def __init__(self, videos: list[Video], location: str) -> None:
        self.videos = videos
        self.location = location
        self.index = 0

    def __next__(self) -> Video:
        while self.index < len(self.videos):
            current_video = self.videos[self.index]
            self.index += 1
            if self.location in current_video.allowed_locations:
                return current_video
        raise StopIteration
```

Coupling / cohesion / encapsulation

9. To improve encapsulation in the `Video` class, which change would be most appropriate?

   - a. Rename `allowed_locations` to have a name that more clearly indicates what it is
   - b. Remove the `allowed_locations` attribute entirely
   - c. Make `allowed_locations` a class variable instead of an instance variable
   - d. Add a method `is_allowed_in(location: str) -> bool` to check if a video is available in a given location

10. How is the cohesion of the `Video` class?

    - a. Low cohesion: the class should be split into separate classes for season and episode data
    - b. High cohesion: all methods and attributes are focused on representing a video episode
    - c. Medium cohesion: the `__lt__` method doesn't belong in this class
    - d. Medium cohesion: some methods are doing multiple unrelated tasks

11. The `Playlist` class stores videos in a `list[Video]`. What type of coupling does this create?

    - a. No coupling, because `list` is a built-in Python type
    - b. Coupling between `Playlist` and `Video`, because `Playlist` depends on the internal structure of `Video`
    - c. Coupling between `Playlist` and `Video`, because `Playlist` controls the behavior of `Video` objects
    - d. Coupling between `Video` and `list`, because `Video`'s structure depends on how `list` is implemented

12. Which of the following changes to `Video` would break the code in `PlaylistIterator` due to tight coupling?

    - a. Adding a new attribute `duration` to the `Video` class
    - b. Renaming `allowed_locations` to `available_regions` in the `Video` class
    - c. Adding a new method `get_season_info()` to the `Video` class
    - d. Changing the implementation of `__eq__` in the `Video` class

Iterator

13. What happens if you iterate over the same `Playlist` object twice?

```
playlist = Playlist([video1, video2], "US")
[video for video in playlist]  # First iteration
[video for video in playlist]  # Second iteration
```

-
    - a. Both iterators will work correctly and return the same videos
    - b. The second iterator will be empty because the iterator is exhausted
    - c. A `RuntimeError` will be raised on the second iteration
    - d. The second iterator will continue from where the first one ended

14. Why does `PlaylistIterator` need the `index` attribute?

```
- a. To keep track of the total number of videos
- b. To sort the videos
- c. To maintain state about the current position in the iteration
- d. To modify the videos
```

15. How does `PlaylistIterator` handle videos that are not allowed in the current location?

```
- a. It removes them from the list before iteration begins
- b. It raises a `ValueError` for restricted videos
- c. It returns `None` for restricted videos
- d. It skips them during iteration by checking each video and incrementing the index
```

16. Why does the `Playlist` class inherit from `Iterable[Video]`?

```
- a. To be able to store a collection of `Video` objects
- b. To make it so that `Playlist` can be used in a `for` loop
- c. To indicate to a reader that `Playlist` can be used in a `for` loop and must implement `__iter__`
- d. To automatically generate an `__iter__` method
```

Comparable

17. Why does `__eq__` include the check `if not isinstance(value, Video)`?

    - a. To handle comparisons with non-`Video` objects gracefully
    - b. To improve performance
    - c. To raise a `TypeError` for invalid comparisons
    - d. It's unnecessary and could be removed

18. How does `__lt__` determine if one video is "less than" another?

    - a. Alphabetically by title
    - b. By episode number only
    - c. By season first, then by episode number within the same season
    - d. By the number of allowed locations

19. If you tried to do `video1 > video2`, what would happen with the current implementation?

    - a. It would raise a `TypeError` because `__gt__` is not implemented
    - b. It would work correctly, taking the opposite of the boolean `video1 < video2`
    - c. It would work correctly, taking the opposite of the boolean `video1 < video2 or video1 == video2`
    - d. It would be `False` by default

20. Is there a potential inconsistency between `__eq__` and `__lt__`?

    - a. No inconsistency - they use the same attributes
    - b. Yes - `__eq__` includes `title` but `__lt__` ignores it
    - c. Yes - `__eq__` checks `allowed_locations` but `__lt__` doesn't
    - d. No inconsistency - they're independent operations

Recursion

21. This recursive function has a subtle bug. What happens when `n = 0`?

```
def power(base: int, n: int) -> int:
    if n == 1:
        return base
    return base * power(base, n - 1)
```

-
    - a. Returns `base`
    - b. Returns 0
    - c. Returns 1
    - d. Infinite recursion

22. In this code, how many times is `fib(2)` computed when calling `fib(5)`?

```
def fib(n: int) -> int:
    if n <= 1:
        return n
    return fib(n - 1) + fib(n - 2)
```

-
    - a. 1 time
    - b. 2 times
    - c. 3 times
    - d. 5 times

23. What does this function return for `mystery([1, 2, 3, 4], [])`?

```
def mystery(source: list[int], dest: list[int]) -> list[int]:
    if not source:
        return dest
    dest.append(source[0])
    return mystery(source[1:], dest)
```

-
    - a. `[1, 2, 3, 4]`
    - b. `[4, 3, 2, 1]`
    - c. `[]`
    - d. `[[1], [2], [3], [4]]`

24. Without storing and re-using the solutions to sub-problems, which of these recursive patterns will be the LEAST efficient?

a.

```
def pattern_a(n: int) -> int:
    if n <= 0:
        return 1
    return pattern_a(n - 1) + pattern_a(n - 2) + pattern_a(n - 3)
```

b.

```
def pattern_b(n: int) -> int:
    if n <= 0:
        return 1
    return 2 * pattern_b(n - 1)
```

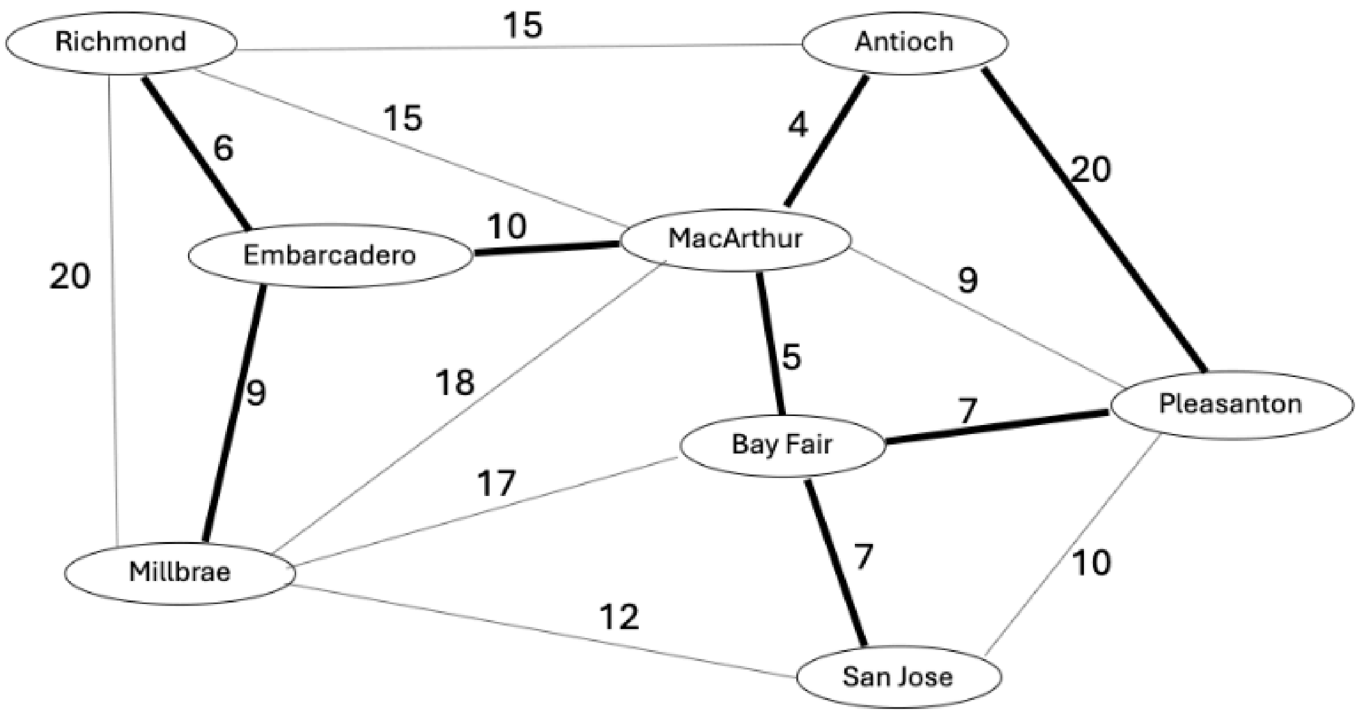
c.

```
def pattern_c(n: int) -> int:
    if n <= 0:
        return 1
    return pattern_c(n // 2) + pattern_c(n // 2)
```

d. All three are equally efficient

Minimum Spanning Trees

25. The bold edges in the following graph do NOT form a Minimum Spaning Tree (MST).



MST

- An MST needs to satisfy all the following criteria. Which one is broken?
  - a. Every node is connected to a selected edge.
  - b. The selected edges comprise a connected graph.
  - c. The selected edges do not form a cycle.
  - d. The selected edges form a spanning tree with lower total weight than any other spanning tree of the graph.

26. Under which condition is the minimum spanning tree of a graph guaranteed to be the only possible minimum spanning tree for that graph?
  - a. When all nodes in the graph are connected to each other (so there are not multiple disjoint sets)
  - b. When all edge weights are distinct (no two edges have the same weight)
  - c. When the graph has no cycles
  - d. When all edge weights are 1

27. What happens if some edges have negative weights?
  -

- a. Kruskal’s algorithm fails
- b. Kruskal’s algorithm works fine
- c. MSTs are undefined for graphs with negative weights
- d. It is impossible for a graph to have an edge with a negative weight

28. In Kruskal’s algorithm, what is the purpose of Union-Find?

- a. To sort edges by weight efficiently
- b. To detect whether adding an edge would create a cycle
- c. To store the edges of the MST
- d. To find the minimum weight edge