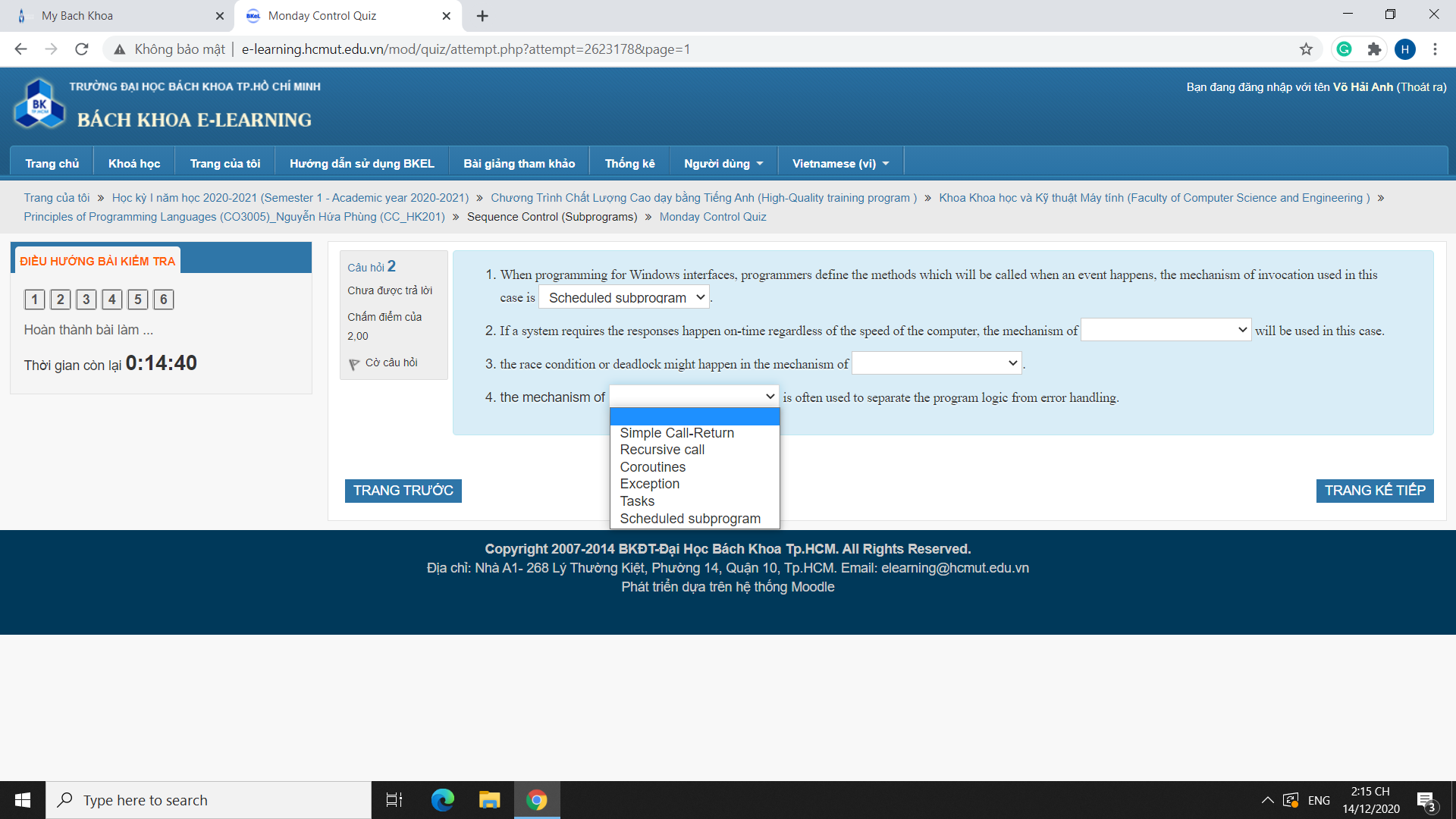
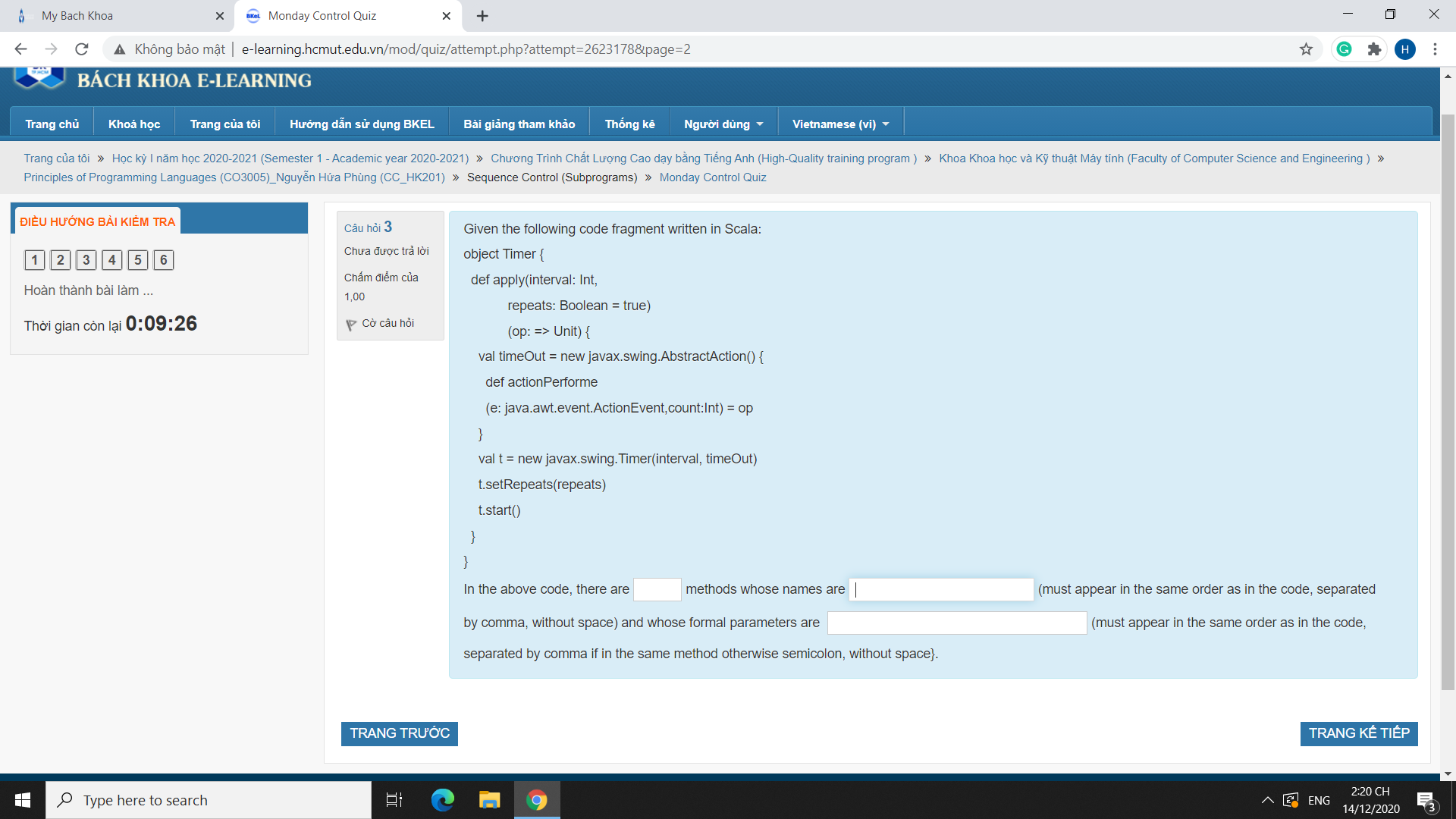
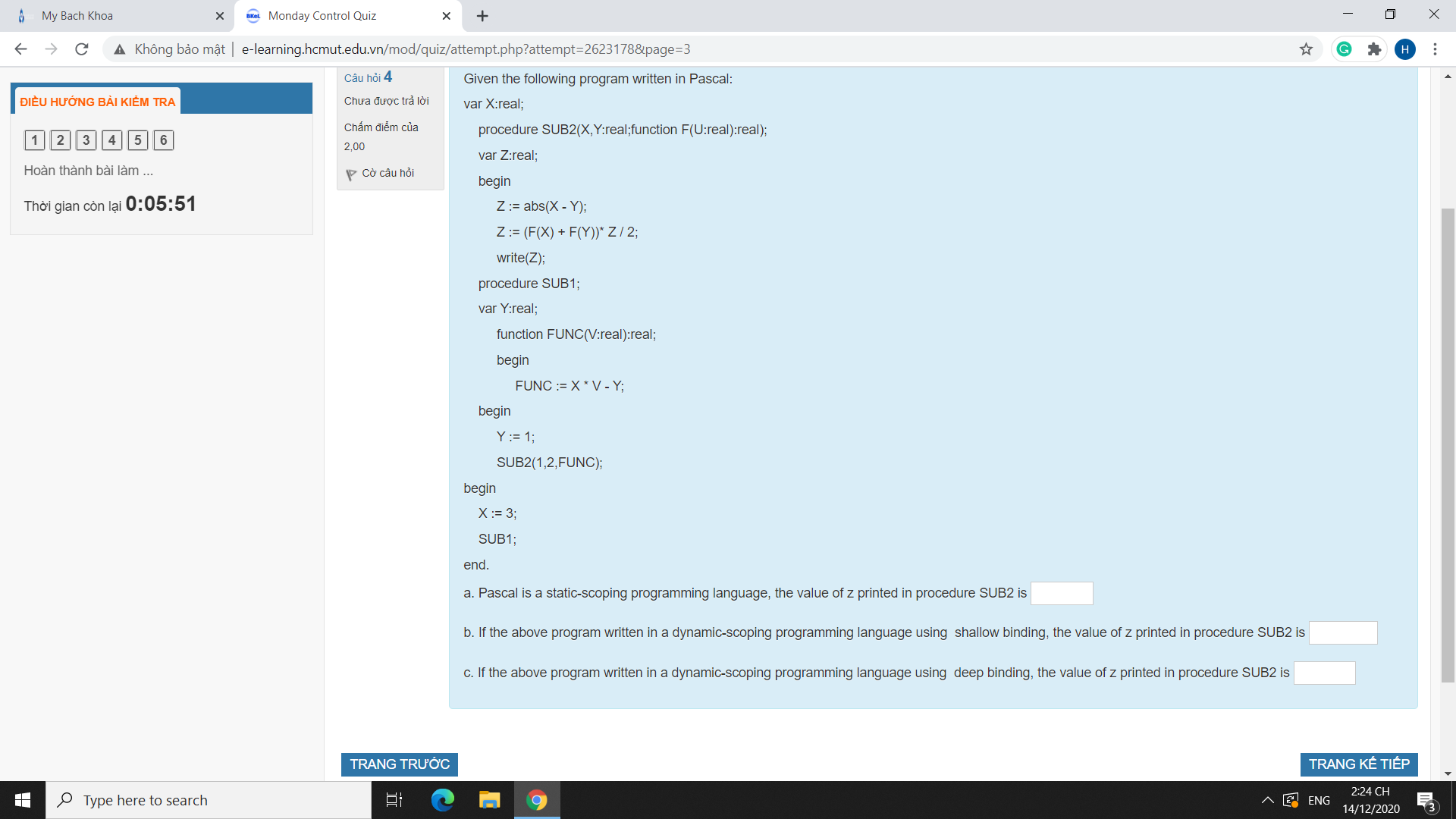


C (return address is kept in the activation record, which is the dynamic part of the activation. Compilers cannot know this beforehand, since all addresses are bound at runtime and therefore dynamic)





2 methods: apply,actionPerforme (sic). Parameters: interval,repeats,op,e,count.



**a)**

* Set global X to 3.
* Call SUB1():
  + - Set local Y to 1.
    - Call SUB2(1,2,FUNC)
      * SUB2's X and Y are set to 1 and 2.
      * Assign abs(1-2) to Z => Z = 1.
      * Call FUNC(1).
      * FUNC was declared inside SUB1, so it takes that environment.
        + Take global X (3), since SUB1's scope does not have X.
        + Take SUB1's Y (1).
        + V was actually passed from SUB2, so it takes on 1.
        + Result is 3\*1-1 = 2.
      * Call FUNC(2).
        + Same reasoning
        + Result is 3\*2-1 = 5.
      * Assign Z = (2 - 5)\*1/2 = -1.5 (Z is a real number).

Prints 3.5.

**b)**

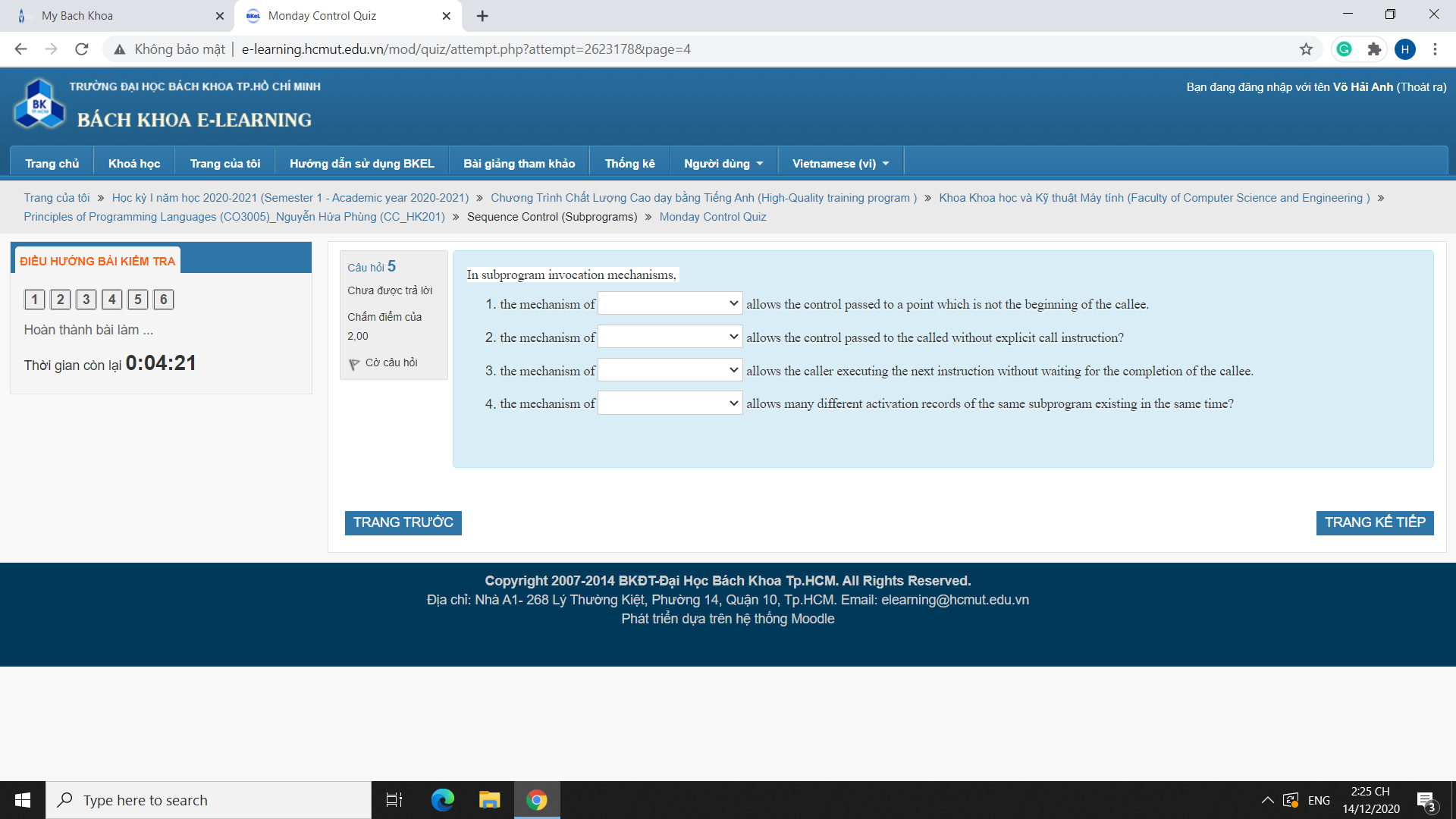
* Set global X to 3.
* Call SUB1():
  + - Set local Y to 1.
    - Call SUB2(1,2,FUNC)
      * SUB2's X and Y are set to 1 and 2.
      * Assign abs(1-2) to Z => Z = 1.
      * Call FUNC(1).
        + FUNC was called by SUB2, so it takes on that environent.
        + Take SUB2's X(1)
        + Take SUB2's Y (2).
        + V was actually passed from SUB2, so it takes on 1.
        + Result is 1\*1-2 = -1.
      * Call FUNC(2).
        + Same reasoning
        + Result is 1\*2-2 = 0.
      * Assign Z = (-1 - 0)\*1/2 = -0.5 (Z is a real number).

Prints -0.5.

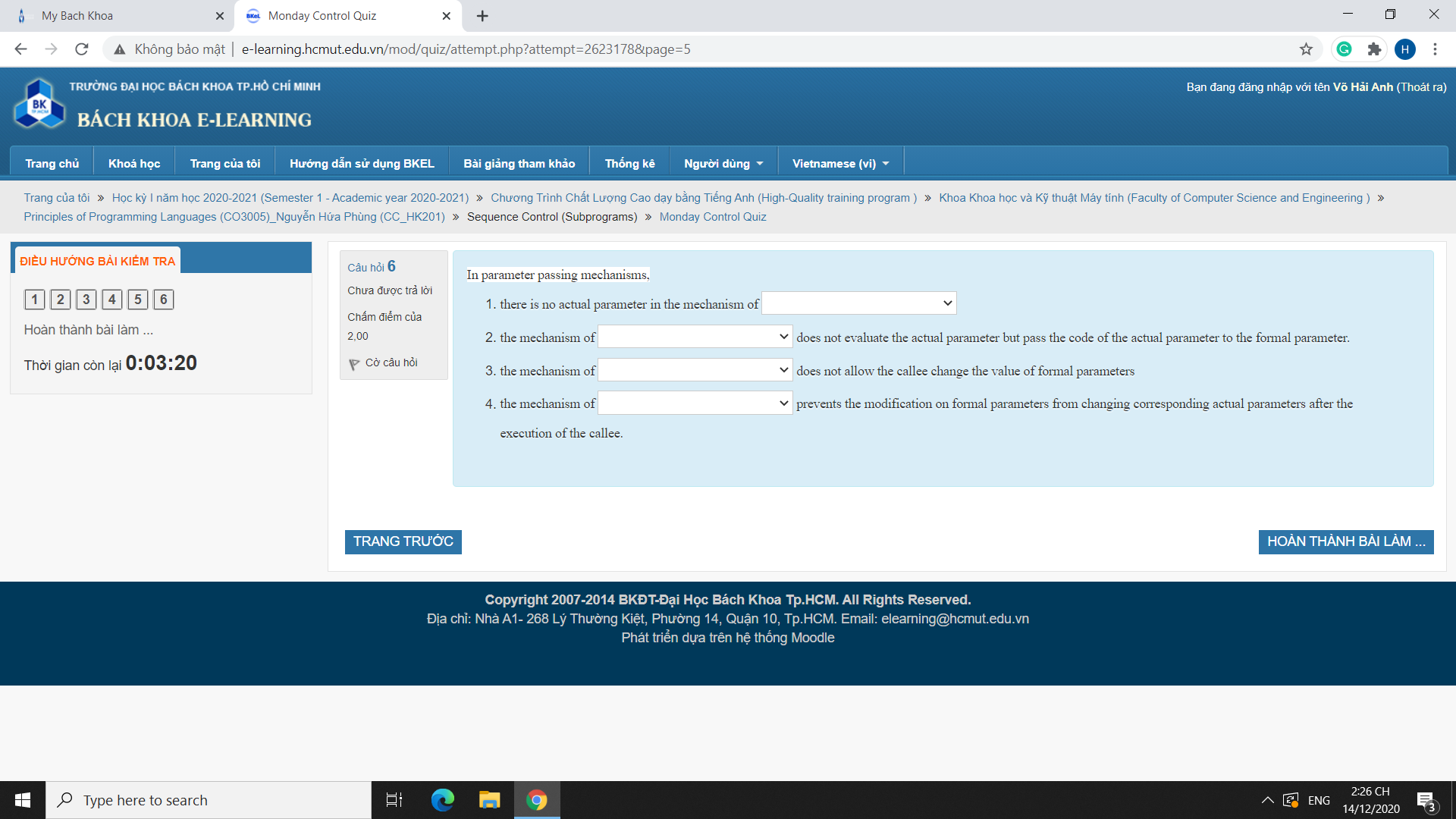
**c)**

* Set global X to 3.
* Call SUB1():
  + - Set local Y to 1.
    - Call SUB2(1,2,FUNC)
      * SUB2's X and Y are set to 1 and 2.
      * Assign abs(1-2) to Z => Z = 1.
      * Call FUNC(1).
      * FUNC was passed from SUB1, so it takes that environment.
        + Take global X (3), since SUB1's scope does not have X.
        + Take SUB1's Y (1).
        + V was actually passed from SUB2, so it takes on 1.
        + Result is 3\*1-1 = 2.
      * Call FUNC(2).
        + Same reasoning
        + Result is 3\*2-1 = 5.
      * Assign Z = (2 - 5)\*1/2 = -1.5 (Z is a real number).

Prints 3.5.



* Coroutines (starts where the previous yield left off).
* Exceptions (may be called by buttons or other interactions).
* Tasks (caller returns to executing concurrently with task).
* Recursions, obviously.



* No actual parameter in call by result.
* Call by name passes the code instead of actual parameters.
* Call by constant reference does not let the body change the formal parameters. Emphasis on formal parameters instead of actual (as call by value would allow the body to change the formal parameters locally, although the changes would not propagate to the outside).
* Call by value (contrast this with the immediate previous point).