Week 3 Quiz 2

1. Select all of the following statements that are an accurate description of hash tables?
   - Hash tables maintain the order of the data added to them
   - The indexes of a hash table can be many types, including integers and strings

     Hash tables map data from a wide variety of types, including integers and strings, to their associated data stored in the hash table
   - A hash table requires a full traversal of its contents to determine if it contains a specified piece of information
   - Hash tables can only be used to store numeric data
2. Which of the following best characterizes a binary search tree?
   - Binary search trees are designed to automatically stay balanced as new nodes are added
   - Each node in a binary search tree can have at most two child nodes
   - When building a binary search tree, the final structure does not depend on the order in which nodes are added
   - If a node in a binary search tree is a parent to two child nodes, then the children will either both be larger than the parent, or both be smaller.
3. What should you do to ensure the robustness of your code when working with LLMs? Select all of the following statements are True.
   - Use LLMs only after you've made all software design decisions (e.g. which data types to use) to avoid them biasing the types of solutions you consider
   - Ask follow-up and clarifying questions of LLMs to clarify any questions about generated code

     Following up with LLMs and asking clarifying questions can help refine their responses and clarify any lingering questions about the code they generate or suggestions they provide.
   - Regularly test LLM-generated code

     Regularly testing and iterating on code helps ensure its robustness and reliability.
   - Trust the code generated by LLMs without question.
4. Imagine you are designing a piece of software alongside an LLM. You share this initial prompt:

   "Write me some Python code to count the number of times each word appears in a given piece of text"

   After generating the code you find that the code is lacking many pieces of functionality that you'd like it to include, like error handling and input validation. You also are concerned that the code may not be optimized for scale to large amounts of text. Which of the following is the best approach to achieve the improvements you want in your code.

- Write the code without LLM support. If the LLM did not include those features initially it is unlikely to be able to do so on future attempts.
- Regenerate the code with the same prompt as before in hopes that these missing features are included
- Iteratively prompt the LLM in the same ongoing conversation with a description of the desired new functionality to add the missing features
- First assign the role of "an expert programmer" to the LLM and then re-run the same prompt as before

  By iteratively prompting the LLM with a clear description of the new features you want in the code it generates, you're likely to end up closer to a working piece of code that includes those features.

5. What benefits does a doubly linked list provide over a singly linked list?
   - Doubly linked lists use less memory than singly linked lists
   - Doubly linked lists can be traversed forwards and backwards as each node contains a pointer to the nodes before and after it
   - Doubly linked lists can contain twice as many nodes as singly linked lists
   - Doubly linked lists are composed of two singly linked lists that store redundant copies of a list of data

   The nodes of a doubly linked list contains pointers to the nodes before and after it, allowing for easy traversal in both directions.