Lab Practice Problem 5

Filename: `practice5_surname.c`

In this activity, you will implement a program that reads a number, stores its digits in a fixed-size array, performs an in-place cyclic left shift on the array, and prints the result.

Try It Out

Write a program that left-rotates the digits of a 5-digit number by `N` positions, wrapping around so any digit that moves past the left end reappears on the right. The following function prototypes are provided that define the **interfaces** you need to implement:

`void populate_array(int arr[], int size, int num);`
Convert `num` into exactly `size` digits where it is right-aligned. Add leading zeros if necessary.

`void print_array(int arr[], int size);`
Print all elements of `arr` consecutively (no spaces) on one line, then output a newline.

`int get_next_pos(int cur_pos, int size, int shift);`
Given a position and a left-`shift` amount, compute the wrapped position to read from next.

`void left_shift_array(int arr[], int size, int shift);`
Perform a cycle walk to rotate the array in place to the left by `shift` positions.

In the `main()` function, use a loop to process all test cases and display the results.

Input

The first line of input contains an integer $0 \leq \mathsf{T} \leq 100$, the number of test cases. The following T lines each contain two space-separated integers: $0 \leq$ `num` $\leq 99{,}999$ and $0 \leq$ `shift` $\leq 1{,}000{,}000$.

Output

For each test case, the program should print one line of output. The line begins with the test case number, followed by a colon and a space. After this, print the 5-digit number that is the result of left-rotating `num` by `shift` positions. The number must include leading zeros if needed.

Sample Input

```
3
10523 2
12345 16
23 1
```

Sample Output

```
1: 52310
2: 23451
3: 00230
```

Guide Questions

1. What should your program output when `shift` is `0`?
2. Why do we pass an argument `size` to functions instead of relying on a constant like `DIGIT_COUNT`? What advantages does this give for testing and reuse? If the task changes from 5 digits to 6, which parts of the code would need updating?
3. In C, when an array is passed to a function, what does the function actually receive? For example, the parameter `arr` in `populate_array()`. Why do such functions typically also need the array's `size`? What risks arise if a function does not know the array's size? For example, if `print_array()` omits the `size` parameter.
4. How does `populate_array()` guarantee the presence of leading zeros for numbers with fewer than 5 digits? How does it determine the correct number of zeros to add?
5. What should happen if `num` has more than 5 digits? For example, `012345` or `123456`. How does your solution currently handle these cases? Does the value of `shift` matter?
6. If the requirement changed so that leading zeros should not be printed (e.g., print `230` instead of `00230`), how would you modify `print_array()` without modifying its prototype?
7. Sketch the cycle walk for the left-rotation starting at position 0 for a fixed `size` (e.g., 5) and various `shift` values (0 through `size-1`). What patterns do you observe in how positions are visited?
8. We worked with digits stored in an array of integers. Now, imagine that instead of digits, we are dealing with letters. In this new context, the data is stored in an array of characters. Assuming that a correct `main()` function is already provided, how would you update the other functions to handle this change? Specifically, what modifications would you make to their prototypes and their definitions?

Sample Solution

The source code can be accessed here.