Demonstration of unsafe string copying in C using `strcpy()`

```
#include <stdio.h>
#include <string.h>

int main() {
   char s1[] = "computer security is fun";
   char s2[15] = "";
   char s3[10] = "";

   printf("Before strcpy operations...\n");
   printf("s1: \"%s\" size: %zu length: %zu\n", s1, sizeof(s1), strlen(s1));
   printf("s2: \"%s\" size: %zu length: %zu\n", s2, sizeof(s2), strlen(s2));
   printf("s3: \"%s\" size: %zu length: %zu\n", s3, sizeof(s3), strlen(s3));

   strcpy(s2,s1);     // copy s1 to s2
   strcpy(s3,s1);     // copy s1 to s3

   printf("After strcpy operations...\n");
   printf("s1: \"%s\" size: %zu length: %zu\n", s1, sizeof(s1), strlen(s1));
   printf("s2: \"%s\" size: %zu length: %zu\n", s2, sizeof(s2), strlen(s2));
   printf("s3: \"%s\" size: %zu length: %zu\n", s3, sizeof(s3), strlen(s3));
}
```

Let's assume the storage for arrays s1, s2, and s3 is allocated in the order of declarations in the source code (high memory address to low memory address).

Step 1: Show the stack frame contents BEFORE call to `strcpy` function (left table on next page)

Step 2: Use the before call stack frame contents to determine the output produced by the first three print statements.

Step 3: Show the stack frame contents AFTER call to `strcpy` function (right table on next page)

Step 4: Use the after call stack frame contents to determine the output produced by the last three print statements.

Stack frame (before call to `strcpy`)

High address	RA (4 bytes)
CFP →	PFP (4 bytes)
s1[24]	
s1[0]	
s2[14]	
s2[0]	
s3[9]	
Low address s3[0]	

Stack frame (after call to `strcpy`):

High address	RA (4 bytes)
CFP →	PFP (4 bytes)
s1[24]	
s1[0]	
s2[14]	
s2[0]	
s3[9]	
Low address s3[0]	