Ilansh
302514401
Ilan Shamir

Part 1
a. The error is that there are 2 stars in front of the first delimiter: the correct code would be:
*(*str + strlen(*str) – 1))
The reason for this is that the initial code looked at the address str which is a pointer to the address of the real string.
Performing the operation (str + strlen(*str) -1) has no meaning, since str is not the address of the first letter of the string, but a pointer to that address.
*str however, is a pointer to the first letter of the string.

b. - foo1 - will compile and run properly. The declaration (int * const arr) protectes the pointer to the array and not the value in the pointed location, thus the change ++(*arr) is legal, since it is applied to the value which arr points to (*arr) and not the address arr.
- foo2 - will not compile, because the change ++arr is done on the address and not the value, the conditions are the same as in foo1 – the address is protected and thus a compilation error occurs.
- foo3 - The code will compile but produce a warning, since we "down-cast" arr intro a non-const pointer p. The integer pointed by arr is protected, and p is a non-const pointer to int (doesn't protect the integer value), so it can change values inside the array pointed by arr, even though arr protects the integer it points to.
-foo4 The code will not compile for similar reasons to foo1. The const declaration protects the int value pointed by arr, and a change is made to this protected int value – error.

c. We should add the line assert(arrLen == sizeof(myDigits->cArray) / sizeof(myDigits->cArray[0]) right after the assignment into arrLen (similar in our case to assert(arrLen == 10).
The problem is that sizeof(MyDigits) isn't necessarily the exact sum of its components sizes, rather it is spread over a batch of memory in a certain way (fields are aligned into 4 byte segments), thus the deduction of the 2 int values will not give us the size of the remaining char array.

d. running my program using valgrind and Test1.in from the ex3 forum:

```
valgrind --leak-check=full PlayBoard out112 < Test1.in

==14014==
==14014== HEAP SUMMARY:
==14014==     in use at exit: 0 bytes in 0 blocks
==14014==   total heap usage: 9 allocs, 9 frees, 1,640 bytes allocated
==14014==
==14014== All heap blocks were freed -- no leaks are possible
==14014==
==14014== For counts of detected and suppressed errors, rerun with: -v
==14014== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 4 from 4)
```