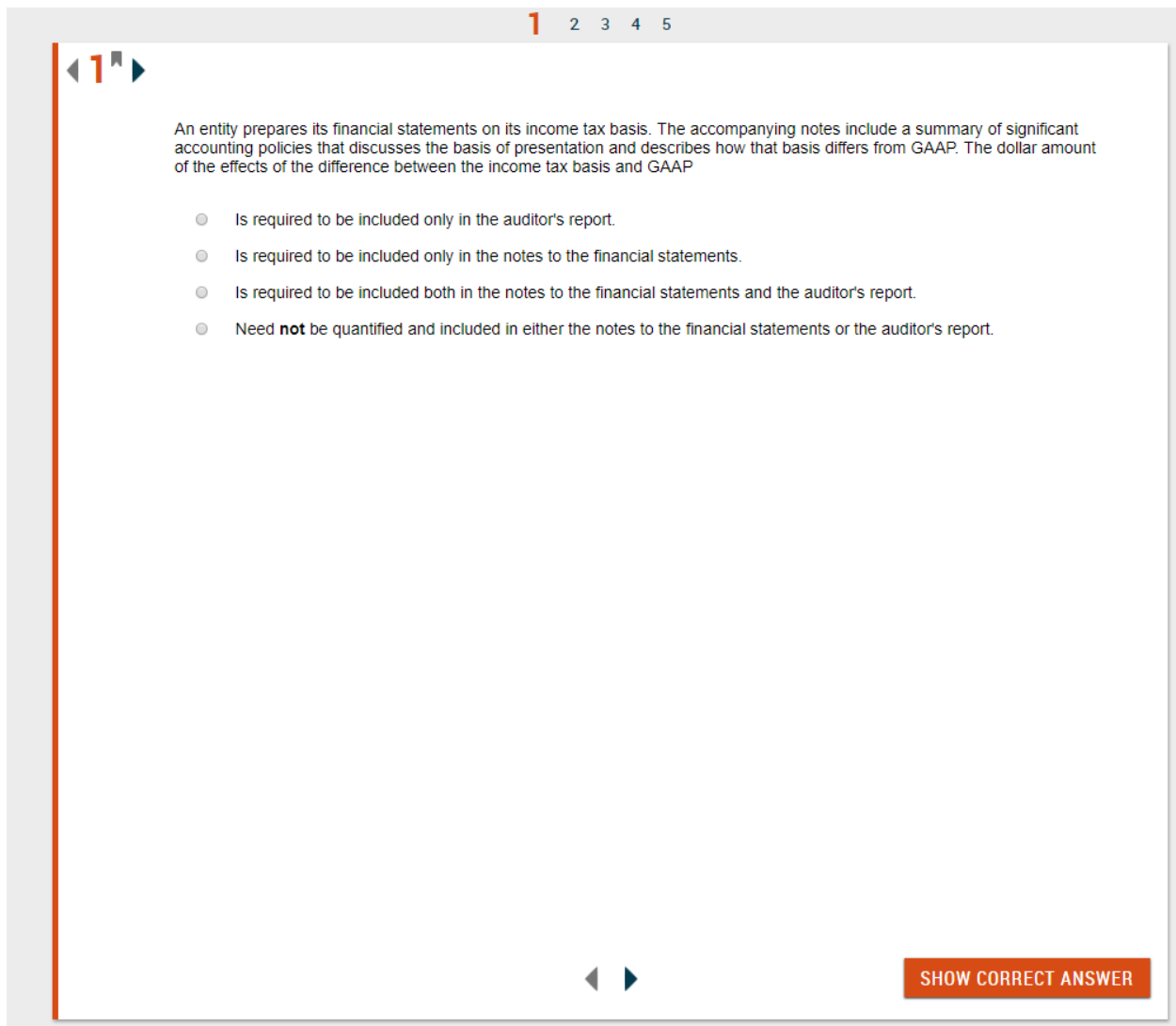


CANDIDATE ASSESSMENT:

In the assessments industry, items are test questions to be answered by candidates. Items can be pretest or operational. Pretest items don't count towards a candidate's score.

Candidates are given a set of items they need to answer. This set is called a Testlet.

A sample Testlet with 5 items (**pictured below**)—



The screenshot shows a digital test interface. At the top, there is a navigation bar with five numbered tabs: 1, 2, 3, 4, and 5. Tab 1 is currently selected and highlighted in orange. Below the navigation bar, on the left, are navigation icons: a left arrow, a large orange '1', and a right arrow. The main content area contains a text-based question: "An entity prepares its financial statements on its income tax basis. The accompanying notes include a summary of significant accounting policies that discusses the basis of presentation and describes how that basis differs from GAAP. The dollar amount of the effects of the difference between the income tax basis and GAAP". Below the question are four radio button options: "Is required to be included only in the auditor's report.", "Is required to be included only in the notes to the financial statements.", "Is required to be included both in the notes to the financial statements and the auditor's report.", and "Need **not** be quantified and included in either the notes to the financial statements or the auditor's report.". At the bottom of the interface, there are two navigation arrows (left and right) and a red button labeled "SHOW CORRECT ANSWER".

Using Test-First Development build the functionality below -

- There is a Testlet with a fixed set of 10 items. 6 of the items are operational and 4 of them are pretest items.
- The requirement is that the _order_ of these items should be randomized such that -
 - The first 2 items are always pretest items selected randomly from the 4 pretest items.
 - The next 8 items are mix of pretest and operational items ordered randomly from the remaining 8 items.

To get started you may use the boilerplate structure below:

```
public class Testlet
{
    public string TestletId;
    private List<Item> Items;

    public Testlet(string testletId, List<Item> items)
    {
        TestletId = testletId;
        Items = items;
    }

    public List<Item> Randomize()
    {
        //Items private collection has 6 Operational and 4 Pretest Items.
        Randomize the order of these items as per the requirement (with TDD)

        //The assignment will be reviewed on the basis of - Tests written first, Correct
        logic, Well structured & clean readable code.

    }
}

public class Item
{
    public string ItemId;
    public ItemTypeEnum ItemType;
}

public enum ItemTypeEnum
{
    Pretest = 0,
    Operational = 1
}
```