C++ Programming

1D Arrays Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Find the 3 minimum values

- Read integer N (>= 3), then read N integers. Find the 3 lowest numbers.
  - Don’t change the array content
  - Don’t iterate on the array more than once
- Input ⇒ Output
  - 5 **4 1 3 10 8** ⇒ 1 3 4
  - 3 **7 9 -2** ⇒ -2 7 9

Problem #2: Search for a number

- Read an Integer N, then read N <= 200 integers [0 <= A[i] <= 500].
  - We will search in this array for numbers
- Then read integer Q (for a number of queries), then read Q integers
  - For each integer, find the **last occurance** in the array. Print its index
  - If doesn't exist, print -1
- Input 5 **1 2 7 3 7** 3 **7 9 2**
  - Means Array of 5 numbers (1 2 7 3 7) and 3 queries (7 9 2)
- Output
  - 4 [7 exists in 2 positions (2 and 4). The last is 4)
  - -1 [9 doesn't exist)
  - 1 [2 exists only in position 1]
- Do it first with nested loops. Can you do without any nested loops?

Problem #3: Find most frequent number

- Read an Integer N, then read N <= 200 integers. Find the value that repeated the most number of times.
  - Each integer is **-500** <= value <= 270
- Example for array: 7 **-1 2 -1 3 -1 5 5**
  - -1 repeated 3 times: the largest
- Don’t use nested loops

Problem #4: Digits frequency

- Read an Integer N, then read N <= 200 integers. For all the digits from 0 to 9, we want to know how many times appeared
  - Input 2 78 307
  - Output:
  - 0 1
  - 1 0 [digit 1 never appeared]
  - 2 0
  - 3 1
  - 4 0
  - 5 0
  - 6 0
  - 7 2 [digit 7 appeared twice]
  - 8 1
  - 9 0

Problem #5: Unique Numbers of unordered list

- Read integer N (<= 900), followed by reading N integers (0 <= value <= 500)
- Print the **unique** list of the numbers, but **preserve** the given order
- Input: 13 **1 5 5 2 5 7 2 3 3 3 5 2 7**
- Output: 1 5 2 7 3
  - Observe: input is **not sorted** list
  - Observe: output preserves the original order: e.g. 5 appears before 2
- Don’t use nested loops

Problem #6: Sorting numbers

- Read integer N (<= 900), followed by reading N integers (0 <= value <= 500)
- Print the **sorted** list of the numbers
- Input: 13 **1 5 5 2 5 7 2 3 3 3 5 2 7**
- Output: 1 2 2 2 3 3 3 5 5 5 5 7 7
- Give your most efficient trial
  - You don’t need to google how to sort numbers
  - Hint: max value in the array is 500

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."