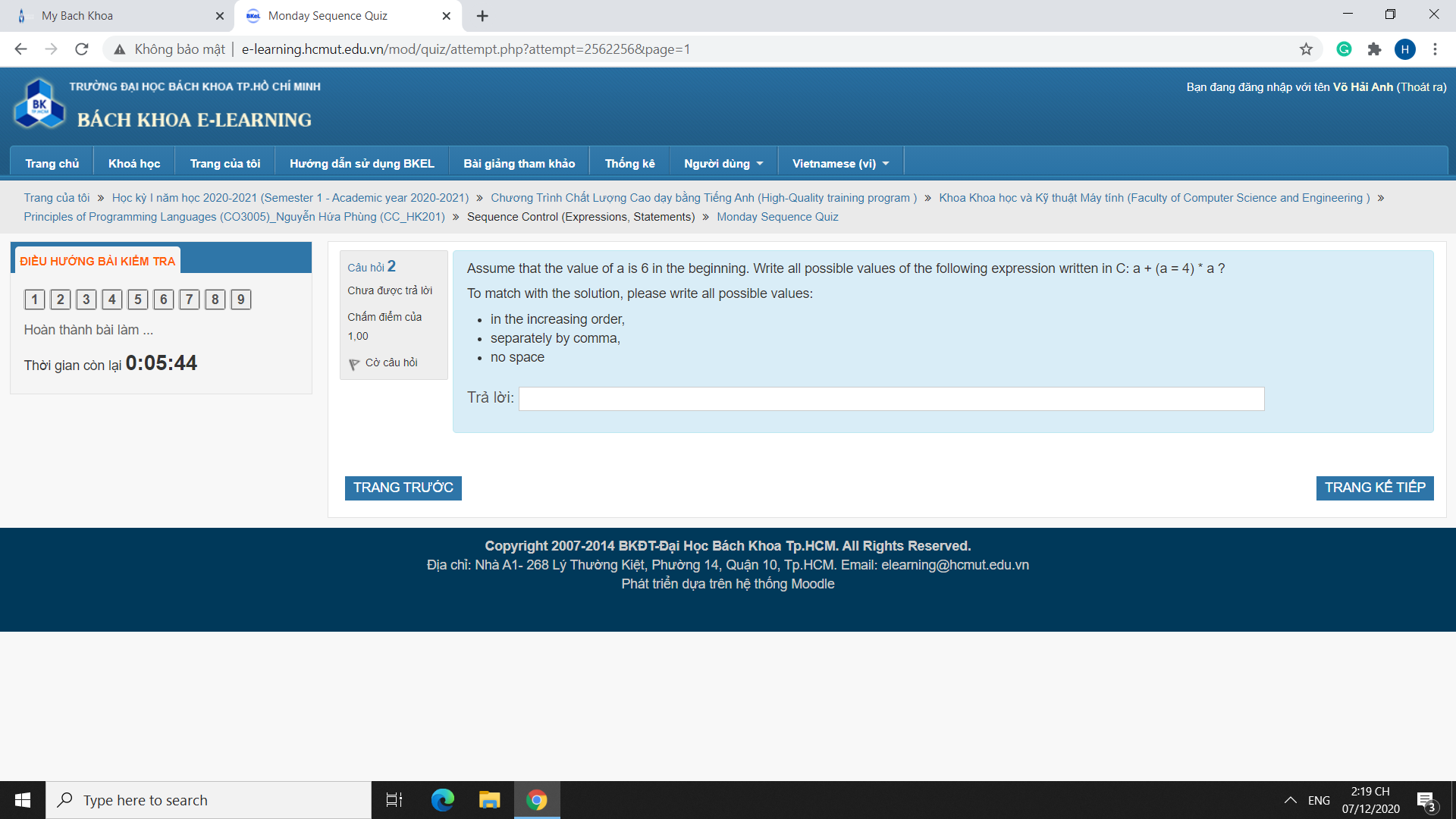


Address

A (video says so)

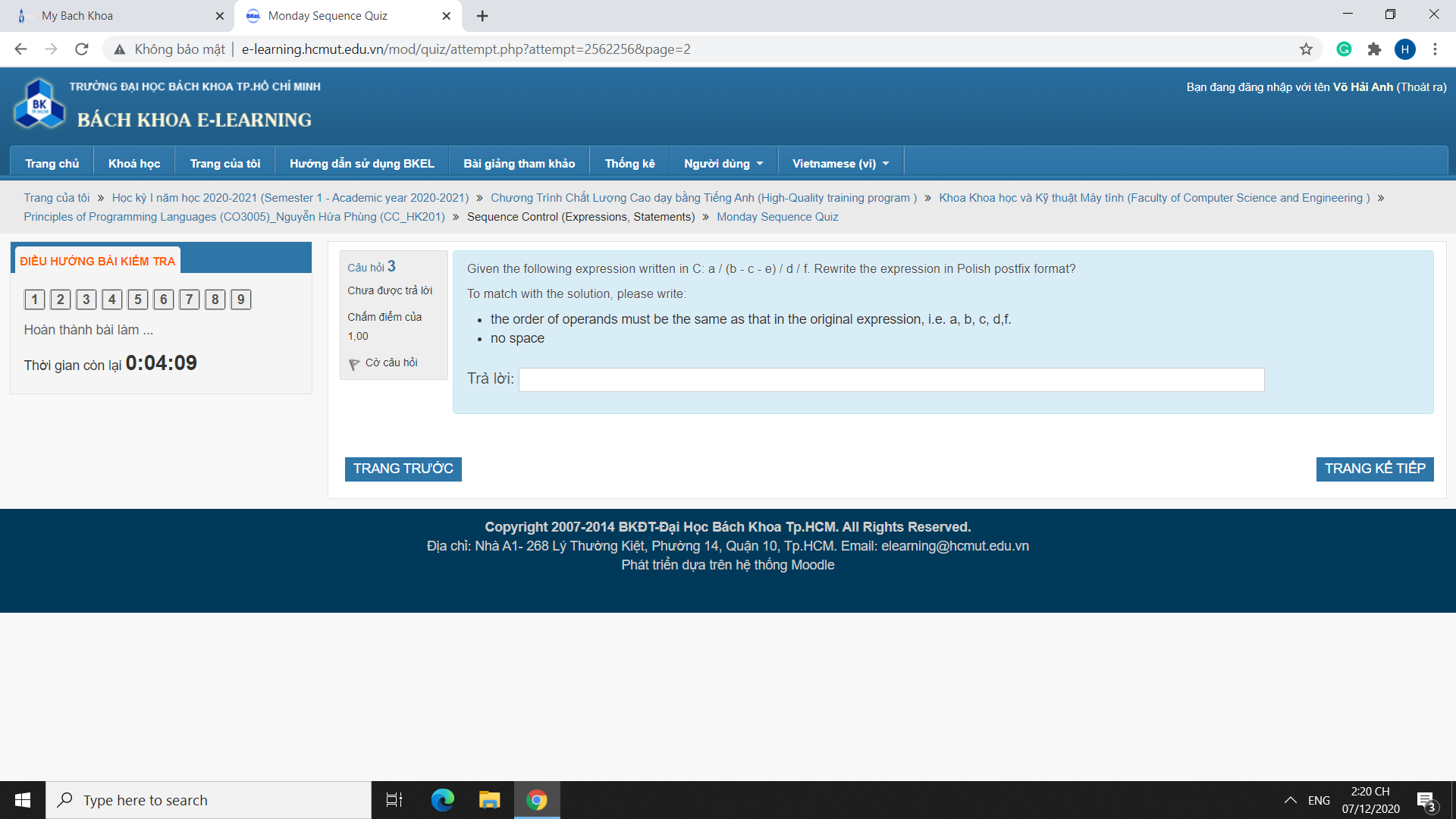


Expression a + (a = 4) \* a.

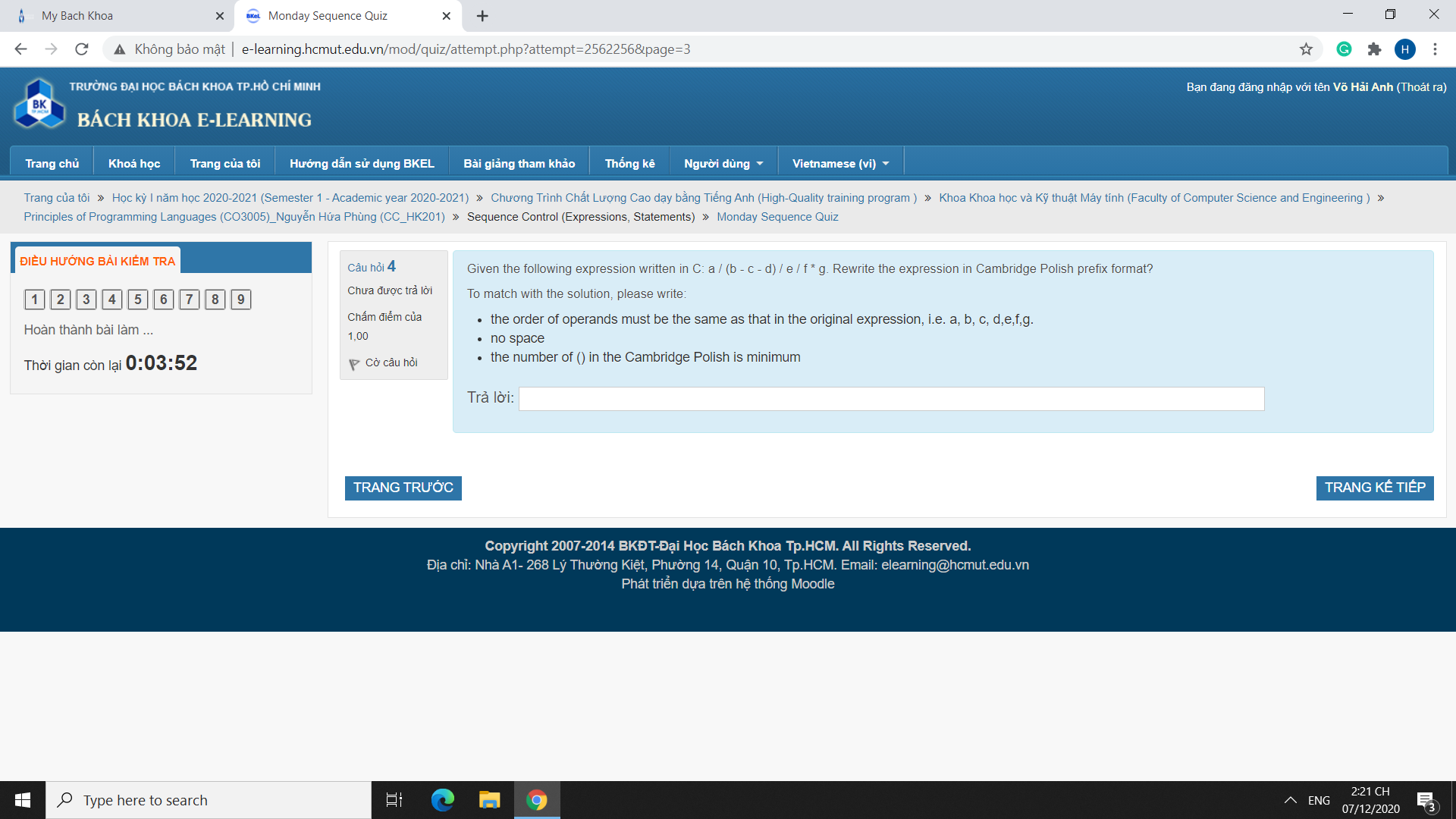
Orders:

* now 4 \* 4 + 4 = 20 (\*L, \*R, +L)
* 6 + now 4 \* 4 = 22 (+L, \*L, \*R)
* 6 \* now 4 + 4 = 28 (\*R, \*L, +L)
* 6 + 6 \* now 4 = 30 (+L, \*R, \*L)

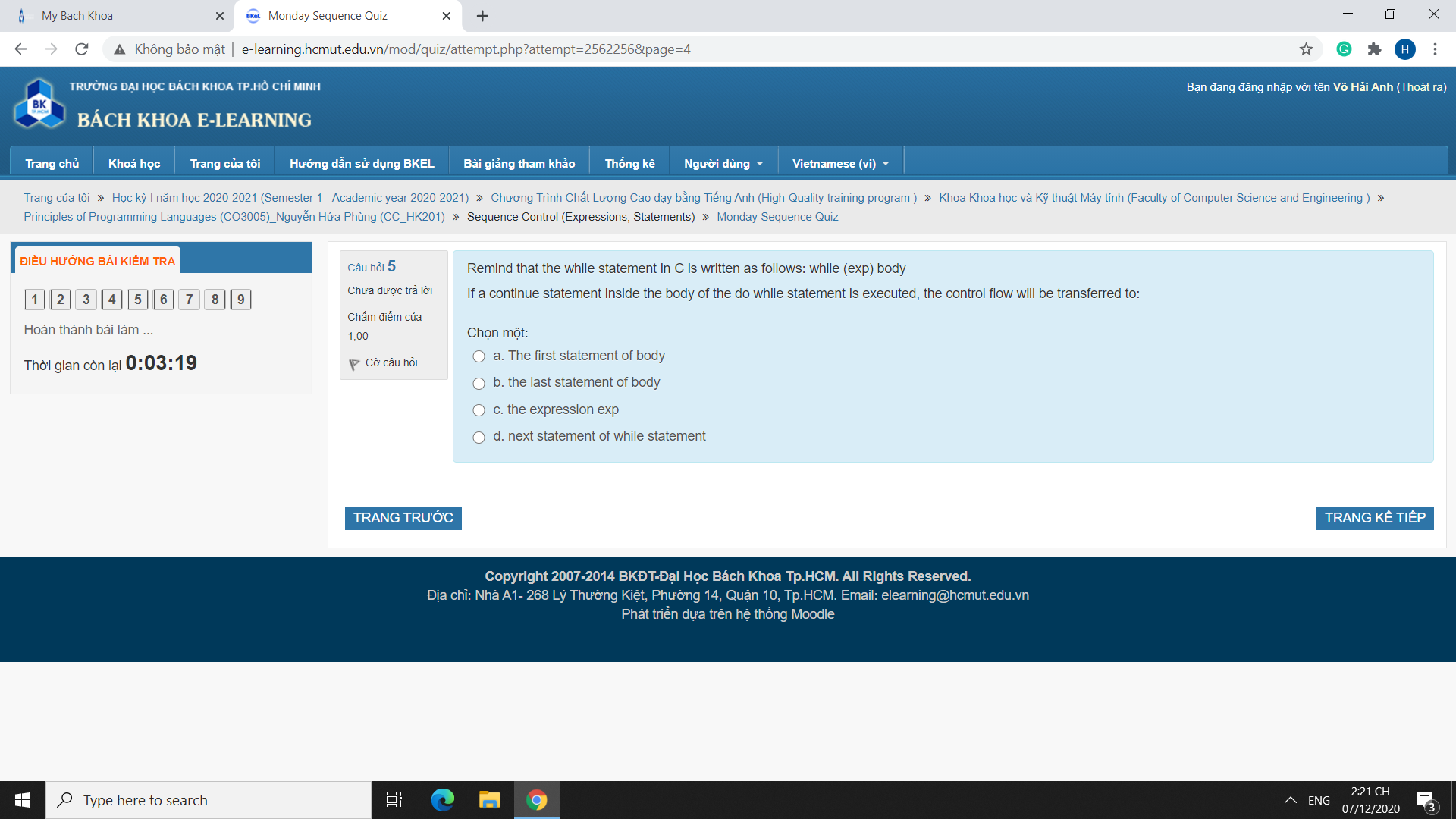
Answer: 20,22,28,30



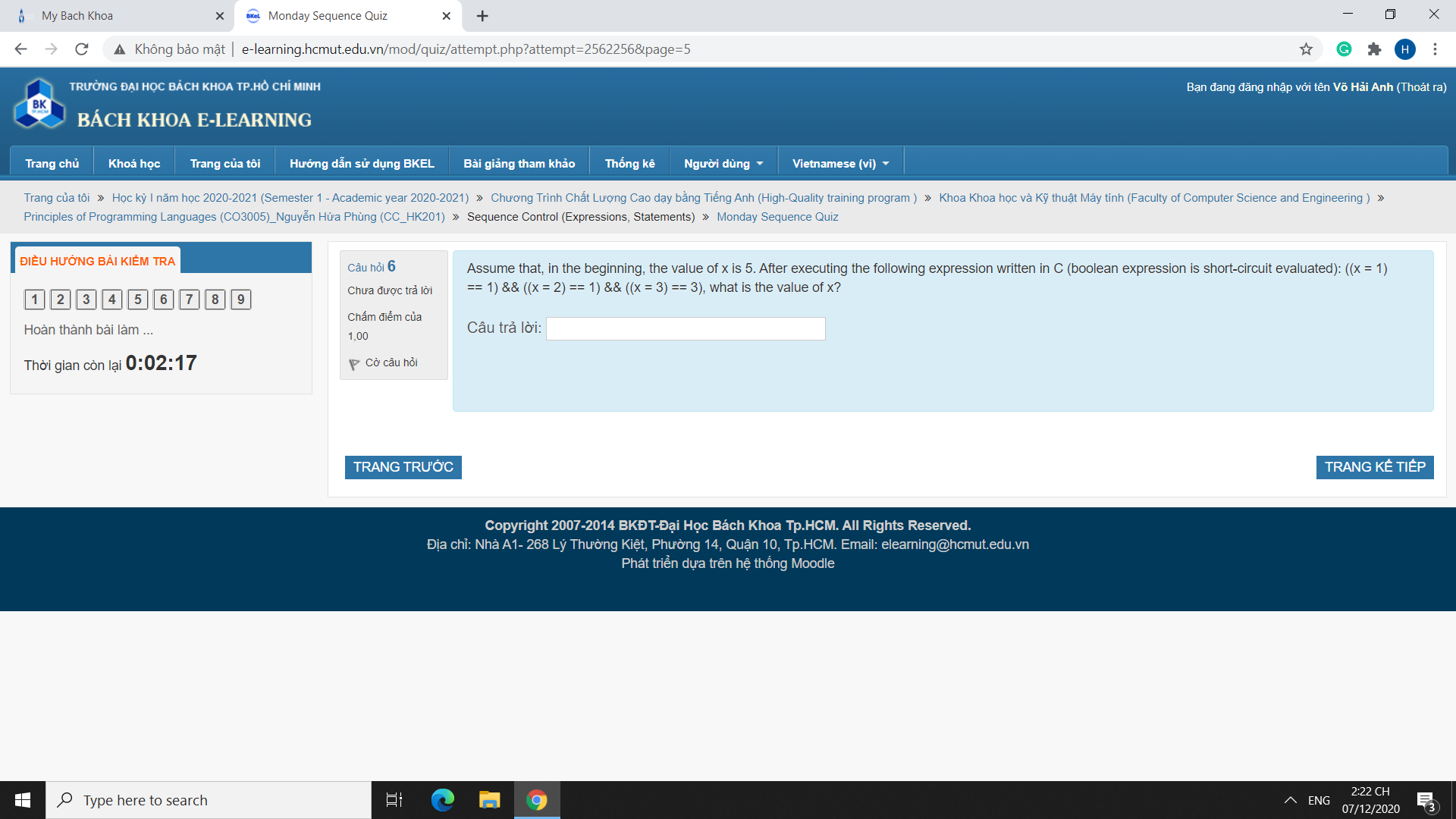
a/(b-c-e)/d/f => abc-e-/d/f/, assuming left-associative division.



a/(b-c-d)/e/f\*g => (\*(/a(-bcd)ef)g)



C, because he asked about *do while*, not *while* (even though the example is for *while*).

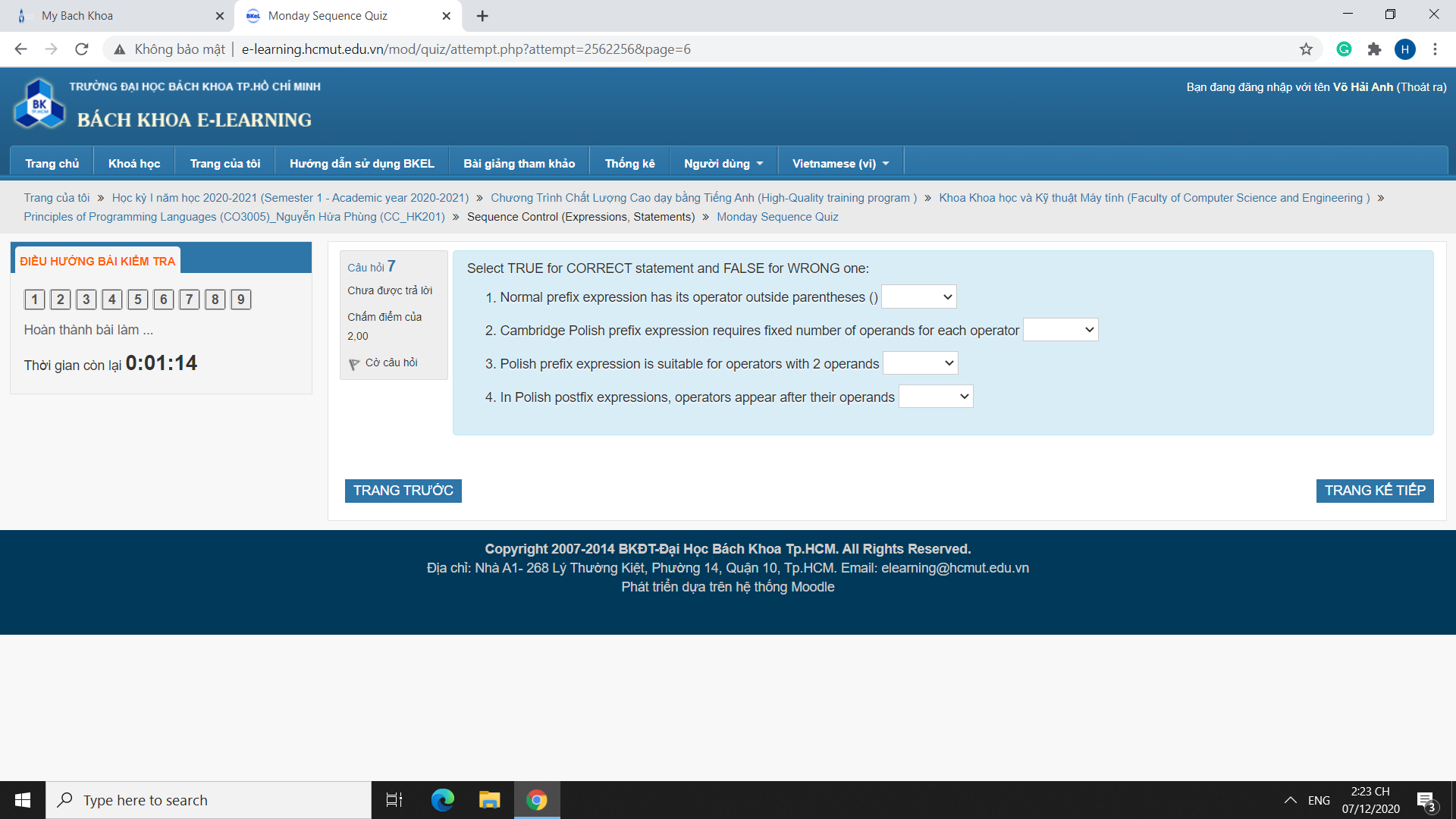


((x=1)==1)&&((x=2)==1)&&((x=3)==3).

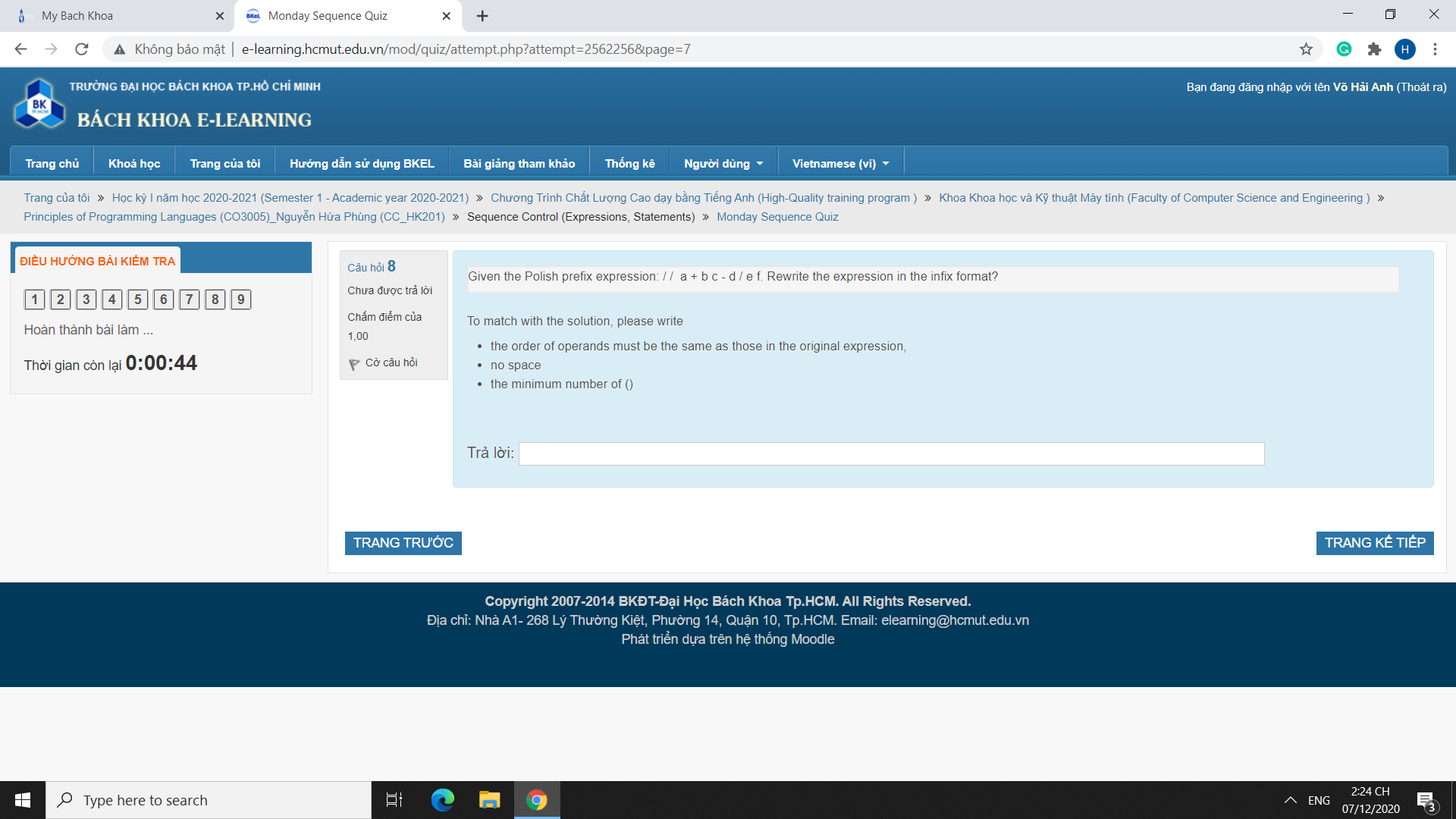
Remember that the && operator is left-associative in C.

* ((x=1)==1) is true, so we evaluate the second term. x is now 1.
* ((x=2)==1) is false, so we stop here. x is now 2.

Answer: 2.



* True
* False
* He didn't say *only*, so technically True. Else, pick False.
* True



//a+bc-d/ef => a/(b+c)/(d-e/f).

