Name: Bhavani Dande

Uid: U09832941

AI Assistance Documentation

Question 1: Find Missing Numbers in Array

Prompt Used:

"Find missing numbers in an array using O(n) time"

Response Received:

Copilot suggested iterating through the array and marking visited indices by negating the values at those indices. Then, a second pass collects indices that remain positive, indicating missing numbers.​

Implementation Details: The suggested approach was implemented as follows:​

* Iterate through the array, for each value, compute the corresponding index and negate the value at that index to mark it as visited.​
* In a second pass, collect all indices that have positive values, as they represent the missing numbers.​

Adjustments: Added checks to handle cases where numbers might be out of the expected range or duplicates are present.​

Question 2: Sort Array by Parity

Prompt Used:

"Sort array by parity"​

Response Received: Copilot suggested using two pointers: one starting from the beginning and the other from the end of the array, swapping elements to ensure even numbers are at the beginning and odd numbers at the end.​

Implementation Details: The two-pointer technique was implemented:​

* Initialize two pointers, left at the start and right at the end of the array.​
* Move the left pointer forward if it's pointing to an even number.​
* Move the right pointer backward if it's pointing to an odd number.​
* If left points to an odd number and right to an even number, swap them.​
* Continue this process until the pointers meet.​

Adjustments: Ensured the solution handles arrays with all even or all odd numbers and added comments for clarity.

Question 3: Two Sum

Prompt Used:

"Two sum problem using dictionary"​

Response Received: Copilot suggested iterating through the array while maintaining a dictionary to store numbers and their indices. For each number, compute its complement with respect to the target and check if it's already in the dictionary.​

Implementation Details: The dictionary-based approach was implemented:​

1. Initialize an empty dictionary to store numbers and their indices.​
2. Iterate through the array, and for each number:​

* Compute its complement by subtracting it from the target.​
* Check if the complement exists in the dictionary.​
* If it exists, return the current index and the index of the complement.​
* If not, add the current number and its index to the dictionary.​

Adjustments: Handled cases where the same element cannot be used twice and ensured the solution works for negative numbers.

Question 4: Maximum Product of Three Numbers

Prompt Used:

"Maximum product of three numbers”

Response Received: Copilot suggested sorting the array and considering the product of the three largest numbers and the product of the two smallest numbers with the largest number, returning the maximum of these.​

Implementation Details: The approach was implemented as follows:​

1. Sort the array in ascending order.​
2. Compute the product of the last three elements (three largest numbers).​
3. Compute the product of the first two elements (two smallest numbers, which could be negative) and the last element (largest number).​
4. Return the maximum of the two computed products.​

Adjustments: Added handling for arrays with less than three elements and included comments to explain the logic.​

Question 5: Decimal to Binary Conversion

Prompt Used:

"Convert decimal to binary"​

Response Received: Copilot suggested repeatedly dividing the decimal number by 2 and recording the remainders, then reversing the sequence of remainders to get the binary representation.​

Implementation Details: The conversion was implemented as follows:​

1. Initialize an empty string to store the binary representation.​
2. While the decimal number is greater than 0:​

* Compute the remainder when dividing by 2.​
* Prepend the remainder to the binary string.​
* Divide the decimal number by 2 (integer division).​

1. If the decimal number is 0, return "0".​
2. Return the binary string.​

Adjustments: Handled the edge case where the input is 0 and ensured the binary string is constructed efficiently.