Demonstration of safe string copying in C using `strncpy()`

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

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

   printf("Before strncpy 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));

   strncpy(s2,s1,sizeof(s2)-1);
   s2[sizeof(s2)-1] = '\0';      // manually null-terminate to be safe
   strncpy(s3,s1,sizeof(s3)-1);
   s3[sizeof(s3)-1] = '\0';

   printf("After strncpy 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 are 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 `strncpy` 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.

```
s1: "computer security is fun" size: 25 length: 24
s2: "" size: 15 length: 0
s3: "" size: 10 length: 0
```

Step 3: Show the stack frame contents AFTER call to `strncpy` 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.

```
s1: "computer security is fun" size: 25 length: 24
s2: "computer secur" size: 15 length: 14
s3: "computer " size: 10 length: 9
```

Stack frame (before call to `strncpy`)

High address	RA (4 bytes)
CFP →	PFP (4 bytes)
s1[24]	'\0'
	'n'
	'u'
	'f'
	' '
	's'
	'i'
	' '
	'y'
	't'
	'i'
	'r'
	'u'
	'c'
	'e'
	's'
	' '
	'r'
	'e'
	't'
	'u'
	'p'
	'm'
	'o'
s1[0]	'c'
s2[14]	
s2[0]	'\0'
s3[9]	
Low address s3[0]	'\0'

Stack frame (after call to `strncpy`):

High address	RA (4 bytes)
CFP →	PFP (4 bytes)
s1[24]	'\0'
	'n'
	'u'
	'f'
	' '
	's'
	'i'
	' '
	'y'
	't'
	'i'
	'r'
	'u'
	'c'
	'e'
	's'
	' '
	'r'
	'e'
	't'
	'u'
	'p'
	'm'
	'o'
s1[0]	'c'
s2[14]	'\0'
	'r'
	'u'
	'c'
	'e'
	's'
	' '
	'r'
	'e'
	't'
	'u'
	'p'
	'm'
	'o'
s2[0]	~~'\0'~~ 'c'
s3[9]	'\0'
	' '
	'r'
	'e'
	't'
	'u'
	'p'
	'm'
	'o'
Low address s3[0]	~~'\0'~~ 'c'