Byte Offset Array Increment

ZPRAC-16-17-Lab10

[BONUS]

In C an integer (int) is represented by 32 bits or 4 bytes.
For example, the integer 20 is represented in binary as:

00010100 00000000 00000000 00000000 (byte ordering is right to left, (little-endian))
---b3--- ---b2--- ---b1--- ---b0---

An array of integers is simply several such integers placed consecutively in memory.
For example, the array {12, 20, 15} is represented in memory as:

00001100 00000000 00000000 00000000 00010100 00000000 00000000 00000000
00001111 00000000 00000000 00000000
----------12---------- ----------20---------- ----------15----------

Your task is to increment such an array at a given byte offset and index and then print index+1 element of original array
Given the above array and offset of 5 bytes and index of 0:

00001100 00000000 00000000 00000000 00010100 00000000 00000000 00000000
00001111 00000000 00000000 00000000
----------5 bytes---------- ----------index 0---------- ----------index 1----------(overflows)

After incrementing index 0 by 1 with 5 bytes offset we have:
00001100 00000000 00000000 00000000 00010100 00000001 00000000 00000000
00001111 00000000 00000000 00000000
----------5 bytes---------- ----------index 0----------

And now index 0+1=1 of the original array:
00001100 00000000 00000000 00000000 00010100 00000001 00000000 00000000
00001111 00000000 00000000 00000000
----------index 0 (12)---------- ----------index 1 (276)---------- ----------index 2 (15)----------

INPUT FORMAT:

N (int) --- integer denoting number of elements in array

a1 a2 a3 ... aN --- N integers (elements of array)
T (int) --- integer denoting number of test cases for byte offset and index increment
b1 i1 --- byte offset and index of test case 1
b2 i2 --- byte offset and index of test case 2
.
.
.
bT iT --- byte offset and index of test case T

OUTPUT FORMAT:
---
o1 --- output for test case 1
o2 --- output for test case 2
.
.
.
oT --- output for test case T

EXAMPLE:

INPUT:
3
12 20 15
2
5 0
2 1

OUTPUT:
276
15

HINT: You are expected to use pointer typecasting for this question. An integer array, say "int arr[100]" is stored as 100 blocks of 4 bytes each. However, if the same array is instead interpreted as a character array (by type casting to char*), it will be "seen" as 400 blocks of one byte each. This will allow for incrementing of each byte (or "block") by a simple integer increment operation on the typecasted array.