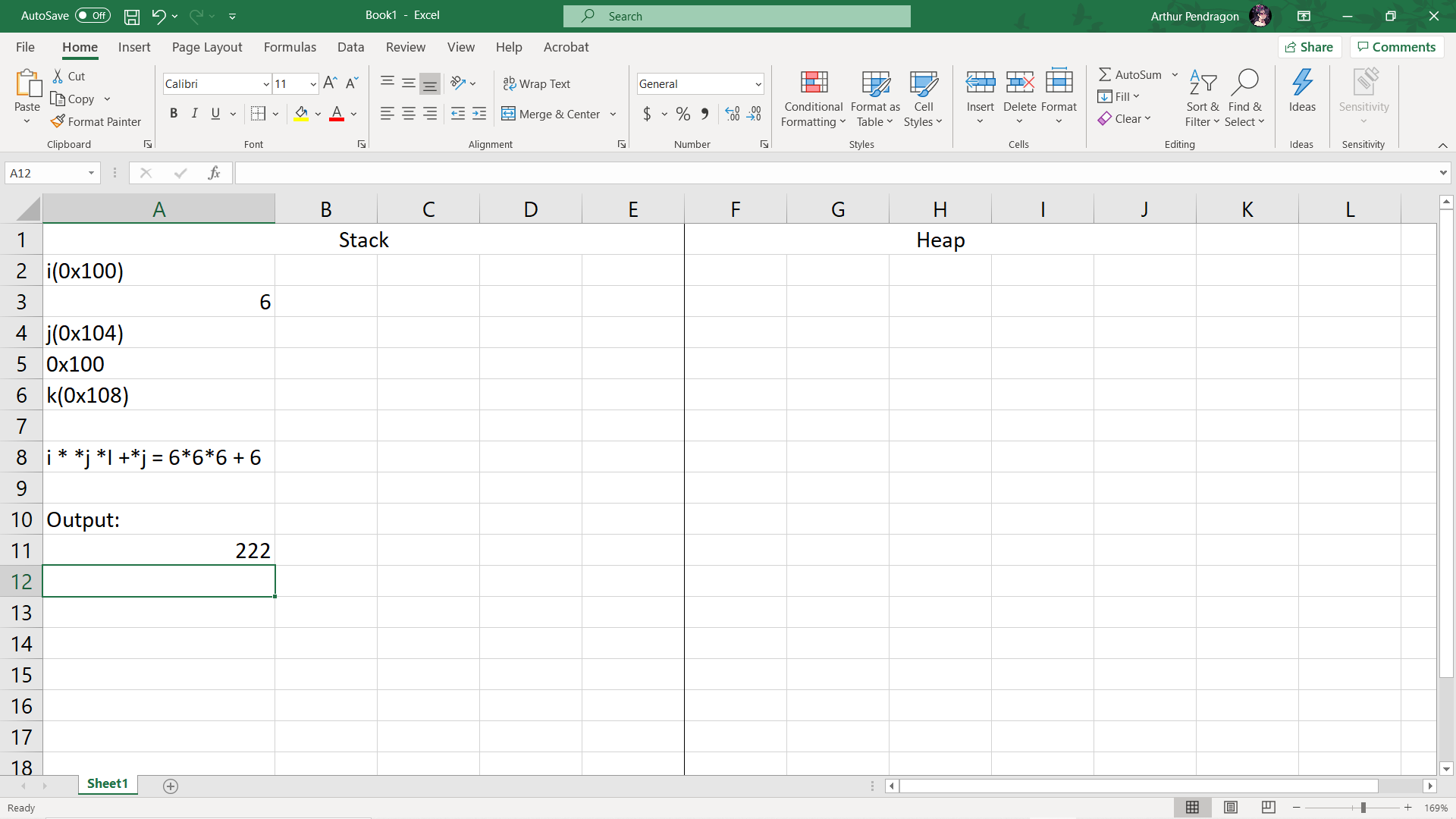
Thưa thầy Hồ Tuấn Thanh, do Excel không hiển thị ‘a’ mà nó chỉ hiển thị a’ nên em sẽ thay thế tất cả các dấu nháy đơn thành nháy kép “ ” hết ạ.

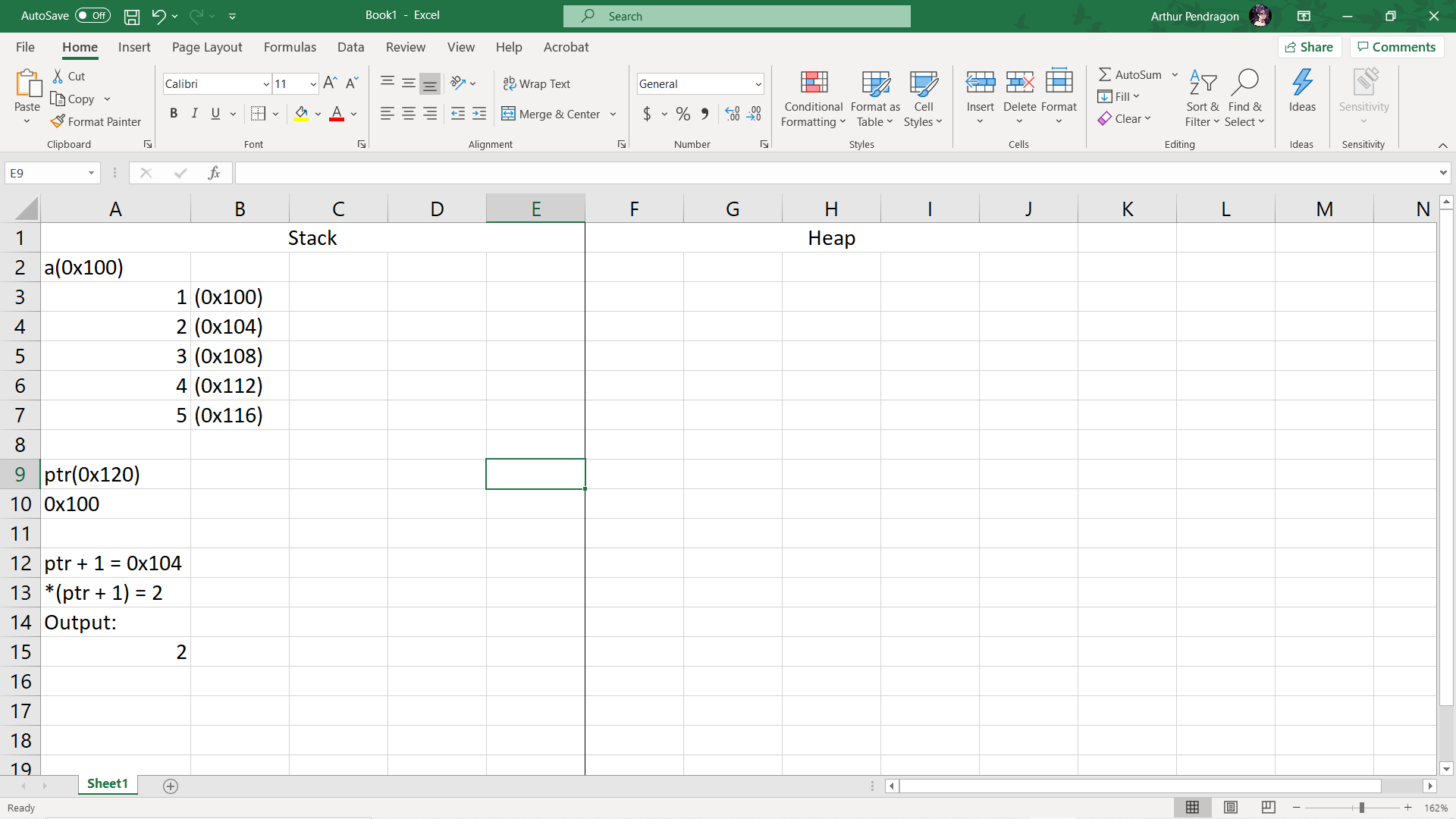
1. **ASSIGNMENT 37:**

Output: 222 (Explain: i \* \*j \* i + \*j = 6\*6\*6+6 = 222)



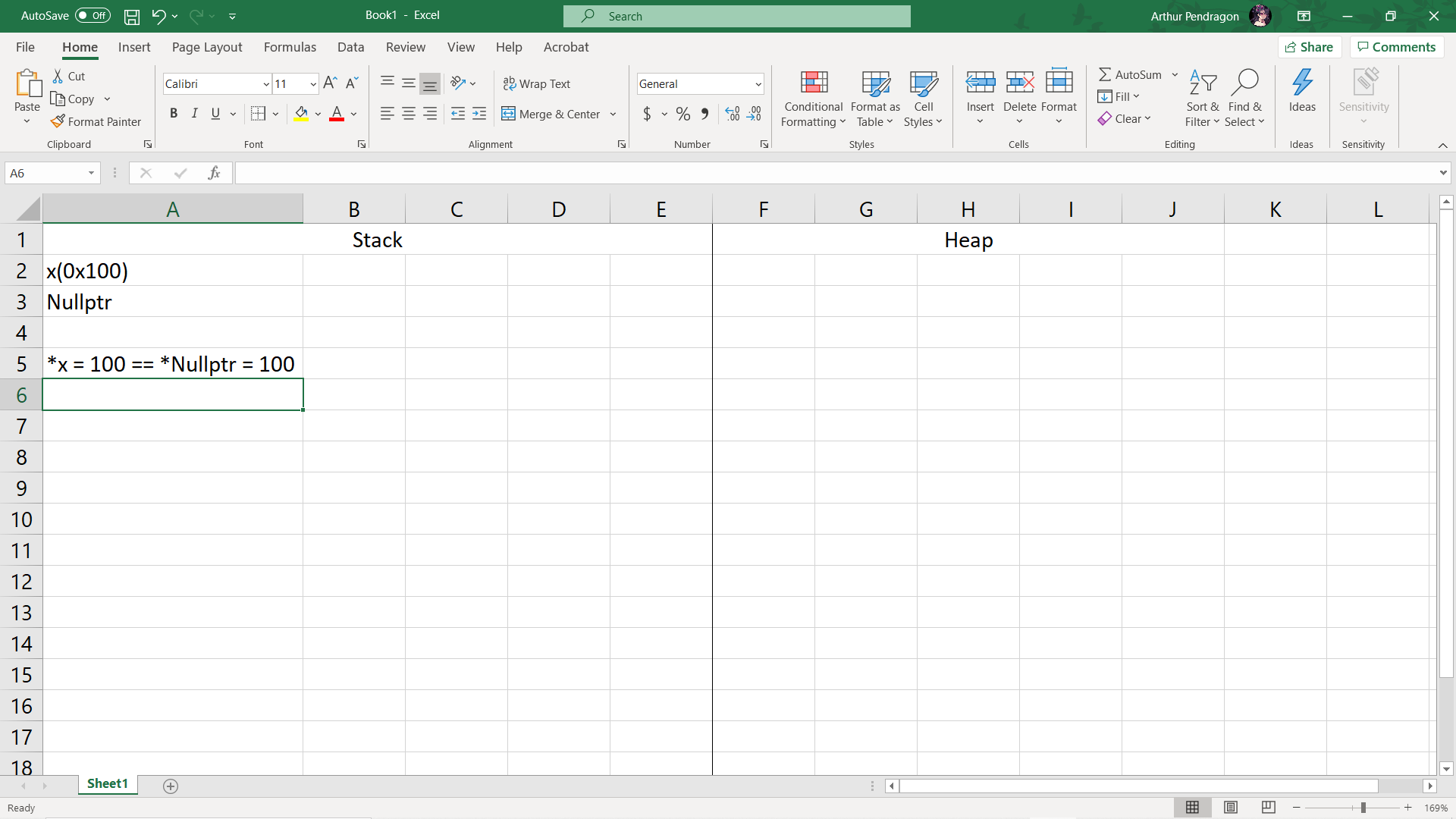
1. **ASSIGNMENT 35:**

Output: 2 (Explain: ptr = a means ptr = a[0], \*(ptr + 1) means the value of a[0 + 1] = a[1] = 2)



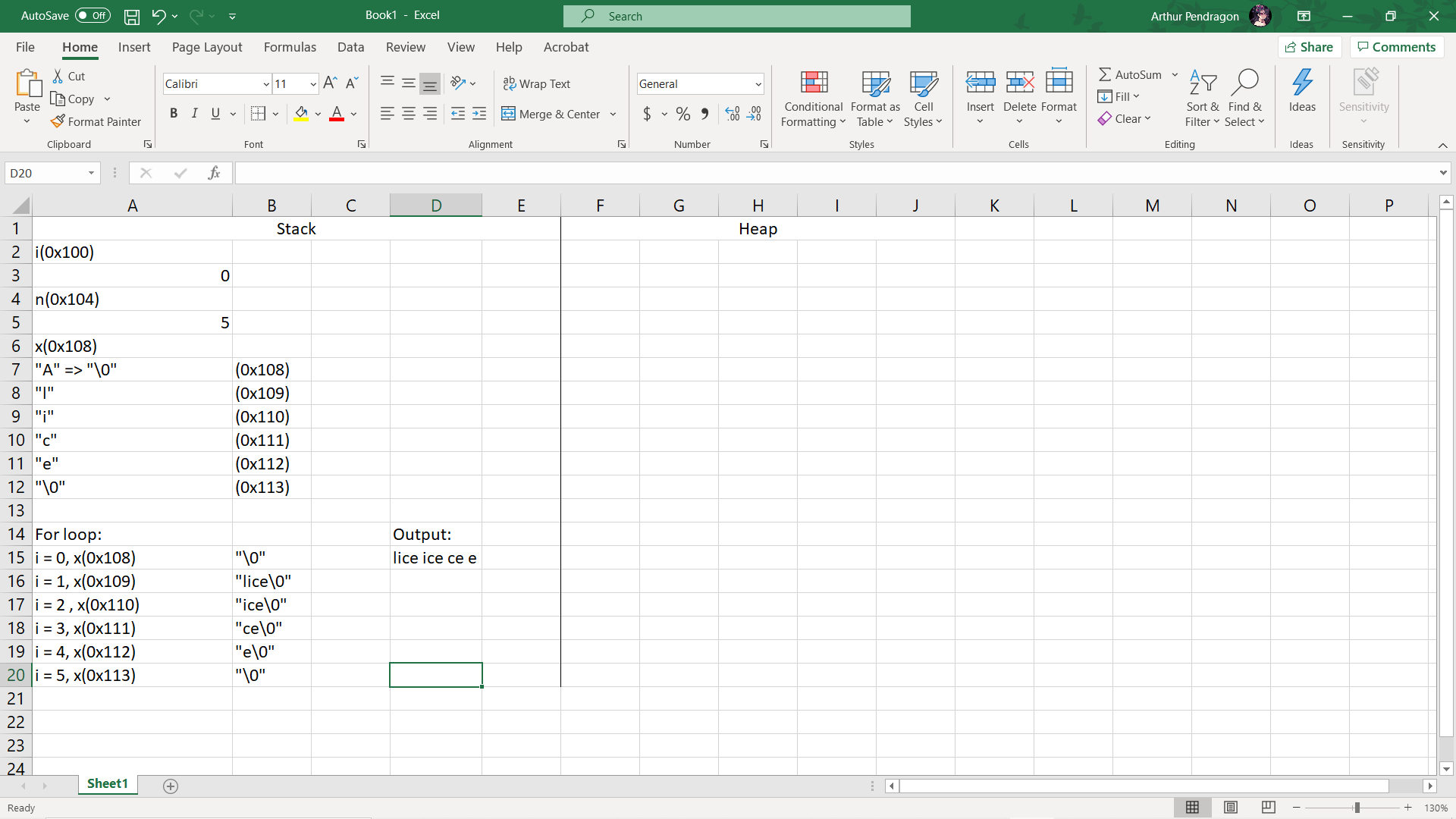
1. **ASSIGNMENT 7:**

C. No error. (Explain: While reading the code there is no error, but upon running the program having an uninitialized variable can cause the program to crash (Null pointer assignment))



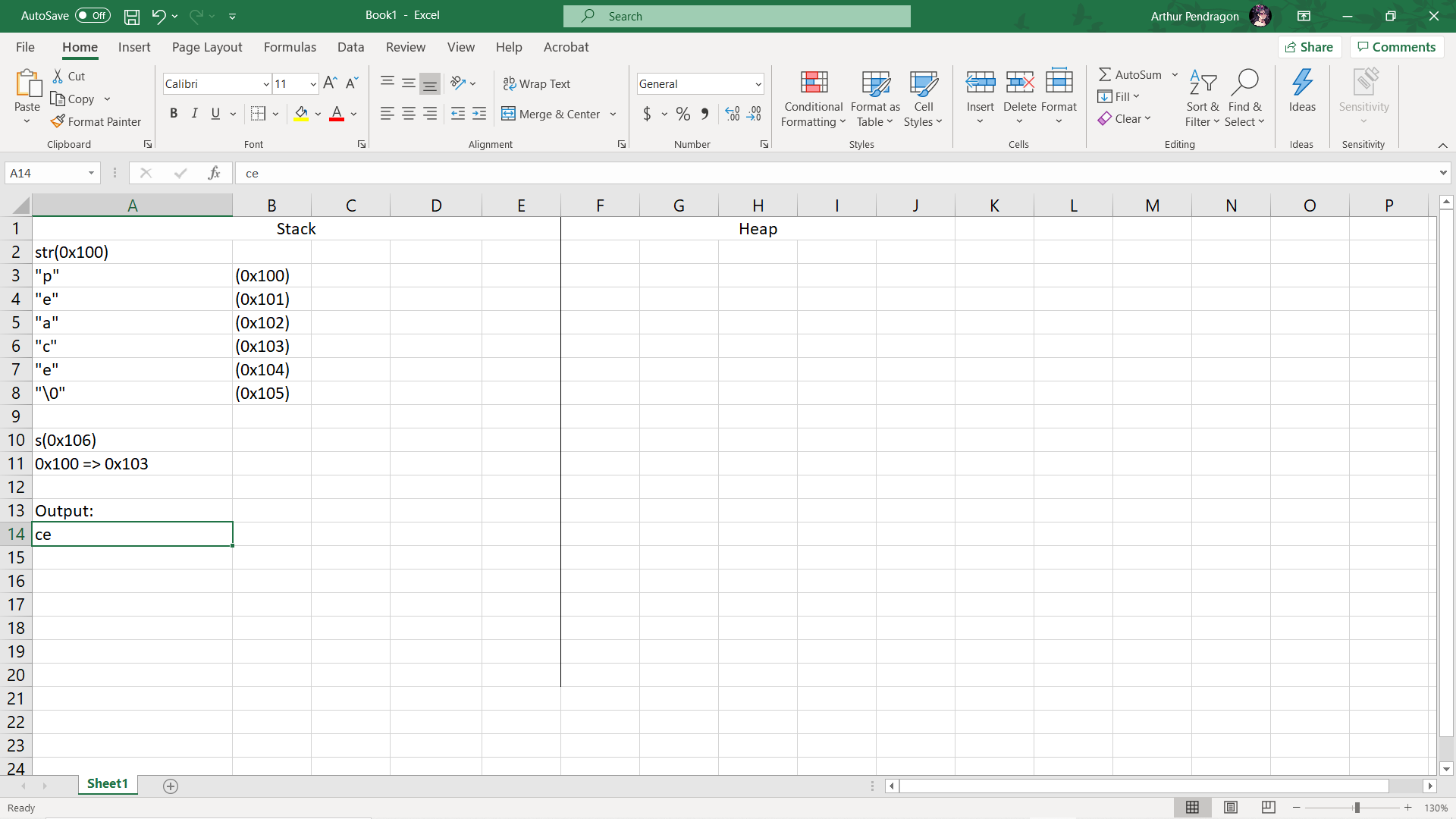
1. **ASSIGNMENT 8:**

D. lice ice ce e



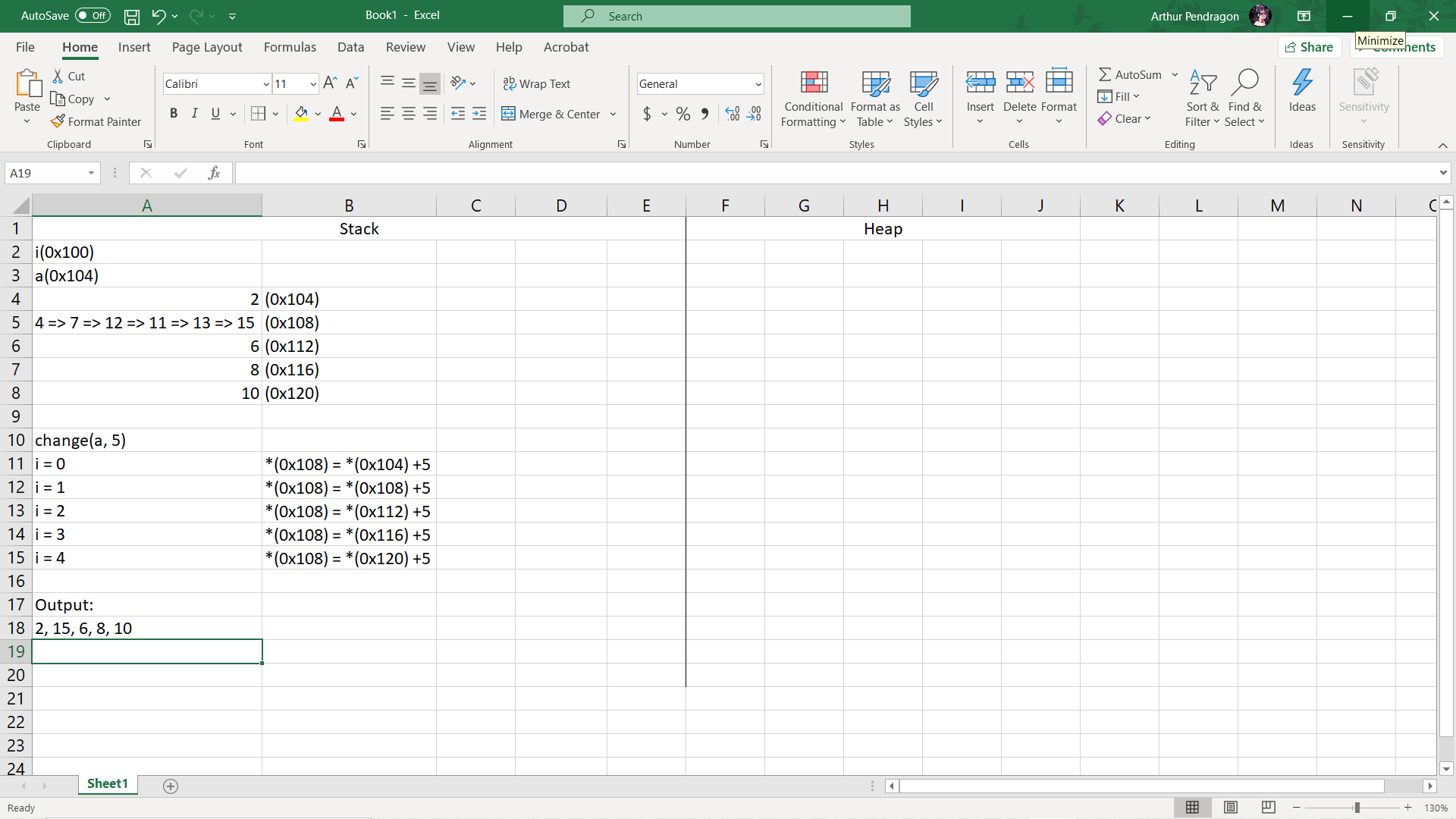
1. **ASSIGNMENT 9:**

D. ce (Explain: Increment the pointer’s value by 3 times the size of a char because s++ is postfix-increment so it doesn’t affect)



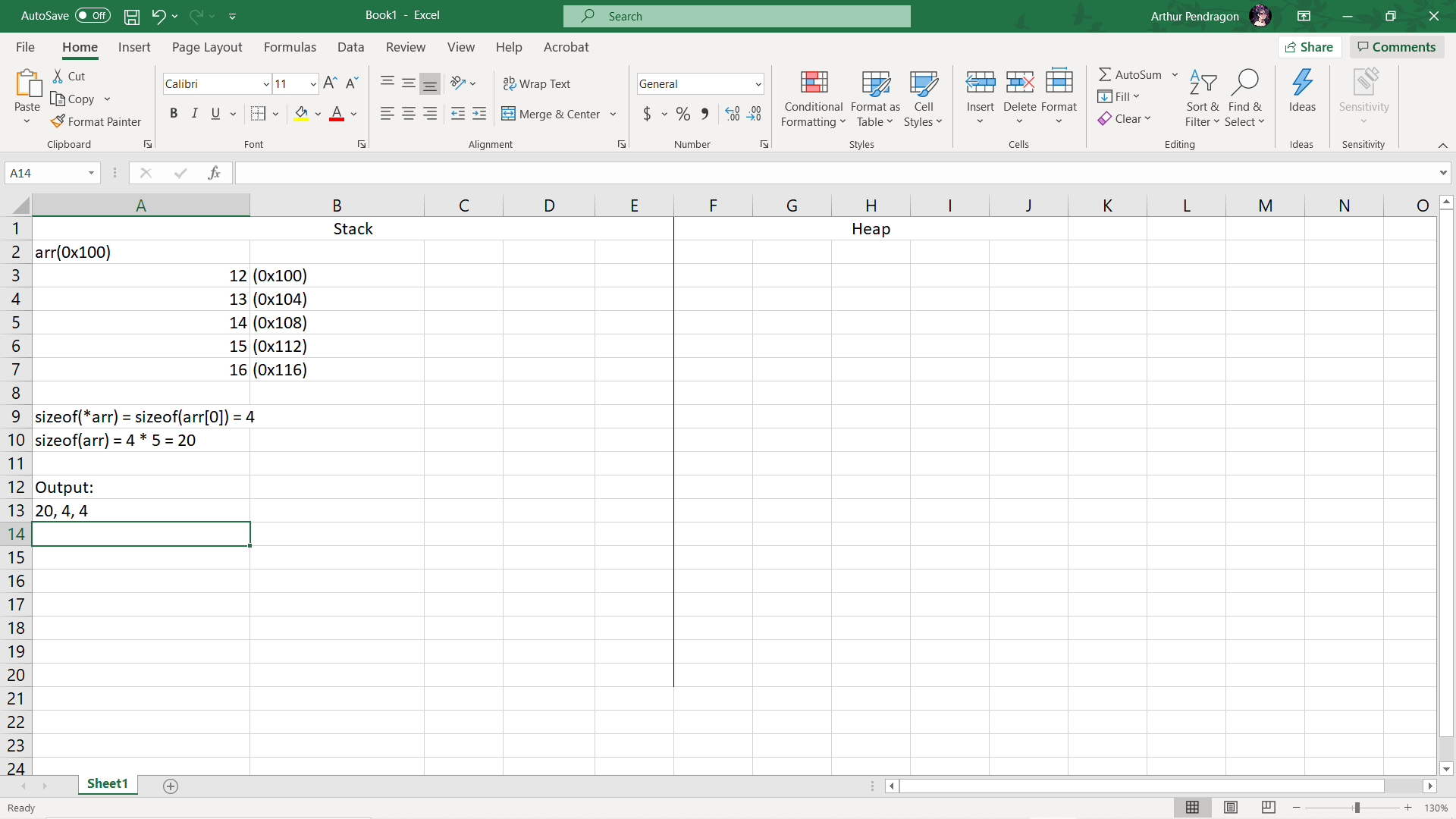
1. **ASSIGNMENT 10:**

B. 2, 15, 6, 8, 10 (Explain: The change function will only do a[1] = a[4] + 5 = 15)



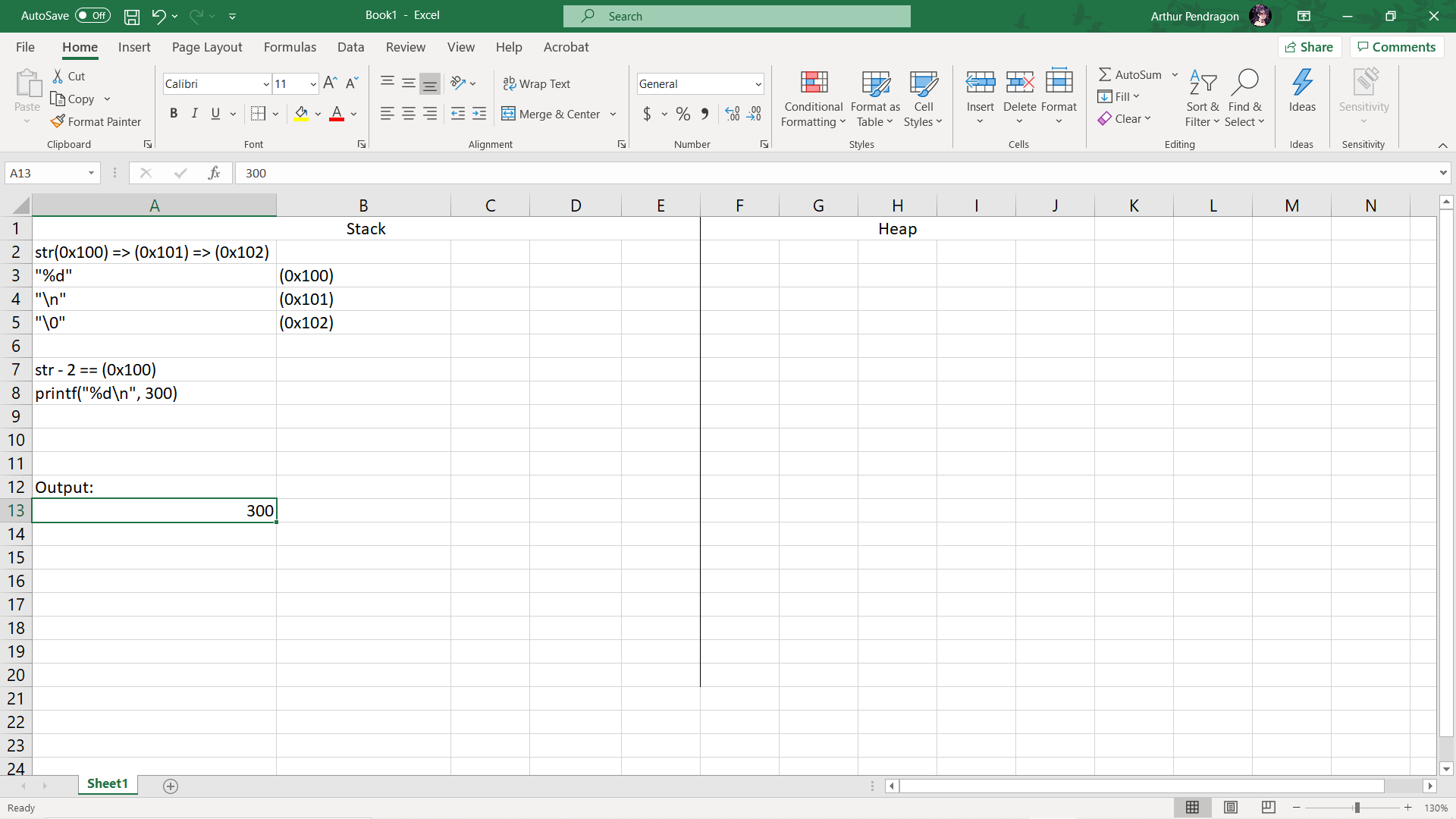
1. **ASSIGNMENT 11:**

B. 20, 4, 4 (Explain: sizeof(arr) = 4\*5 bytes, \*arr is the same as arr[0] so their size are all 4 bytes)



1. **ASSIGNMENT 12:**

D. 300 (Explain: 2 times str++ and str – 2 canceled each other)



1. **ASSIGNMENT 13:**

A. \*

1. **ASSIGNMENT 14:**

A. x is a pointer to a string, y is a string. (Explain: the \* only has effect on the first variable)

1. **ASSIGNMENT 15:**

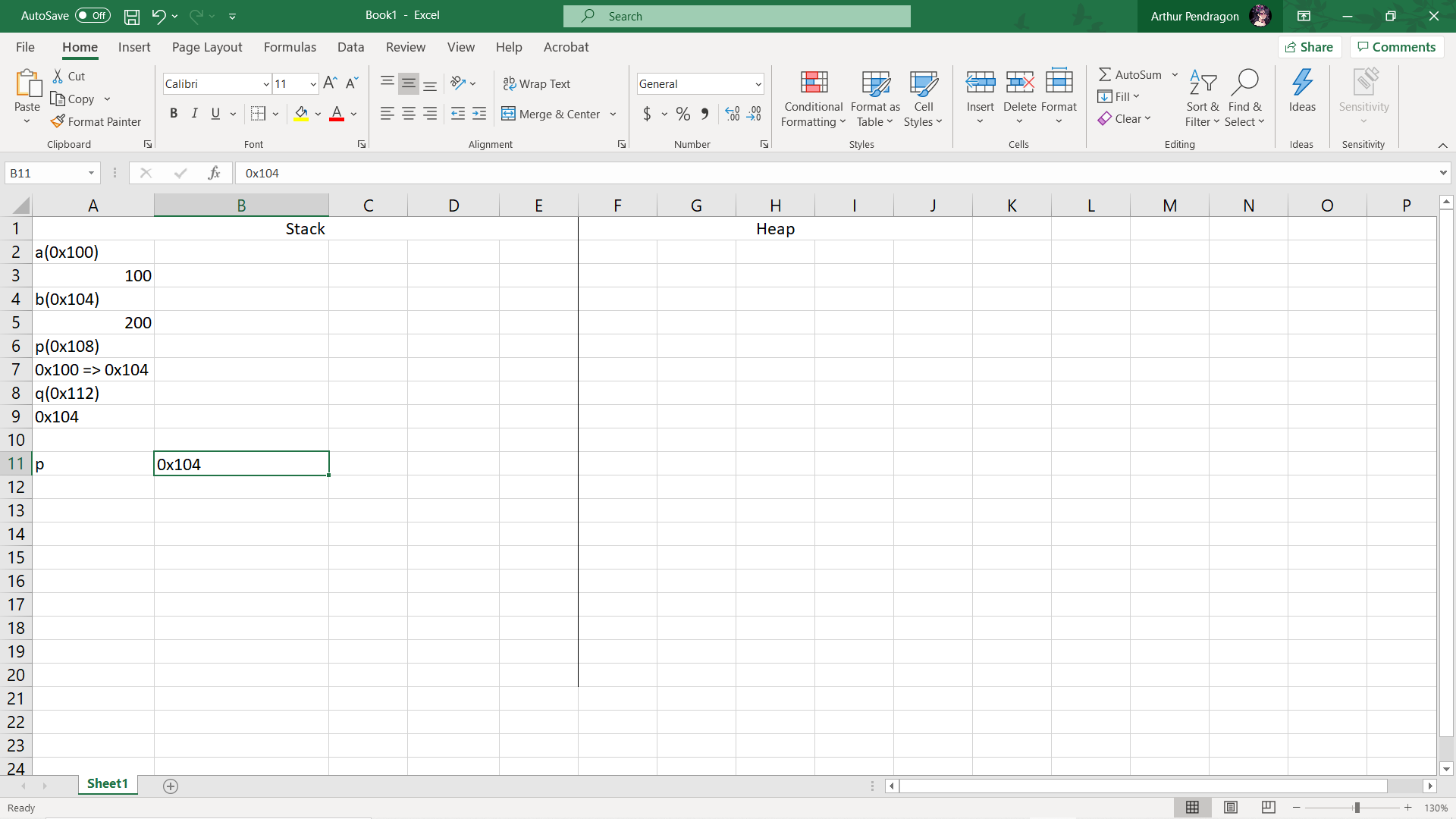
D. point to type (Explain: a pointer can only have 3 states a, b and c)

1. **ASSIGNMENT 16:**

C. int i; double\* dp = &i; (Explain: dp is a pointer of type double so it can not point to the memory address of an integer type variable)

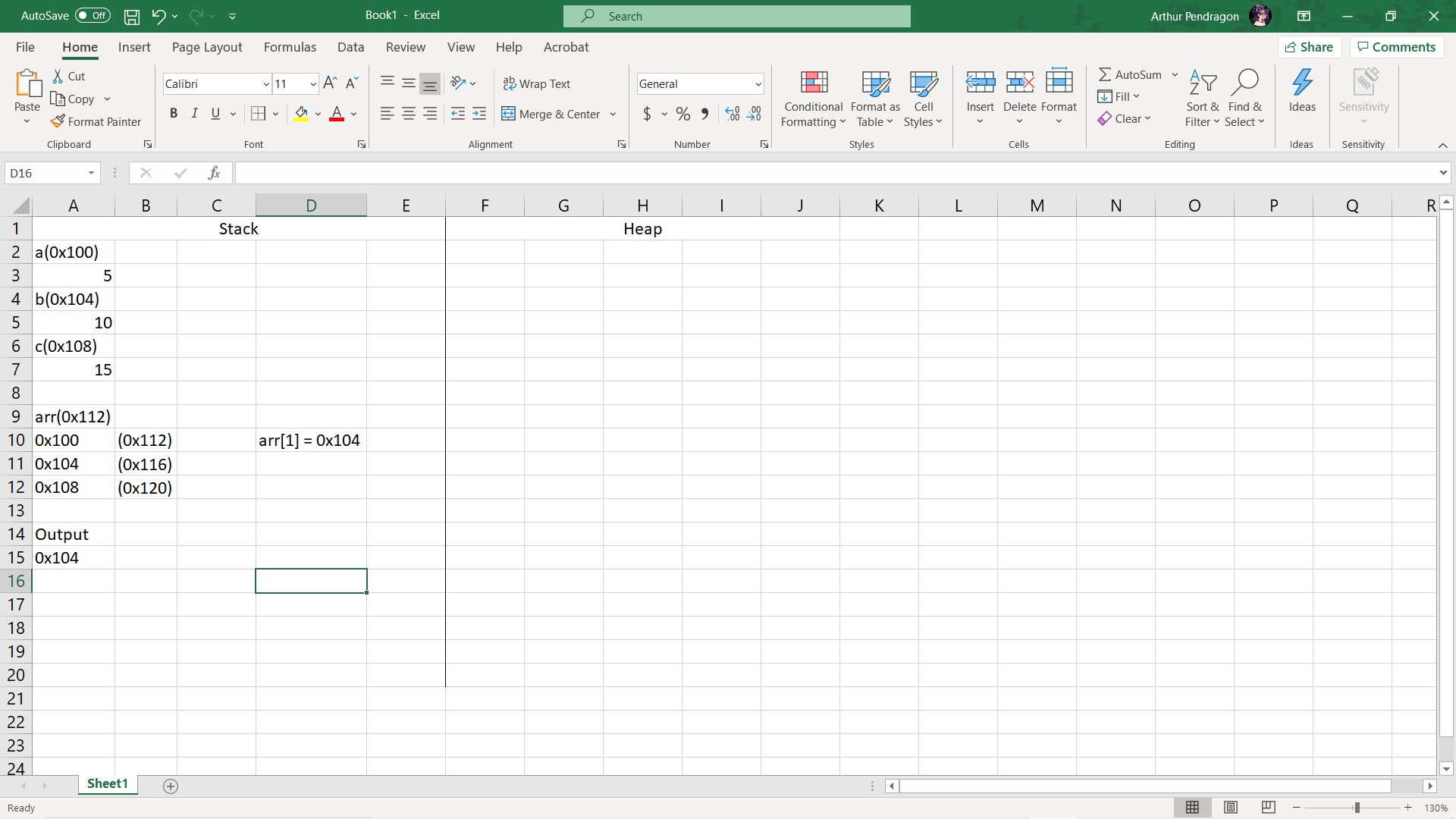
1. **ASSIGNMENT 17:**

B. p now points to b. (Explain: q is a pointer of b. Hence p = q means p now is also a pointer of b)



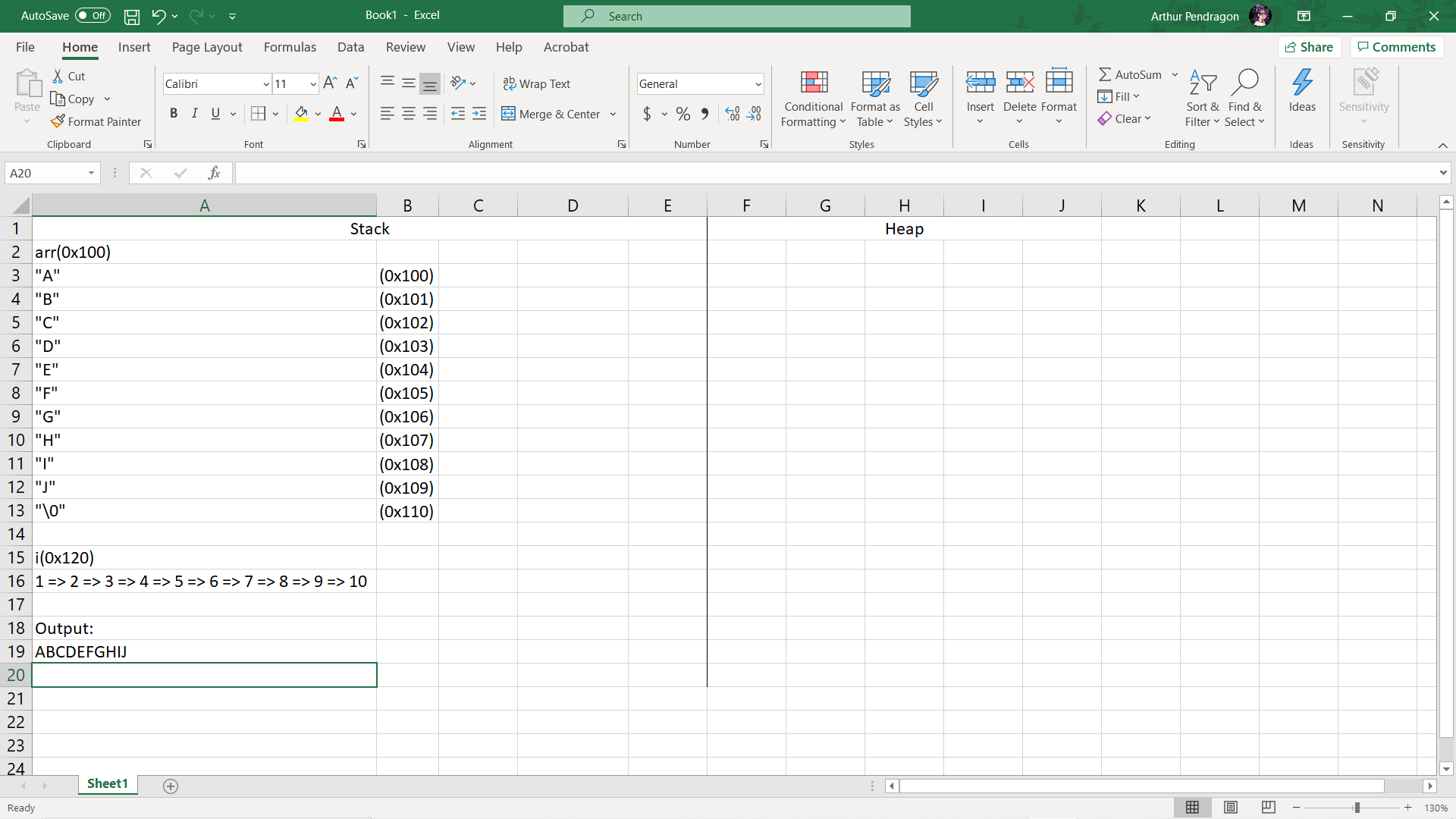
1. **ASSIGNMENT 18:**

D. it will return some random number. (Explain: the return value is the memory address of integer b)



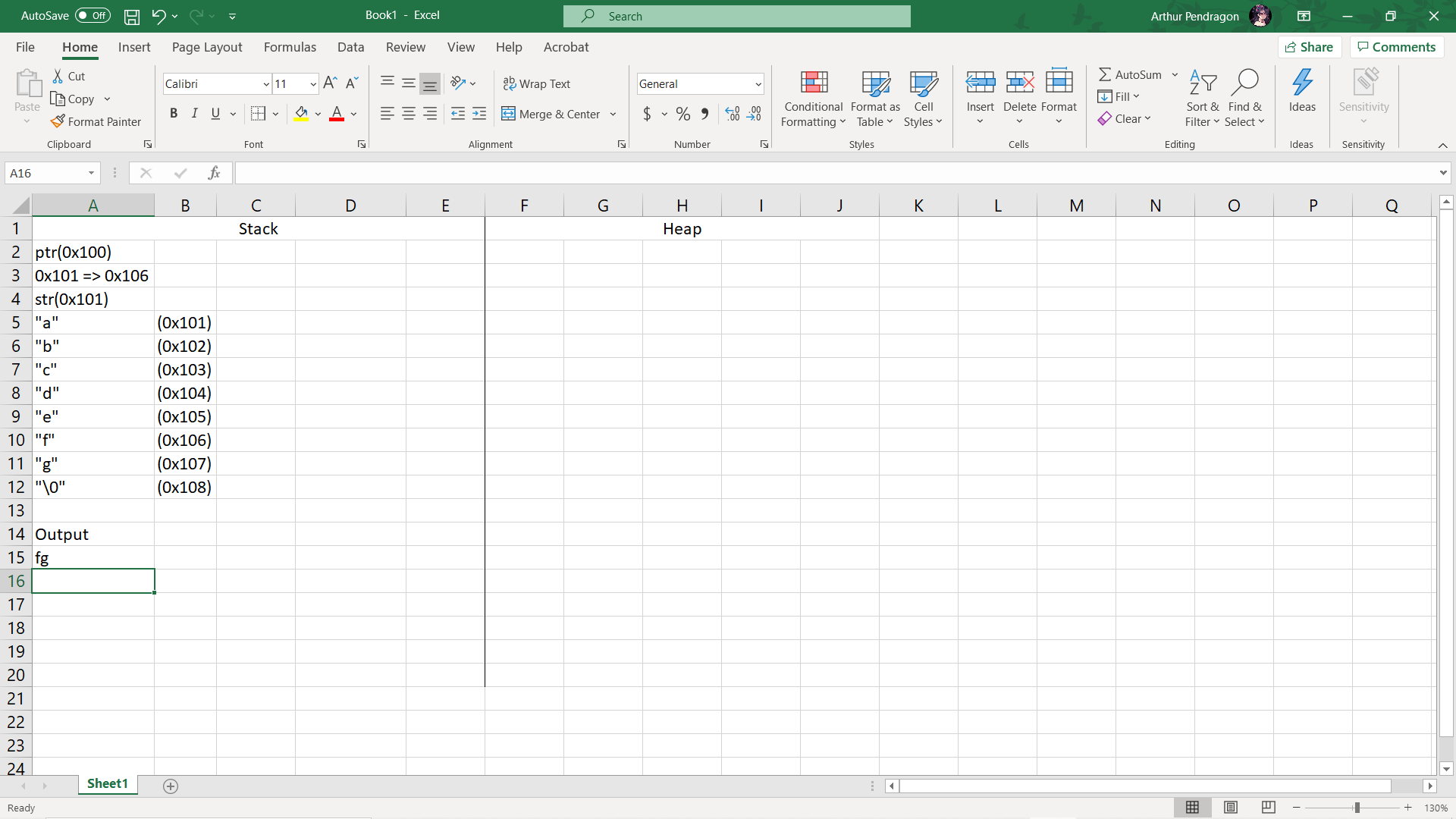
1. **ASSIGNMENT 19:**

A. ABCDEFGHIJ (Explain: first 10 values of the arr array is assigned by the ASCII values from 65 to 74 which is letter A to letter J)



1. **ASSIGNMENT 20:**

A. fg (Explain: ptr = str means ptr points to the address of str[0], ptr + 5 ptr points to the address of str[5])



1. **ASSIGNMENT 21:**

D. All of them. (Explain:

int\* ptr, a = 1;

ptr = 0, ptr = nullptr, ptr = &a are all ok)

1. **ASSIGNMENT 22:**

C. Constant pointer to non-constant data. (Explain: Error invalid conversion)

1. **ASSIGNMENT 23:**

B. const (Explain: constant variables can’t be modified)

1. **ASSIGNMENT 24:**

C. The new operator (Explain: this operator will have the OS provides us blocks of unallocated memory in the heap memory and returns the their address)

1. **ASSIGNMENT 25:**

B. Indirection (Explain: The process is called dereferencing or called indirection)

1. **ASSIGNMENT 26:**

A. sizeof

1. **ASSIGNMENT 27:**

A. Pointer contains an address of a variable.

1. **ASSIGNMENT 28:**

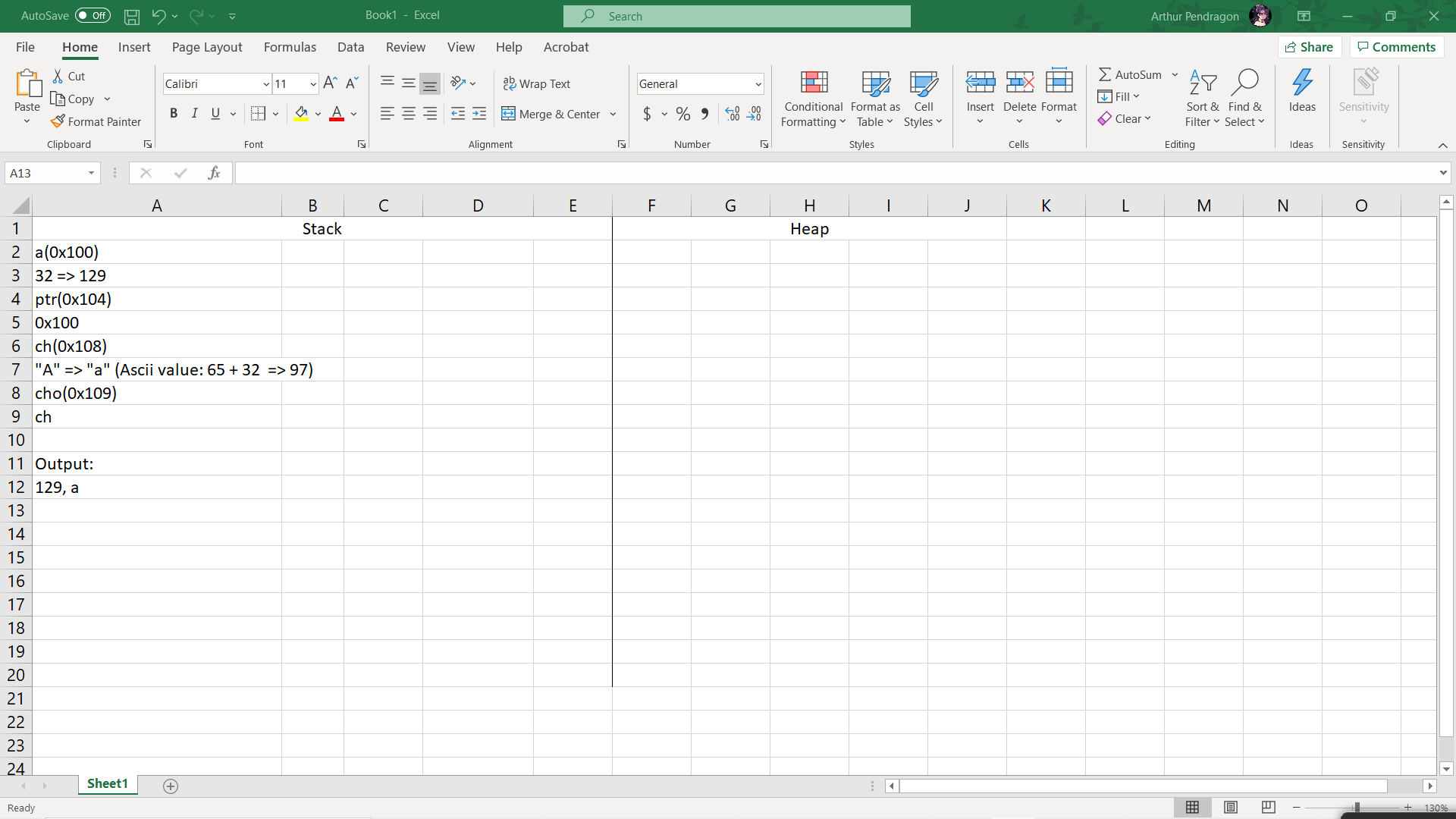
C. 3 (Explain: memory address of a variable, zero or null pointer)

1. **ASSIGNMENT 29:**

C. Address operator (Explain: &)

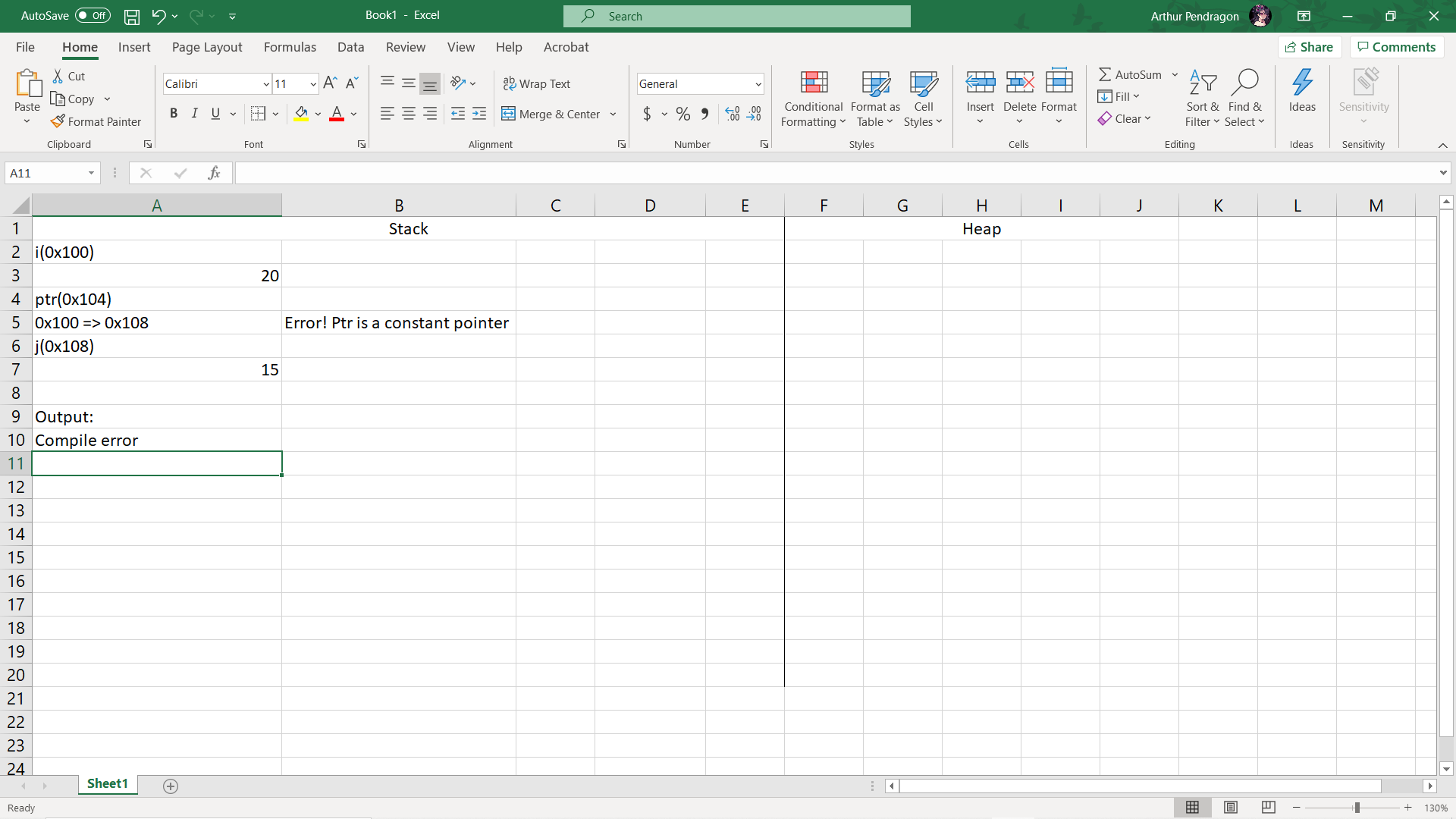
1. **ASSIGNMENT 30:**

C. 129, a (Explain: &cho refers to the ascii value of char ch: letter ‘A’ = 65. cho += a means 65 + 32 = 97 and 97 is the ascii value of letter ‘a’. Pointer \*ptr is a pointer of integer a. \*ptr += ch means adding the ascii value of ch which is now 97 to integer a)



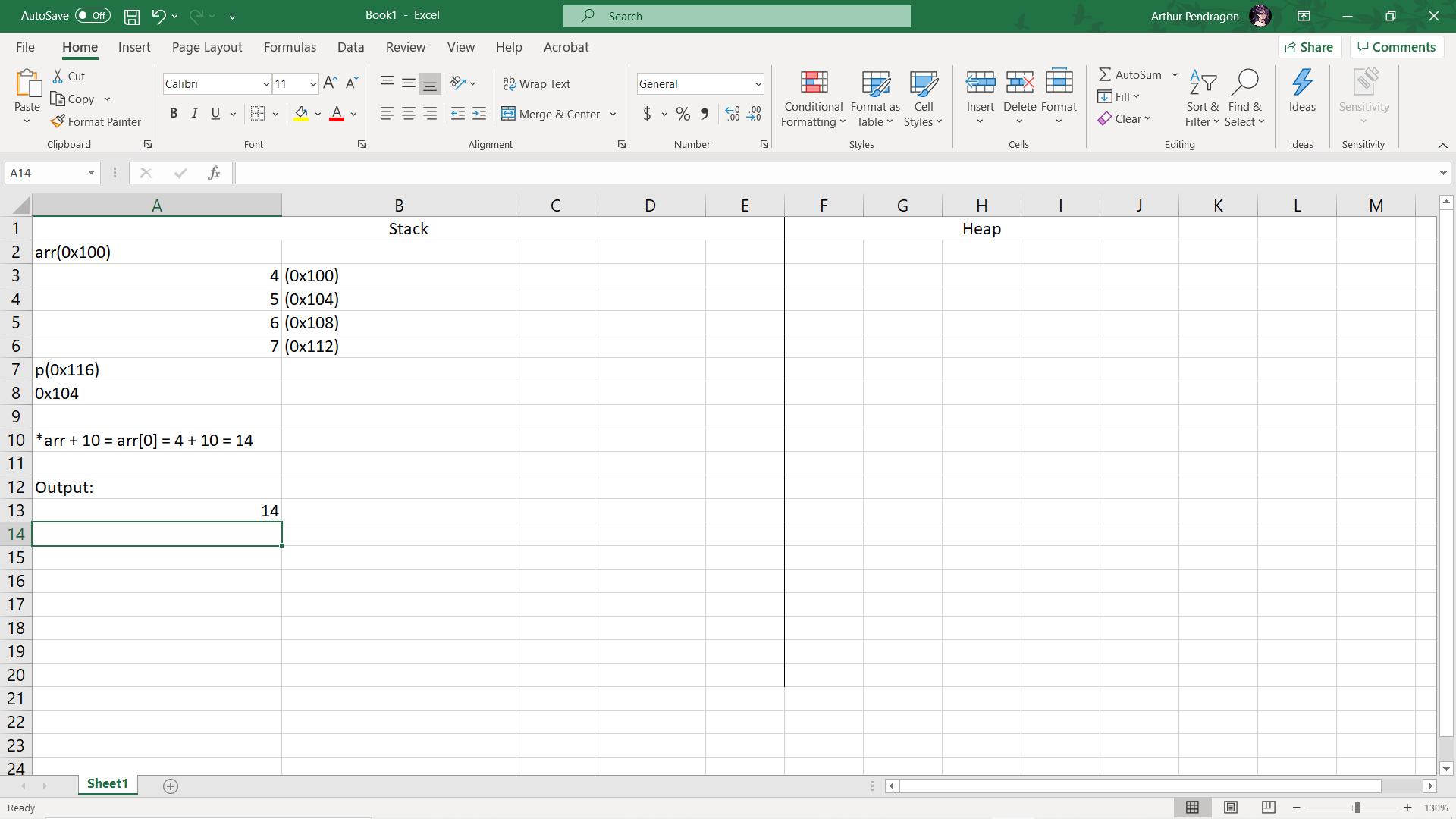
1. **ASSIGNMENT 31:**

D. Compile error (Explain: i is an constant integer so increments it via pointer ptr is impossible)



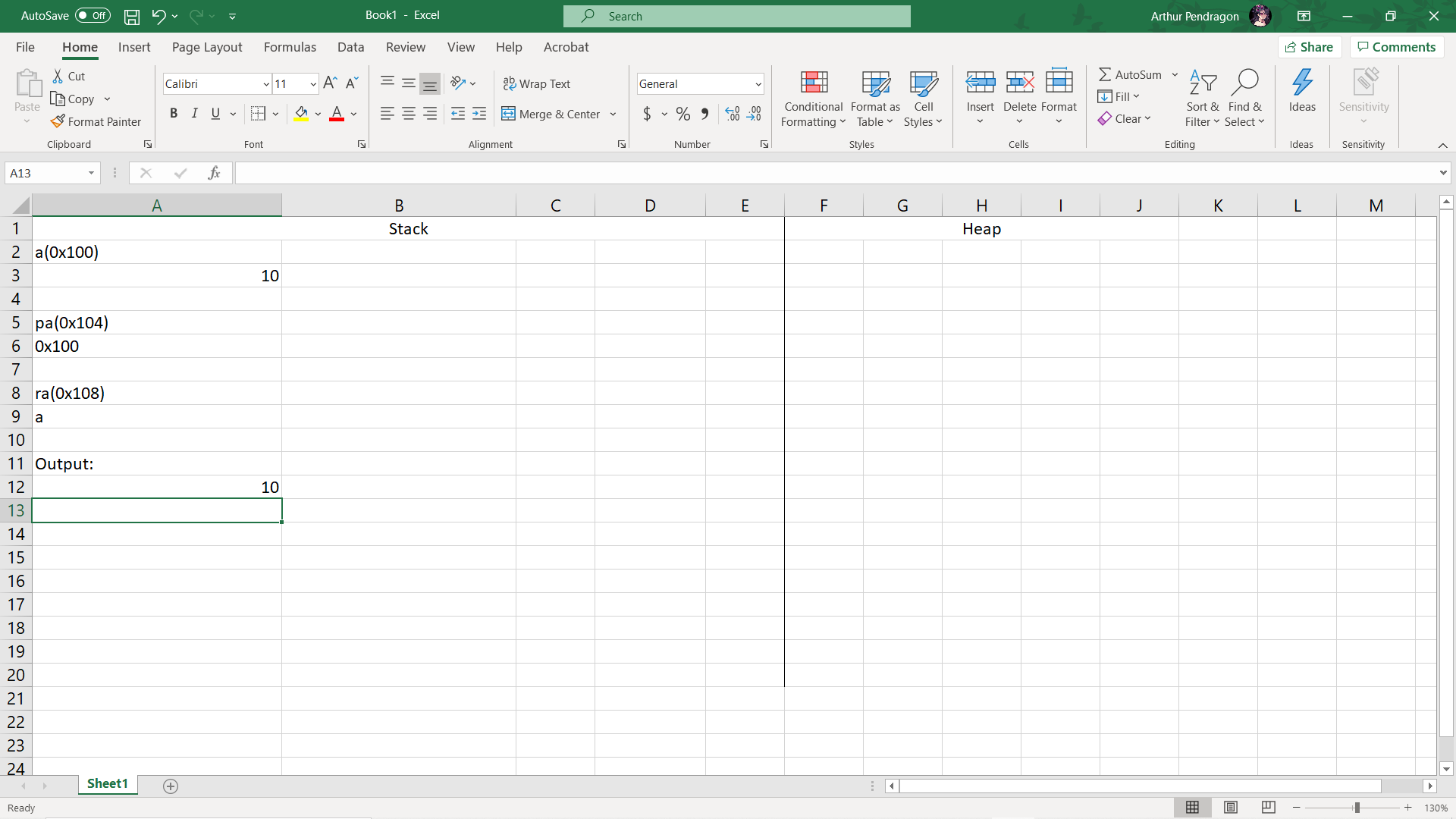
1. **ASSIGNMENT 33:**

C. 14 (Explain: \*arr refers to arr[0] = 4 so the output is 4 + 10 = 14)



1. **ASSIGNMENT 34:**

A. 10 (Explain: &ra = a means variable ra is a reference to integer a so ra’s value and address are all the same as integer a)



1. **ASSIGNMENT 36:**

Output: 15 (Explain: ptr is a pointer of integer a, \*ptr = \*ptr \*3 means a\*3 = 15)

