DIDATTICA DELL'INFORMATICA

Questions

Hazzan et al., Guide to Teaching Computer Science,

Prof.ssa Filomena Ferrucci
fferrucci@unisa.it

1

Questions: Not only assessment

- Deepen students' understanding of the learned CS concepts
- Refine students' understanding of complex concepts
- Let students acquire different cognitive skills (mental abilities we use while thinking, learning, and studying)
- Express students' knowledge in different forms
- A variety of questions provides intellectual challenges and maintains learners' concentration, interest, and motivation

2

Questions types

- Develop a solution
- Develop a solution that uses a given module
- Tracing a given solution
- Analysis of code execution
- Finding the purpose of a given solution
- Examination of the correcteness of a given solution
- Completetion of a given solution
- Instructions manipulations
- Efficiency estimation
- Question design
- Programming style questions
- Transformation of a solution
- Combining several types of questions
- Story questions

3

9.2.1 Type1. Development of a Solution

Description: A development question presents an open problem in which learners are required to develop their solution to a given problem. The solution can be expressed by a descriptive algorithm, pseudo-code or a program in a specific programming language.

Example: Write a method that returns the number of (integer) divisors for a given integer n.

Variations: A development question can ask for, for example: (a) a single method (as the presented example); (b) a sequence of tasks to be performed; (c) a complete program; or (d) a method with a specific efficiency (in the example presented it can be $O(\sqrt{n})$).

Discussion: This type of questions can be solved in different ways. In some cases, the differences are not meaningful; in other cases, the different solutions represent different algorithmic approaches.

Variation (d) is not a fully open question since learners are asked to address a specific constraint—specific efficiency, and cannot develop any solution for the problem; therefore, this variation requires wider range of considerations than the other ones and is considered a harder question than the other variations.

4

Quali sono gli obiettivi formativi?

5

9.2.2 Type2. Development a Solution That Uses a Given Module

Description: In this case, the development question relates to a pre-prepared module and asks learners to present a solution to a given problem while considering and using a given module. A documentation of the module is included in the question and the student must use it in their solution.

Example: Write a method that returns an integer number between 1−*n* that has the largest number of divisors for a given integer *n*. Use the method *numberOfDividers*(*n*), that returns the number of divisors for a given integer *n*.

Variations: A development question that relates to a given module can be presented, among other ways, in one of the following forms: (a) write an instruction that invokes a given method; (b) write a method that uses a given method (as in the given example); (c) write a method that uses a given module a specified number of times; (d) write a method that uses several different given methods; (e) questions in which the given module is not a method, but rather it is, for example, a specified data structure or a specified class.

6

Quali sono gli obiettivi formativi?

7

9.2.3 Type3. Tracing a Given Solution

Description: A given code is presented and the learners are asked to track the code execution.

Example of a general pattern: Present a tracing table that follows the execution of a given method. The table should include a column for each variable and for the code output.

Variations: A tracing question can ask to follow, for example: (a) a complete program; (b) a single method; (c) a recursive method; (d) object creation. In addition, the following instructions can be used in each of the above variations: (1) follow the code execution according to a given input; (2) follow the code execution when learners choose the input; (3) follow the code execution according to several different specified inputs which are selected in a way that guides the learners to find what the given code performs; (4) find different sets of inputs so that each set represents a different flow by which the code is executed; (5) find a set of inputs that yields a specific output.

Discussion: Variations (1)–(3) can be considered as closed questions. The learner is required to trace a given code with a specified (given or chosen) input, and there is only one correct solution. Variations (4) or (5) require learners to apply deeper considerations and to examine the presented code from a higher level of abstraction. In these cases, it is not sufficient to understand different instructions; rather, they require code analysis—what the purpose of the code is and how it is achieved. Clearly, more advanced cognitive skills are needed in order to address meaningfully these variations.

8

9.2.4 Type4. Analysis of Code Execution

Description: A given code is presented and learners are asked to analyze specific aspects of the code execution.

Example of a general pattern: Look at the given code that includes a loop and answer the following questions:

1. For what values of *x* and *y* the loop is not executed at all?
2. For what values of *x* and *y* the loop is executed exactly one time?
3. For what values of *x* and *y* the loop will never terminate?

Variations: Variations (4) and (5) of Type3 questions—tracing a given solution—can be viewed also as variations of this type of questions.

Discussion: In this type of question, the learner is required to analyze the code execution and to understand it as a whole. Specifically, in order to solve such questions, learners should exhibit mainly two cognitive skills: understanding programming structures and understanding the logic of a given code. Therefore, a higher level of thinking is needed to solve such questions than that needed for solving a tracing question; accordingly, this type of question is considered harder than the "tracing a given code" type of question.

9

9.2.5 Type5. Finding the Purpose of a Given Solution

Description: A given solution to an unknown problem is presented and the learners are asked to state the purpose of the solution, that is, to determine what problem it solves.

Example of a general pattern: Look at the given method and write the method target, that is, what is the problem that the method solves?

Variations: A "finding the purpose of a given solution" question can relate to either: (a) a sequence of instructions; (b) a single method (like in the presented example of a general pattern); (c) a full program; (d) a class.

Discussion: This type of questions is considered harder than tasks that ask to develop a solution for a given problem. For solving this type of questions, a set of cognitive skills is required. Specifically, in addition to the understanding of the code execution and the ability to trace it, one should comprehend someone else's way of thinking.

To help students solve this type of questions, a question can contain scaffolding subquestions. For example, a question can include several tracing subquestions (Type3 questions), which aim to guide the students to discover the purpose of the code.

10

9.2.6 Type6. Examination of the Correctness of a Given Solution

Description: A given problem and its solution are presented. The student is asked to determine whether the given solution solves the given problem correctly.

Example: The following method was written by a student as a solution for the following problem:

Write a method that returns *true* if all values of a given array of integers are equals; otherwise, it returns *false*.

Is the method correct?

```
public static boolean equalValues(int[] arr) {
        for (int i=0; i<arr.length; i=i+2) {
            if (arr[i] != arr[i+1])
                return false;
        }
    return true;
}
```


11

Variations: This type of question can be presented in different forms: (a) determine whether a given solution to a given problem is correct (as in the example presented); (b) check if a given solution to a given problem is correct and explain your answer; (c) if the given solution is incorrect, give an example of an input that shows it; (d) if the given solution is incorrect, give an example of an input that presents a correct output and, therefore, may mislead one to conclude that the given solution is correct; (e) if the given solution is incorrect, correct the solution by introducing the minimal required changes (without this restriction, students may present a totally different solution); (f) the presented solution can contain more than one mistake, and the question can state it explicitly or not.

Additional variations of this type of questions may address syntactic mistakes. It is suitable to present such variations while introducing new instructions or data structures. It is not recommended, however, to use these variations in more advanced stages since they do not indicate learners' understanding of the *algorithmic* problem, and, further, they do not contribute meaningfully to learners' understanding since, in fact, the compiler directs how to debug such mistakes.

Discussion: In order to solve this type of questions, students should apply algorithmic thinking and logical skills. Here, as in Type5 questions, students should analyze a solution that may not fit their own way of thinking had they been asked to develop a solution. However, since the purpose of the solution is given, these tasks are considered easier than Type5 questions.

In the example presented, the two minimal required corrections are: (1) change the increment of variable *i* to *1* (instead of *2*); (2) change the range of variable *i* to be *i<arr. length-1*. Correction (1) is based on a logical consideration, while correction (2) addresses the array index, which is a more technical consideration.

12

9.2.7 Type7. Completion of a Given Solution

Description: A given problem and an incomplete solution of the given problem, in which some of the instructions are missing, are presented to the learners. The learners are asked to complete the missing instructions, so that the solution will solve the problem correctly.

Example: The following method was written by a student as a solution for the following problem: Write a method that for a given array of integers returns the number of array elements that are bigger than their two neighbors (the previous element and the subsequent element in the array).

```
public static int numberOfBigger(int[] arr) {
        _____;
        for (int i=_____; i<_____; i++) {
            if (_____)
                _____;
        }
        return _____;
}
```


13

Variations: "A completion a given solution" question can be varied by changing the number of the missing instructions. This number should be determined by taking into the consideration that it may affect the difficulty and complexity of the question.

In general, the missing instructions can relate to one or more aspects of the algorithm and the teacher should consider whether to focus on one or more aspects. For example, if the teacher's target is to focus on the use of a *boolean* flag, the missing instructions should be only those that relate to this flag; if the target is to focus on the loop limits, the limits should be the missing parts and, sometimes, also the increment of the loop control variable.

In the example presented, the missing instructions are related to three aspects: the counter control (initialization, increment, and return); the range of the loop (the first and the last array elements should not be accessed in the loop because they do not have two neighbors); and the specific condition to be checked.

Discussion: This type of question, as well, requires students to understand the logic of the given solution, when the actual question difficulty is determined according to students' level and stage of learning. Still, with respect to each CS subject, relatively simple "completion of a given solution" questions exist, where the understanding of the logic of the solution is straightforward. At the same time, however, there are more challenging questions and the teacher should be aware of this complexity. For example, asking to complete instructions in a given bubble sort algorithm, in which meaningful instructions are missing, without introducing the rational of this sorting approach, is considered a difficult question.

14

9.2.8 Type8. Instruction Manipulations 1/3

Description: A problem and its solution are given. Students are asked to address different manipulations performed on the solution.

Example: The following method executes a variation of the selection sort.

```
                public static void selectionSort(int[] arr) {
                    int p, temp;
(1)                 for (int i=0; i<arr.length-1; i++) {
(2)                     p = i;
(3)                     for (int j=i+1; j<arr.length; j++) {
                            if (arr[j]<arr[p])
                                p = j;
                        }
(4)                     if (p != i) {
                            temp = arr[i];
                            arr[i] = arr[p];
                            arr[p] = temp;
                        }
                    }
                }
```


15

Type 8. Instruction Manipulations 2/3

Answer the following questions and explain your answers:

1. Will the algorithm correctness be effected if the instruction marked by (2) is removed and the instruction marked by (3) is replaced with the following instruction:
   (3) *for (int j=i; j<arr.length; j++) {*
2. Will the algorithm correctness be affected if the instruction marked by (4) is removed and the contents of the two array elements are swapped anyway?
3. Will the algorithm correctness be affected if *all* the body of the loop marked by (3) is replaced with the following instructions?

(3) *for (int j=i; j<arr.length; j++) {*

16

Type 8. Instruction Manipulations 3/3

Variations: An "instruction manipulations" question can be implemented by the (a) addition of instructions; (b) removal of instructions; (c) changing instructions; (d) replacement of instructions. The question itself can address the target of a specific code or the tracing of the changed code and the examination of differences between outputs.

Discussion: Questions that manipulate a given solution enable to concentrate on meaningful aspects of the algorithm. In the teaching process, a discussion on such manipulations can clarify the essence of a given solution as well as other CS topics. We illustrate this idea by the concept of generalization. Students can be instructed to change a given method slightly, in a way that generalizes the original method and solves a broader task. For example, a method that sorts an array can be slightly changed in order to sort a section of the array between two given indices. After the change, the method can sort different sections of an array, as well as the entire array (with the indices 0 and the array length—1).

17

9.2.9 Type9. Efficiency Estimation

Description: Students are asked to estimate the efficiency of a given solution.

Example of a general pattern: Estimate the efficiency of the presented method in terms of *big O*. Explain your estimation.

Variations: This type of questions can represent different levels of cognitive complexity, as is described in what follows. (a) A general pattern that enables to discuss efficiency in early stages of CS learning: Focus on the loop in the given method: How many times is the loop executed?; (b) Estimate the efficiency of a specific method (as in the example of a general pattern); (c) Estimate the efficiency of a method that invokes another method, when the efficiency of the invoked method is taken into the consideration; (d) Compare the efficiency of different methods that solves the same task; (e) Estimate the efficiency of a recursive method; (f) Develop of a solution to a given problem with a specific efficiency.

Discussion: Instructors should not wait till they teach complicated algorithms in order to teach the concept of efficiency; alternatively, questions that deal with efficiency can be integrated from early stages of teaching and learning CS. For example, questions, such as the one presented in variation (a), hints at the seeds of the concept of efficiency that, in general, is considered to be abstract and difficult for understanding.

Note that variation (f) is actually a "development of a solution" (Type1 question) with a restriction about its efficiency, which requires learners not to be satisfied when finding an algorithmic idea that solves the given problem; rather, they should estimate its efficiency and if it does not fit the restriction mentioned in the question, they should look for another solution or improve the solution they found.

18

9.2.10 Type10. Question Design

Description: Students are asked to design a question.

Example1: Design a question that checks learners' understanding of the sort-merge algorithm.

Learners' answer to this question can be based, for example, on tracing regular and/or extreme cases of the input to this algorithm.

Example2: Design a question in such a way that its solution uses a method that finds the most frequent value in an array. An example for such a question is:

Pupils' grades in a CS test are given. Write a method that prints the most frequent grade in this class.

Variations: A design question can relate, among other options, also to the design of (a) additional tasks in a given question that clarify extreme cases; (b) a question that its solution should use a given method (as Example2 is); (c) a question that intends to check the understanding of a specific concept or algorithmic idea (as Example1 is); (d) a whole test or worksheet that examines a specific CS concept/topic.

Discussion: This type of questions invites learners to adopt a different point of view. In addition to the experience learners gain by examining a question from the educator's point of view, this kind of questions encourages them to scrutinize the learned concepts, to reflect on what they learned, and, by doing so, also to evaluate their own understanding. In addition, the design of the questions is a kind of active learning that encourages creativity.

19

9.2.11 Type11. Programming Style Questions

Description: Learners are asked to examine the programming style of different solutions presented for the same task.

Example of a general pattern: Look at the given collection of *correct* solutions for a given problem. Examine the solutions and state which of them, in your opinion, is the best solution. Explain your choice.

Variations: The different solutions for the given problem can differ in one or more aspect(s), according to the teacher decision, such as (a) different kind of loops; (b) the need to use an array for the solution; (c) different algorithmic approaches (e.g., given two correct solutions to a given problem, the learners should decide which one is a better solution and explain why). If the teacher decides to integrate in the question several aspects, the different aspects can be presented explicitly in the question when learners are asked to analyze the solutions according to each aspect.

Discussion: This type of question enables to foster a discussion about different aspects of programming style, which in turn increases learners' awareness to these aspects.

20

9.2.12 Type12. Transformation of a Solution 1/2

Description: A problem and its solution are presented to the learners in a specific programming approach, programming language, or programming paradigm. The learners' task is to transform the solution into a different programming approach, a different programming language, or a different programming paradigm.

Example of a general pattern1: The presented loop is implemented by a *while* loop. Implement it by a *for* loop.

Example of a general pattern2: The method presented is implemented by a *while* loop. Implement it by a recursive method that achieves the same target.

Example of a general pattern3: The following method sorts an array of integers in the *imperative* programming paradigm. Implement it in the functional programming paradigm.

Variations: The different transformations presented in this kind of questions can be (a) between programming paradigms (as in Example of a general pattern3). This variation can be carried out only after the two said programming paradigms were learned; (b) within the same programming paradigm but between programming languages; (c) within the same programming language but between structures (as in the example of general pattern1 or, for example, from a nested *if* statement to *switch-case* statement); (d) within the same programming language but between different

21

Type 12. Transformation of a Solution 2/2

algorithmic approaches (as in the example of general pattern2); (e) between different representations, for example, from pseudo-code to any formal language. This variation, however, does not foster problem-solving skills and does not involve meaningful CS concepts. It can serve, however, for practicing different kinds of algorithm representations.

Discussion: The focus in this type of questions should be placed on conceptual aspects, rather than on syntactic aspects. By conceptual aspects we refer, for example, to problem analysis according to two different programming paradigms or the transformation of a sequential solution into a recursive solution in the same programming language.

In a similar way to the "programming style questions" (Type11), "transformation of a solution" questions enable to concentrate on core CS concepts. In addition, such questions lead students to explore different problem-solving approaches. It should be remembered, though, that since this type of questions demands skills of high level of abstraction, it does not necessary fit all learners.

22

9.2.13 Combining Several Types of Questions 1/2

Though the above types of questions attempt to classify CS questions, in most cases, questions either combine several types of questions (as the following example illustrates) or cannot be classified at all (see Activity 75).

Example: The target of the following two methods is to determine whether an integer number *n* is a prime number or not.

Method A

```
public static boolean prime(int n) {
    for (int i=2; i<n; i++) {
        if (n % i == 0)
            return false;
    }
    return true;
}
```

Method B

```
public static boolean prime(int n) {
    if (n % 2 == 0)
        return false;
    for (int i=3; i<n; i=i+2) {
        if (n % i == 0)
            return false;
    }
    return true;
}
```


23

1. *Type3. Tracing a given solution*: Trace each method when *n* is *19*. 2/2
2. *Type4. Analysis of code execution*:

- For each method, determine how many times the loop is executed for $n = 19$.
- Find a value of *n*, for which the loop in Method B is executed ten times. Is there only one answer?

3. *Type5. Finding the purpose of a given solution*: What is the purpose of each method? (can be asked if the problem is not indicated).
4. *Type6. Examination of the correctness of a given solution*: Check the correctness of the two solutions. Do they solve the problem correctly?
5. *Type9. Efficiency estimation*: What is the efficiency of each method?
6. *Type6. Correctness; Type8. Instruction manipulations; Type9. Efficiency*:

- If you change the upper loop limit in Method B to be $n/2$ (instead of *n*), is the solution still correct? If it is, what is the method efficiency after the change?
- If you change the loop limit in Method B to be $\sqrt{n}$ (instead of *n*), is the solution still correct? If it is, what is the method efficiency after the change?

24

9.4.1 Story Questions 1/2

Story questions presented to CS learners can be divided literally into two main kinds: pure-algorithmic tasks and narrative-algorithmic tasks. *Pure-algorithmic* tasks are problems that directly and explicitly address the program structures and variables, and present the task by using this terminology. *Narrative-algorithmic* tasks are problems that neither relate to program structures nor to its variables; rather, in this kind of questions, the problem to be solved is embedded in a story and in order to solve the problem, learners should recognize both what is given and what the target of the problem is. Specifically, learners should decide which elements included in the question formulation are relevant and which ones are irrelevant, and based on this decision, to solve the task. Most of the examples presented in the list of types of questions (Sect. 9.2) are pure-algorithmic tasks where the problem target is presented explicitly.

Table 9.3 presents three story tasks in two ways: as pure-algorithmic tasks and as narrative-algorithmic tasks.

It is important to address these two kinds of story questions in the MTCS course, not only because they are different, but also because they require different problem-solving skills. A pure-algorithmic question indicates specifically the task to be solved; in narrative-algorithmic tasks, students should discover the task to be solved. Since in the real world, most problems are based on narratives, the ability to solve of narrative-algorithmic tasks is an important skill that CS learners should acquire. It should be remembered, though, that these questions are more complicated.

Accordingly, when teaching a new CS content, story questions should be addressed in several stages: (1) present a general story that embeds the new learned topic, so that the class gains the essence and target of the new topic; (2) focus for a while on pure-algorithmic questions, to allow a gradual knowledge construction process of the new tool or structure; (3) integrate narrative questions in the continuation of the teaching process.

25

Table 9.3 Tasks as pure-algorithmic tasks and as narrative-algorithmic tasks 2/2

The task	Pure-algorithmic formulation	Narrative-algorithmic formulation
Find the maximum of a list of numbers	Write a method that returns the maximum value of a given list of integers	In a sport competition, 5 classes of 30 pupils each participates in two jumping competitions. Write a program that displays for each class the best result in each of the two jumping competitions for given two results of each student
Checks whether a given array is sorted	Write a method that returns true if a given array is sorted; otherwise, it returns false	A teacher wishes to encourage his or her pupils, and to give them a written recognition if their grades are improved in each test. Write a method to determine whether a given student deserves the recognition based on his list of grades
Change characters to their successive characters according to the Unicode table	Write a method that changes a given array of characters in a way that replaces each character with its successive character according to the Unicode table	A message that should be sent between financial partners should be encoded. The message includes words, spaces, and dots. Write a method that returns a coded message in which each letter of the given message (String) is replaced by its successive letter in the alphabetical order. The letter "Z" will be replaced with the letter "A." Spaces and dots should not be changed

26

Not only programming!

Table 9.4 Illustration of the types of questions in the context of Automata theory

Type of question	An example pattern
Type1: Development of a solution	Design a finite automaton that recognizes a regular language L
Type2: Development of a solution that uses a given module	Given the A1 finite automaton that recognizes the language L1 and A2 finite automaton that recognizes the language L2, design a finite automaton that recognizes the language L1 ∪ L2
Type3: Tracing a given solution	Given a push down automaton P, and a word w, show the sequence of states that P goes through while processing w
Type4: Analysis of code execution	Given a finite automaton A, present A word that the automaton accepts A word that the automaton rejects A word that its processing is terminated in the trap state
Type5: Finding the purpose of a given solution	Given a Turing Machine T, determine what language it accepts
Type6: Examination of the correctness of a given solution	Does the given Turing Machine T recognize the language L?
Type7: Completion a given solution	Complete the Push Down Automaton P, so it will recognize the language L
Type8: Instruction manipulations	Given a Turing Machine T, if the transition from state q1 to q2 is replaced by the next transition [to be described], what language will the machine recognize?
Type9: "Efficiency" estimation	Given a finite automaton A that recognizes the language L, can you present a different finite automaton that recognizes the same language with fewer states?
Type10: Question design	Design a question that requires the presentation of a BNF grammar for an irregular language
Type11: "Programming" style questions	Given three different Push Down Automata that recognize a language L, examine the automata and state which of them, in your opinion, is the best. Explain your answer
Type12: Transformation of a solution	Given a Turing machine T, present a BNF grammar that expands the same language

27

The process


Planning


Solving


Estimation

28

1. *Planning*: We lay out questions that CS teachers should ask and answer in the process of question planning:

- What is the target of the question?
- What does the question intend to examine?
- What knowledge and skills are students supposed to possess in order to solve the question?
- Does the previous learning process enable students to acquire those skills?
- Does the previous learning process include the needed knowledge?
- What is the level of abstraction needed to solve the problem?
- Is the question varied from other questions presented so far in the class?

2. *Solving*: Teachers must solve any question before presenting it to the learners. It is important for any question, but especially necessary in the process of test preparation (see Chap. 10). Indeed, there are questions that look simple, but turn out to be difficult. Until a complete solution is presented, one cannot be sure that the question fits its purpose as well as the other aspects presented in the planning stage. If possible, it is recommended to ask a colleague to read at least the question formulation and verify that the question is understandable, clear, and not ambiguous. When the answer is based on code writing, it is necessary to run it on the computer and check that it works properly.
3. *Estimation of the needed time to solve the question*: Time is a crucial resource in teaching processes. Therefore, teacher should estimate the time required by learners to solve any particular question, which is usually longer than the time required by the teacher to solve it. The time estimation in this context relies on different factors, such as the effort involved in reading, planning, and writing the answer to the question. If a specific question turns out to be time consuming, it is important to remember that, in most cases, a different type of question, that requires less effort and meets the same pedagogical targets, can be developed.

29

Warning!

Coerenza tra obiettivi e modalità di verifica!!!

30

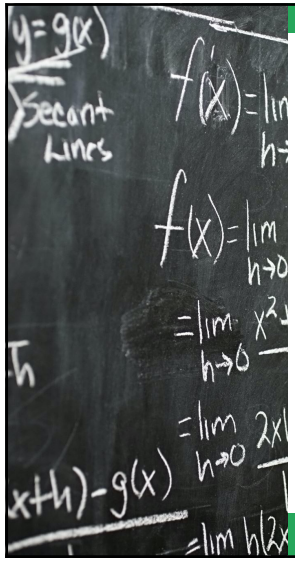
- Canterbury QuestionBank

http://web-cat.org/questionbank/questions-p1.html

Challenge

- Kahoot *kahoot.it*

31



Task

- Coppie: individuato un argomento, ognuno prepara una question (identificando l'obiettivo che vuole verificare) e pone all'altro la domanda. Si riflette insieme per comprendere l'efficiata della domanda per quell'obiettivo
- Di gruppo: Individuato un argomento a scelta (meglio se non di programmazione) formulare le domande per le 12 tipologie

32