Write a Python function:

fast_copy(source, count)

that:

- Allocates a destination structure large enough to hold count elements.
- Determines how many complete transfer groups of eight elements are required.
- Performs any necessary initial element transfers using structured control flow rather than iteration.
- Completes the remaining transfers in a repetitive block that assigns exactly eight elements per iteration.
- Returns the destination structure containing the copied elements.

```
def fast_copy(source, count) #fast copy function
    dest = [0] * count   # Allocates a destination structure large enough to hold count elements.
    src = 0
    dst = 0
    groups = count // 8    #grouping elements
    remainder = count % 8 #partial group finding
#Any elements that do not fit evenly into a full group must be processed before the repeating portion begins
    if remainder >= 1:
        dest[dst] = source[src];
        dst += 1;
        src += 1
    if remainder >= 2:
        dest[dst] = source[src];
        dst += 1;
        src += 1
    if remainder >= 3:
        dest[dst] = source[src];
        dst += 1;
        src += 1
    if remainder >= 4:
```

```
            dest[dst] = source[src];
            dst += 1;
            src += 1
        if remainder >= 5:
            dest[dst] = source[src];
            dst += 1;
            src += 1
        if remainder >= 6:
            dest[dst] = source[src];
            dst += 1;
            src += 1
        if remainder >= 7:
            dest[dst] = source[src];
            dst += 1;
            src += 1

    while groups > 0:    #Coping group elements at once
            dest[dst]     = source[src]
            dest[dst + 1] = source[src + 1]
            dest[dst + 2] = source[src + 2]
            dest[dst + 3] = source[src + 3]
            dest[dst + 4] = source[src + 4]
            dest[dst + 5] = source[src + 5]
            dest[dst + 6] = source[src + 6]
            dest[dst + 7] = source[src + 7]
            dst += 8
            src += 8
            groups -= 1
return dest
```

Execution Constraints

1. Reduced Control Checks

Evaluating a loop condition for every element transfer introduces unacceptable overhead.

Instead, the logic must be structured so that control checks occur only after multiple transfers have been completed.

2. Fixed-Width Transfer Groups

Transfers must be organized into groups of exactly eight elements. Once the system enters the repeating portion of the routine, each iteration must perform eight explicit assignments without additional branching inside the group.

3. Partial Group Handling

The total number of elements to transfer may not align with the group size. Any elements that do not fit evenly into a full group must be processed before the repeating portion begins.

4. Explicit Control Flow for Partial Transfers

The partial group must be handled without using a compact counting loop (e.g., no short for or while constructs). Instead, control flow must be arranged so that execution begins at a position corresponding to the required number of initial assignments and then proceeds sequentially.

5. Element-by-Element Assignment Only

High-level bulk operations are not permitted. Each element must be copied using a single explicit assignment, simulating a register-to-memory transfer.