1. **Evaluate the run-time and memory of data structures that could be used to address the requirements**.  
   Vector;

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 2 | n | n |
| **Total Cost** | | | 6n + 1 |
| **Runtime** | | | 1(n) |

Hash Table;

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 2 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 2 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 9n + 1 |
| **Runtime** | | | O(n) |

Binary Tree;

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 8n + 1 |
| **Runtime** | | | O(n) |

Analysis of Each Data Structure:

Vector:

* 1. Advantages:
     1. I appreciate the simplicity of implementing a vector.
     2. It maintains the order of insertion, which suits listing courses.
     3. If my dataset is relatively small or not expected to undergo many search or insert operations, a vector could work well.
  2. Disadvantages:
     1. I understand that search time could be slow with a linear search (1(n)) for larger datasets.
     2. The vector's sorting time complexity is 1(nlogn).

Hash Table:

* 1. Advantages:
     1. I acknowledge that hash tables provide fast average-case lookups, inserts, and deletes (O(1) amortized time).
     2. If my dataset grows or requires frequent search operations, a hash table could be efficient.
     3. It's designed for mapping between keys (course codes) and values (course objects).
  2. Disadvantages:
     1. I understand that worst-case operations can degrade due to collisions.
     2. A good hash function is needed, and it might involve more memory overhead.

Tree:

* 1. Advantages:
     1. I see the efficiency of search, insert, and delete operations in balanced trees.
     2. The guaranteed sorted order could be useful for tasks like sorting and listing.
     3. Balanced trees handle dynamic data well and maintain balance during changes.
  2. Disadvantages:
     1. I recognize that trees are more memory-intensive and complex due to their structure.
     2. Unbalanced trees could lead to worst-case 8(n) operations.

Recommendation and Justification:

Based on the analysis and my preference for using a vector, I've considered the following factors:

* 1. My program appears to focus on reading and displaying course information without heavy search, insert, or delete operations.
  2. If my dataset is expected to remain relatively small or manageable, the linear search and insertion times might not have a significant impact on performance.
  3. The simplicity of the vector implementation might lead to faster development and easier maintenance, which I appreciate.
  4. The requirement to maintain insertion order aligns well with the vector's characteristics, as it keeps things straightforward and intuitive.

However, I'm aware that if my dataset grows significantly or if search and sort operations become performance bottlenecks, I should consider using a more efficient data structure, such as a hash table or a balanced tree, to address these aspects and potentially improve my program's overall performance.