Smart Task Scheduler Exercise in Kotlin

1. Setting Up Kotlin

To set up Kotlin on your computer (PC or Mac), download and install IntelliJ IDEA Community Edition (free) from JetBrains's website. Kotlin support is built into IntelliJ IDEA, and the IDE will help you with Kotlin syntax and features. In the exercise ZIP, you will find a project that you can open with IntelliJ IDEA to start working in.

2. Exercise Description: Smart Task Scheduler

You'll create a task scheduling system that demonstrates Kotlin's key features while creating a practical little application. The system should help users manage and prioritize tasks with different deadlines and dependencies.

Tasks have the following properties:

- Title - Text
- Priority (High, Medium, Low)
- Deadline (nullable)
- Dependencies (other tasks that must be completed first)
- Estimated duration (hours)
- Status (Not Started, In Progress, Completed)

The following functions should be available:

- Add and remove tasks
- Update task status
- Get a next recommended task based on priorities and dependencies
- Calculate an optimal sequence of tasks considering deadlines and dependencies
- Calculate whether all deadlines can be met given estimated durations and assuming an 8h work day, starting work on the next calendar day.

Please note that there is no GUI required, so simply call the functions from main to demonstrate that they are working. Use print statements to provide outputs.

3. Key Kotlin Concepts and Hints

Here are some key Kotlin concepts that I recommend looking up, as you will need them to solve the exercise:

- *Data Classes*: You will need them to represent tasks and their properties
- *Null Safety*: Handle optional task properties
- *Lambda Functions*: Task filtering and scheduling logic
- *Collections Processing*: Managing and organizing tasks. While you can apply your knowledge from Java to use for this, Kotlin provides a lot of cool collection processing support that will make it much easier to implement.

You can find some code snippets in the starter project for each of these concepts, the file name is conceptSnippets.kt.

Once you are ready, start with implementing basic task management before tackling scheduling logic. Test with different scenarios, by calling your functions from main. Make use of Kotlin's smart casts when handling nullable types. If you are stuck, try implementing what you need in Java, and using built-in IntelliJ conversion to Kotlin to get at least a basic version going.

4. Assessment Criteria

Your solution should meet these criteria:

- Meet the functional requirements as described, with meaningful print statements as outputs.
- Task dependencies are properly handled (no circular dependencies)
- Proper use of null safety features (no forced nullability with !!)
- Effective use of data classes and their built-in functions
- Clear and meaningful variable/function names following Kotlin naming conventions (camelCase for functions/variables, PascalCase for classes)
- Include appropriate documentation comments

5. Submission

Submit your solution as a ZIP file or as a text file with a link to a publicly accessible GitHub repository.