Longest Increasing Subsequence.

Program Overview:

The program is designed to find the length of the longest increasing subsequence (LIS) from a list of numbers entered by the user. A LIS is a subsequence of the given sequence in which the elements are sorted in increasing order and the subsequence is as long as possible.

Implementation Details:

1. **Input Handling:**
    - The program prompts the user to enter the number of elements in the array.
    - Then, the user is asked to input each element of the array.
2. **Computing the LIS Length:**
    - The program utilizes dynamic programming to efficiently compute the length of the LIS.
    - It initializes an array **lis** of the same length as the input array to store the length of the LIS ending at each index.
    - For each element in the array, the program iterates through all previous elements to determine the length of the LIS ending at that element.
    - It compares the LIS length of the current element with the LIS length of previous elements, updating **lis[i]** accordingly to represent the length of the LIS ending at index **i**.
3. **Output:**
    - Once the LIS lengths for all elements are computed, the program finds the maximum LIS length.
    - The maximum LIS length represents the length of the longest increasing subsequence in the array.
    - The program then displays the length of the longest increasing subsequence to the user.

Conclusion:

The program efficiently computes the length of the longest increasing subsequence using dynamic programming techniques. By iteratively updating LIS lengths for each element in the array, it accurately identifies the maximum length of the increasing subsequence.

Through user-friendly input handling and clear output presentation, the program enhances usability and provides a seamless experience for the user. Its ability to handle dynamic input sizes and efficiently compute the LIS length makes it applicable in various scenarios where the identification of the longest increasing subsequence is required.

Overall, this program demonstrates an effective implementation of dynamic programming concepts to solve a real-world problem, showcasing its versatility and utility in solving similar tasks.

Algorithm: -

1. **Input Handling:**
    - Prompt the user to enter the number of elements in the array.
    - Read the array size **n**.
    - Create an integer array **arr** of size **n** to store the input elements.
    - Prompt the user to enter each element of the array.
    - Read the array elements into the **arr** array.
2. **Compute the LIS Length:**
    - Initialize an integer array **lis** of size **n** to store the length of the LIS ending at each index.
    - Iterate over each element **arr[i]** in the array:
        - Initialize **lis[i]** to 1, representing the LIS length ending at index **i**.
        - Iterate over all elements before **arr[i]** (indexed as **j**):
            - If **arr[i]** is greater than **arr[j]**, update **lis[i]** to the maximum of its current value and **lis[j] + 1**.
    - After completing the iterations, **lis[i]** will store the length of the LIS ending at index **i**.
3. **Find the Maximum LIS Length:**
    - Iterate over the **lis** array and find the maximum value.
    - Store the maximum value as **maxLIS**, which represents the length of the longest increasing subsequence in the array.
4. **Output:**
    - Print the length of the longest increasing subsequence (**maxLIS**) to the console.

Pseudocode:

```
Pseudocode:

function findLISLength(arr: array of integers, n: integer) -> integer:
    lis = array of integers of size n, initialized with 1s
    maxLIS = 0
    for i from 0 to n-1:
        for j from 0 to i-1:
            if arr[i] > arr[j]:
                lis[i] = max(lis[i], lis[j] + 1)
        maxLIS = max(maxLIS, lis[i])
    return maxLIS

Main:
    Read n from user
    Create array arr of size n
    Read elements into arr
    lisLength = findLISLength(arr, n)
    Print "Length of the longest increasing subsequence is: " + lisLength
```

This algorithm outlines the steps involved in calculating the length of the longest increasing subsequence from a given array of integers. It uses dynamic programming to efficiently compute the LIS length for each element and then finds the maximum length among all LIS lengths. Finally, it outputs the length of the longest increasing subsequence to the user.