1.
   a. Blood types
      i. Strength: There is a significantly low Big O notation for frequent name searches
      ii. Weakness: When inserting and deleting elements you would have to shift elements around, which is inefficient for rare operations
      iii. Better choice: Array sorted list, since there are infrequent updates to the list.
   b. Favorite Song
      i. Strength: There is simple integration, good for frequent additions and deletions
      ii. Weakness: They are bad for linear search for individual songs
      iii. Better Choice: Array unsorted list this is because constant-time access for searches, efficient for frequent lookups.
   c. Animal Sightings
      i. Strength: O(1) insertion/deletion at the beginning/end, ideal for recent sightings
      ii. Weakness: Requires sorting, potentially costly for large lists
      iii. Better Choice: Linked List Unsorted list, this prioritizes recent items.
2.
   a. Linear Search and Binary Search
      i. Linear search iteratively checks each element in a list sequentially until the target is found or the end is reached. Binary search is done recursively, it divides a sorted list in half comparing the target to the middle element. If not found the search continues in the relevant half. The big O notation for Linear seach is O(n), while the big O notation for binary search is O(log n). This means that binary search time grows much slower than linear search, which grows linearly.
   b. Stack ADT and Queue ADT
      i. Stack is Last-In-First-Out (LIFO) data structure, where elements are added and removed from the top. Queue is a First-In-First-Out (FIFO) data structure, where elements are added to the "back" and removed from the front. Both are fundamental data structures for managing element order. Stacks are better for function call history or undo/redo functionality, while queues are better in processing elements in the order that they are added. Stacks are generally easier to implement efficiently with arrays.
   c. Dynamically Allocated Arrays is an array whose size can be adjusted at runtime using memory allocation functions like malloc and free. Vector classes are a container class in C++ that manages a dynamically allocated array internally, providing methods for element access, insertion, and deletion. Both are dynamic array storage but they are different in their approach. The arrays need manual memory management, while vector classes handle it internally. This makes them safer and easier to use, but they are less efficient in memory-critical applications.