Question 1:

(Decimal to hex) Write a program that prompts the user to enter an integer and displays its corresponding hex number. Here are some sample runs:

Enter a decimal value: 11
The hex value is B

Enter a decimal value: 16
The hex value is 10

For converting decimal to hex, you CANNOT use Python's built-in methods such as hex() function, string's format(), or format specifier, so you must use a while loop to resolve the problem.

Question 2:

Write a menu-driven program that allows a user to enter a list of scores and then choose between finding the smallest, largest, sum, average or mode. The mode is the score that occurs the most often. It can be determined as a by-product of building the frequency array. After building the frequency array, use it to determine which score occurred the most.

Use a if-elif statement to determine what action to take. Provide an error message if a valid choice is entered. Run the program five times, once with each option and once with an invalid option.

Note: You cannot use Python built-in functions for this question. For example, collections.Counter, count(), sum() etc cannot be used. Additionally, you cannot use a Python dictionary to find the mode.

Question 3:

Write a program that merges two sorted lists of numbers into a new sorted list. You cannot use any Python built-in functions, such as sort() or sorted(). Other lists or data structures cannot be used to store temporary results. Most importantly, the complexity of your algorithm must be of the order of n, i.e. O(n).

Example,
List1: [1, 3, 5, 7]
List2: [2, 4, 6, 8]
Result: [1, 2, 3, 4, 5, 6, 7, 8]

Question 4:

A school cafeteria is giving an electronic survey to its students to improve their lunch menu. Create a program that uses a list to store votes for the survey. The program should display four food items in the display. Then it prompts students to enter whether they like or dislike a particular food. Display a report that's updated as each

new set of responses is entered. Use a 2-D Integer array named votes, with four rows and two columns to summarize the results. Each row corresponds to a food item. The columns store the number of “like” and “dislike” votes, respectively.

The output is expected to be printed in a nice tabular format that is easy for humans to read, as shown in the sample below. Your marks will be reduced if you simply code it as "print(votes)".

	Like	Dislike
Pizza	2	0
Hot Dog	1	1
Ham	0	2
Cheese	1	1

Question 5:

Consider a black-and-white image made up of pixels, where each pixel’s shade is represented by a number: 0 for black, 255 for white, and other numbers for different shades of gray.

1. Create a function named apply_negative_filter(image) that:
    - Receives a grid (2D list) of pixel brightness values.
    - Changes each pixel’s value to its opposite shade (black becomes white, white becomes black, and gray becomes the opposite shade of gray).
    - Returns a new grid with the inverted shades.
2. Create another function named apply_edge_detection(image) that:
    - Works with the same grid of pixel brightness values.
    - Looks at each pixel (except the ones on the edge) and compares its brightness to the pixels directly next to it (above, below, left, and right).
    - If the difference in brightness is large (more than a set amount, like 30), the pixel is changed to white (255) in a new grid. If not, it’s changed to black (0).
    - The edge pixels in the new grid should all be black (0).
    - Returns the new grid showing only the edges of shapes in white
3. Write a main() and print_image() to test your functions with a sample image of random numbers:

```
def main():
    # Modify the following code to generate a sample image filled with random brightness values, where each value ranges from 0 (black) to 255 (white).
    sample_image = [
        [0, 50, 100, 150, 200, 255],
        [255, 200, 150, 100, 50, 0],
        [0, 50, 100, 150, 200, 255],
        [255, 200, 150, 100, 50, 0]
    ]

    print("Original Image:")
    print_image(sample_image)

    negative_filtered_image = apply_negative_filter(sample_image)
    print("\\nNegative Filter Applied:")
    print_image(negative_filtered_image)

    edge_detected_image = apply_edge_detection(sample_image)
    print("\\nEdge Detection Filter Applied:")
    print_image(edge_detected_image)
```