

Q.4 Declare a static local variable inside a function.  
Observe how its value persists across function calls.

# include <stdio.h>

Void demoFunction () {  
 static int count = 0;

Count ++;

Print f ("Count = %d/n", count);

}

Int main () {

Print f ("Calling demoFunction multiple time : /n");

demoFunction ();

demoFunction ();

demoFunction ();

demoFunction ();

Return 0;

}

The image shows a Mac desktop with an "Online Compiler" window open. The window has tabs for "main.c" and "Output". The "Output" tab is active, displaying the execution of a C program. The code in "main.c" demonstrates the use of the static keyword to ensure a variable's value persists across multiple function calls. The output shows the count being incremented from 1 to 4 each time the demo function is called.

```
main.c
1 #include <stdio.h>
2
3 void demo_function() {
4     // 'static' keyword ensures 'count' is initialized only on the
5     // first call
6     // and its value is preserved across all subsequent calls.
7     static int count = 0;
8
9     // Increment the persistent count
10    count = count + 1;
11
12    // Print the current value of the persistent count
13    printf("Count = %d\n", count);
14
15 int main() {
16     printf("Calling demo_function multiple times:\n\n");
17
18     // Call the function multiple times to observe the persistent
19     // count
20     demo_function();
21     demo_function();
22     demo_function();
23     demo_function();
24
25     return 0;
}
```

Output

```
Calling demo_function multiple times:
Count = 1
Count = 2
Count = 3
Count = 4
=====
== Code Execution Successful ==
```