1.
    a. Sorting on demand means organizing data only when needed, unlike pre-sorting everything like with sorted lists or binary search trees. This saves resources for randomly accessed data, large datasets, and frequently changing information. While pre-sorted structures excel at specific searches, sorting on demand offers a more adaptable solution for various use cases.
    b. Insertion Sort works like building a sorted hand of cards, iteratively placing each element in its correct spot among previously sorted data. This makes it fast for small or already-sorted collections but inefficient for random data. Heap Sort, in contrast, builds a priority queue (heap) to efficiently find the maximum (or minimum) element, removes it as sorted, and rebuilds the heap with remaining data. While slower to set up than Insertion Sort, Heap Sort excels in worst-case scenarios regardless of initial order due to its efficient element selection process. So, for small or pre-sorted data, Insertion Sort shines, while Heap Sort dominates for larger, random datasets.
2.
    a. Set ADT stores a collection of unique elements. The main difference between the two ADT, unsorted and sorted lists, are that it does not allow for duplicates
    b. An example of when using bit-vector implementations are good is when using a spell checker. Bit vectors would track unique words encountered in a document. An example of when using bit-vector implementation are bad is in terms of storing songs in a music streaming service. The large number of songs would need a huge bit vector.
3.
    a. A hashing function is a special algorithm that takes an arbitrary amount of data and squishes it into a fixed-size value, called a hash code.
        i. An example for numeric data would be, having a hashing function that squares a number and then takes the remainder when divided by 10. The number 121 would hash to 1.
        ii. An example for non numeric data would be, having a hashing function that converts strings into numbers based on character encoding scheme, and then perform the same kind of operations as with numeric data. For instance, the string “apple” might have has to 5 using a specific function.
    b. A hash table is a data structure that uses hashing functions to achieve fast insertion, searching, and deletion. The hash code is used to find the actual data in the hash table. This offers a significant performance advantage over sorted or unsorted lists because you can jump straight to the relevant data using the hash code, instead of iterating through the whole list.
    c. An issue to hash tables though is collisions. Collisions happen when different data points hash to the same value. THis can cause data being crammed into the same

slot in the hash table, slowing down lookups. To mitigate this, hash tables often use techniques like separate chaining or open addressing. Separate chaining creates linked lists at each slot to store multiple colliding elements. Open addressing uses a probing strategy to find the next available slot when a collision occurs.

d.

   i. The better approach would be option 2, because option groups employees by departments and this would lead to a high number of collisions. Option 2 is better because it uses the sequential nature of the employees IDs within the company. When more employees are hired, their last four digits will be more spread out, causing fewer collisions

   ii. To improve option 2 you can add the department code to the hash function. You would concatenate the department code with the employee number to make an 8 digit string. You would then apply the chosen hashing function to the entire 8 digit string. This would distribute the employees more evenly through the hash table since it is incorporating both their department and hiring order. This would decrease the collisions compared to if it was just the last 4 digits.