Types of Questions in Computer Science Education

9

Abstract

It explores and discusses different types of questions that computer science (CS) educators (middle and high school teachers as well as university instructors) can use in different teaching situations and processes: in the classroom, in the computer lab, as homework, or in exams. The chapter discusses also keywords that appear in problem-solving questions which reflect the need to apply high-order cognitive skills by learners when answering these questions. The chapter lays out the advantages of using a variety of question types both for learners and teachers, and focuses on the design process of different question types. Though the types of questions presented are mainly related to programming assignments, most of them are suitable also for other CS contents.

9.1 Introduction

The main target of this chapter is to enrich computer science (CS) educators' toolbox with respect to the design of different types of questions and to illuminate the important role of CS educators in exposing their pupils/students to different types of questions throughout the learning-teaching processes. Learners' work and exploration of different types of questions and their variations deepen their understanding of the learned CS concepts, refine their understanding of complex concepts, and let them acquire different cognitive skills. Such an experience also provides learners with the opportunity to express their knowledge in different forms. Further, the use of a variety of question types sometimes provides intellectual challenges and maintains learners' concentration, interest, and motivation.

O. Hazzan et al., *Guide to Teaching Computer Science*,
DOI 10.1007/978-1-4471-6630-6_9

This teacher's role should be delivered to the prospective CS teachers, and therefore, it is recommended to use/mention/practice a variety of different types of questions throughout the Methods of Teaching Computer Science (MTCS) course in different opportunities and contexts. Clearly, instructors of the MTCS course should also apply the same pedagogical principles and vary the types of tasks and questions they use while discussing the teaching of CS with the prospective CS teachers.

One of the common problem-solving scenarios in CS education starts with the presentation of an open problem that describes some story, continues with the problem analysis and planning of its solution, and ends with the presentation of the solution as an algorithm either in pseudocode or a specific programming language. It is important, however, that CS teachers be aware of the fact that additional types of questions exist.

Several pedagogical targets can be achieved by the integration of different types of questions in the teaching of CS, as is laid out in what follows:

1. Different types of questions enable to illuminate different aspects of the learned content.
2. The integration of different types of questions throughout the teaching process helps maintain the students' interest, attention, and curiosity.
3. Different types of questions enable teachers to vary their teaching tools.
4. Different types of questions require the students to use different cognitive skills—mental abilities we use while thinking, learning, and studying. This target is important to enable each learner express his or her unique individual cognitive skills, to articulate his or her knowledge in his or her unique way, and to develop and enrich one's cognitive skills.

Cognition is the process of thought and it can be analyzed from different perspectives. For example, in psychology or philosophy, the concept of cognition is closely related to abstract concepts, such as mind, reasoning, perception, intelligence, and learning, all of which describe the mind capabilities. The field of cognition studies specifies mental processes, such as comprehension, inference, decision making, planning, and learning. With respect to CS, we are all familiar with the advanced cognitive skills of abstraction, generalization, concretization, and meta-reasoning.

Research works in CS education deal with different types of questions from the cognitive perspective. For example, Thompson et al. (2008) reviewed the work that has been done throughout the last 10 years with respect to the application of Bloom's taxonomy (Bloom et al. 1956) to CS course design, evaluation, and assessments, and provided an interpretation of the taxonomy that can be applied to introductory programming exams. Their interpretation focuses on the cognitive skill involved in addressing several types of questions. Jones et al. (2009) focus on written examinations. They determine the difficulty level of each question by keyword/s found in the question, and present cross-analysis that addresses student performance, cognitive skills, and learning outcomes. Other kind of works that combine CS questions and cognition relates to automata systems of question-answering processes (see, e.g., Pomerantz 2002; Yang et al. 2008).

We first concentrate on question *patterns* and present 12 types of questions that CS educators can use (Sect. 9.2). For each type of question, we lay out several variations. Clearly, additional types of questions, as well as the combination of different types of questions, exist and can be developed and used by CS teachers. Section 9.3 presents keywords of problem-solving questions. Then, Sect. 9.4 presents three general kinds of questions (story questions, closed questions, and unsolvable questions). Section 9.5 describes how the different types of questions can be assimilated to different CS contents and demonstrates these assimilations in the context of Automata Theory. In Sect. 9.6, we present guidelines on how to develop questions to be used in a CS class. We suggest several course activities to be facilitated in the MTCS course in order to expose the prospective CS teachers to this topic.

9.2 Types of Questions

This section presents 12 types of questions and suggests how they can be used in CS teaching processes. Emphasis is placed on the pedagogical approaches they represent and on cognition considerations.

Each type of question is presented according to the following pattern: *classification* of the type of question reflected by its title; short *description* of the specific type of question; a concrete *example* or an *example of a general pattern* that demonstrates the said type of question; different *variations* of the discussed type of question; and a short pedagogical and cognitive *discussion* about the said type of question. Since our purpose here is to present the variety of questions to future high-school prospective teachers, most of the examples are quite simple. Clearly, for each type of question, it is possible to develop a range of questions on different complexity levels both from the algorithmic and the cognitive points of view. In addition, with respect to each type of question, it is possible to present additional variations that require different cognitive skills.

We add several remarks about the actual use of this collection of questions in the actual teaching of CS situations:

- The order of the presentation of the 12 types of questions in this chapter is arbitrary; no specific rule is applied for their ordering.
- No specific rules or guidelines can be formulated with respect to the order by which it is recommended to preset the different types of questions in the class; each CS educator should select the appropriate type of question and its complexity level according to the specific learners' characteristics and the specific teaching situations.
- The suggested types of questions are sometimes overlapping each other; this point is addressed when it is relevant.
- A question can contain several types of questions in its different subtasks; this point is illustrated by a particular example in Sect. 9.2.13 "Combining several types of questions."

- In general, questions can be divided into two types: Pure algorithmic tasks and story-based algorithmic tasks; this perspective is discussed explicitly in Sect. 9.4.1 "Story questions."
- The types of questions presented in this section are mainly programming assignments; most of them, however, can be easily assimilated for other CS contents.

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.

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.

Discussion: The fact that students should relate to a given module influences the development process of the solution. For example, in the case of a given method, as in the above example, the learner has to suit the developed method to a specific subtask that the given method implements. This type of questions is considered harder than Type1 questions because in this case learners need to meet a constraint—the use of the given subtask.

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.

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.

9.2.5 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.

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;
}
```

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.

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 _____ ;
}
```

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.

9.2.8 Type8. Instruction Manipulations

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;
                }
            }
        }
```

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++) {
```

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).

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.

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.

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.

9.2.12 Type12. Transformation of a Solution

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

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.

9.2.13 Combining Several Types of Questions

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;
}
```

Here is a list of questions of different types that can be asked with respect to these methods separately or in any combination according to the teacher's pedagogical purposes.

1. *Type3. Tracing a given solution*: Trace each method when *n* is *19*.
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 estimatio*n: 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?

Activities 74 and 75 focus on the above collection of question types. The activities can be facilitated in different opportunities throughout the MTCS course, not necessarily only when the topic types of questions is at the focus of the discussion. The important thing is to keep mentioning that a variety of questions exists in CS education and keep emphasizing the importance of its use. For example, when students are asked to prepare questions and activities for CS learners on a specific CS topic, this variety of questions should be reminded and the students should be asked to use it.

Activity 74: Question Classification

This activity is based on three stages: individual work, group work, and a class discussion.

A collection of questions is given to the students together with a list of types of questions. The students are asked to classify the questions according to their types. Another option is to ask the students to classify the questions according to their criteria (and not to distribute the list of types of questions).

They can be asked first to work on the classification individually, and then to discuss their classification in small groups, trying to reach an agreement on one classification. Obviously, a question can be classified into two or more types of questions, but such cases, in fact, just sharpen the students' thinking about the nature of CS questions. After the group work, a discussion in the

course plenum takes place in which the classifications are presented and their rationales are articulated.

We note that in most CS textbooks at the end of each chapter, a set of questions is usually presented and the questions to be sorted can be taken from there. In this case, it is relevant to increase the students' awareness to the fact that they should examine also these sets of questions while choosing a text book to be used in their future filed work.

Activity 75: Classification of Nonsimple Questions

The students get a worksheet with patterns of questions, and are asked to work in groups and for each pattern, either to indicate its type (by using the list of types) or state that no type of questions fits it. In the second case, students are asked to formulate a new type of question and to indicate its pedagogical targets.

Table 9.1 presents a worksheet about the classification of question patterns.

Table 9.1 Worksheet about the classification of nonsimple questions

Worksheet—question classification
In what follows, a list of questions patterns is presented
For each question pattern determine its type; if you cannot find a suitable type, formulate a new type that captures its essence and indicate the pedagogical purposes of this type of questions
1. A trace table is given. Write a sequence of instructions that produces this trace table
2. In the computer lab: A given program contains a new built-in function called doSomething. Investigate the purpose of this function and formulate it. Document your investigation process
3. In what follows, several runtime errors are presented. For each of them, write a program that during its running, the error is received
4. A narrative-algorithmic question is presented. Choose the most appropriate data structure needed to solve the question, and explain your choice
5. In the computer lab: Play a given game and speculate what classes/data structures/functions used for its programming

9.3 Problem-Solving Questions[1]

The centrality of the problem-solving process was emphasized in Chap. 5, in which we offered different pedagogical tools to help learners acquire problem-solving strategies and gain experience in these skills.

[1] Based on Ragonis and Shilo (2013).

Table 9.2 Problem-solving question categories and keywords

Category	Keywords	Interpretation
1. Address/define criteria	Address/apply/note/mention/specify/indicate/sort/mark	Address and apply different kinds of criteria and attribute the criteria to a list of elements; define criteria
2. Argue and justify	Argue/state/assert/determine, follow by justification: explain/argue/prove/justify/demonstrate/illustrate/clarify	State opinion and further establish the claim using any kind of justification
3. Analyze	analyze/examine/investigate/explore	Identify and analyze the meaning or significance of components and factors
4. Compare	Compare/classify	Compare different objects/issues by applying principles, and observing from different view points; generalize insights
5. Complete	Complete/add	Complete or add components to a given structure according to detailed requirements
6. Convert	Convert/represent in different forms/modify/adjust/change/transform	Convert a given paragraph/section/clause according to specified, meaningful qualitative -related instructions (not technical translation)
7. Discover	Discover/identify/find out/say what	Discover a phenomenon/indicate an occurrence/find out the purpose/identify components and the relations between them
8. Develop	Develop/compose/write/create new elements	Develop a new component/write a new module
9. Integrate	Integrate/order/arrange/merge/combine	Integrate some given components to a new structure

Boyer et al. (2010) present recommended questions for instructors to use in class discussions during question-answer sessions, in order to promote students' problem-solving skills in CS classes. The main issue emphasized is the debate climate, in which answers are not given and questions are posed to encourage meaningful thinking. In a thorough analysis of problem-solving questions, based on an extensive review of textbooks, worksheets and exams, Ragonis and Shilo (2013) investigated keywords in questions. They especially looked for questions that their target, as characterized by the authors, is to examine high-order thinking skills.

The keywords used in problem-solving questions were grouped into nine categories, by the cognitive resources learner needs to apply in order to solve a question. Each category was then represented by several keywords. Table 9.2 displays the categories, the keywords that express each category and a short interpretation of their meaning and usage. The categorization reflected different point of view than the known cognitive taxonomies (e.g., Bloom taxonomy 1956; Biggs and Collis SOLO taxonomy 1982) .

Examples of questions for each of the nine categories are presented in the Ragonis and Shilo (2013).

Activity 76: Test Analysis
This activity can be facilitated in the context of any topic discussed in the course. As preparation to the activity, the course instructor collects examples of tests to be used in the group work. Also, Table 9.2 is distributed to the students.

The students are divided into groups and are asked to analyze the test questions by examining the cognitive skills a learner needs to apply in order to solve the questions. In this activity, students are asked to apply their critical thinking: They can add keywords to the categories and to discuss the differences between "problem-solving questions" to "non problem-solving question." They can also add a new category.

Discussion can be then facilitated in the course plenary on whether a specific content influences cognitive demands or not.

9.4 Kinds of Questions

In this section, we present three kinds of questions, which represent a more global question classification. The first two kinds—*story questions* and *closed questions*—can be applied with respect to most of the 12 types of questions presented in Sect. 9.2. The third kind of question addresses *unsolved problems*.

9.4.1 Story Questions

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

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

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

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.

9.4.2 Closed Questions

The common interpretation of a *closed question* is a question that is presented together with a list of possible answers and the learners' task is to choose the correct answer from this list. The frequent types of closed questions are multiple-choice questions or true/false questions. It should be remembered, though, that in fact, the answers are closed but not the questions.

The 12 question types presented in Sect. 9.2 could be further discussed as questions that can be presented as story questions and/or as closed questions. In what follows, we classify them according to this criterion.

Types of questions that can be presented as closed questions:
- Type3—Tracing a given solution
- Type4—Analysis of code execution
- Type5—Finding the purpose of a given solution
- Type6—Examination of the correctness of a given solution
- Type9—Efficiency estimation

Example: A closed question of Type6 can present a list of methods that potentially solve a given task, and ask learners to mark the methods that solve the task correctly.

Types of questions that cannot be presented as closed questions:
- Type1—Development of a solution
- Type2—Development of a solution that uses a given module
- Type10—Question design
- Type12—Transformation of a solution

Since the target of these types of questions is to require learners to develop a solution that meets specific requirements, if they become closed questions, this target will not be achieved.

Types of questions that cannot naturally be presented as closed questions:
- Type7—Completion of a given solution
- Type8—Instruction manipulations
- Type11—Programming style questions

Example: A closed question of Type7 can present a list of optional instructions to be added to a given code in a specific place in order to solve a given task, and learners are asked to mark which of them fits for this purpose.
The target of Activity 77 is to train the prospective CS teachers in question formulation. This training is especially important in the context of story questions which usually include more words.

In ITiCSE-13, a working group developed a set of 654 multiple-choice questions (a kind of closed question) on CS1 and CS2 topics called the Canterbury QuestionBank (Sanders et al. 2013). This work describes twelve patterns of multiple-choice questions, labels each one with a name, and adds a short description, a sample question(s), and an identification of which of the twelve types of questions, presented in this chapter, fits it the most (Sects. 9.2.1–9.2.12). The QuestionBank is publicly available as a repository for computing education instructors and researchers.

Activity 77: Question Formulation, Work in Pairs
First, the students work in pairs and each mate of the pair is asked to develop an open question on one specific topic from the curriculum. Then, they are asked to exchange the questions they write, solve the question that their mate develop, and give a constructive feedback about the question to their mate. It is important that both mates will develop a question on the same CS topic, since during the question development they deepen their understanding and consideration with respect to the specific topic, and consequently, their feedback to their mate is more valuable.

9.4.3 Unsolvable Questions

The rational for this specific short discussion stems from the fact that CS learners should be aware of the fact that not every problem is solvable (see also Chap. 5). There are incomputable problems—problems that have no solution, meaning, there is no algorithm that solves them, and further, a proof exists that shows that such algorithm does not exist. In addition, there are problems that have an algorithmic solution, but they cannot be computed in practice due to their time complexity.

It is important that a CS educator be aware to the fact that sometimes there is no hint at whether a problem is solvable or not. Further, from a high school CS teacher's point of view, the message that should be delivered is that any question design should be done very carefully. Sometimes questions may look simple, but their solution may be very complicated or even does not exist.

Hazzan (2001) addresses this kind of question by focusing on their presentation to CS learners. The idea is to formulate a problem in a way that would not give any hint of whether the problem is solvable or not, or of the conditions under which it is solvable. Such a formulation has two main merits. First, learners get the idea that there are unsolvable problems, and second, learners acquire skills for determining whether a problem is solvable or not, and if a problem is solvable under certain conditions, to find out these conditions. This idea is illustrated in Hazzan (2001) with respect to different problems, for example, the halting problem, the tiling problem, and map coloring problem.

9.5 Assimilation of the Types of Questions to Different Computer Science Contents

As we stated in the beginning of the chapter, the presented types of questions are mainly programming-related questions. Still, most of them can be used in a variety of CS contexts. Table 9.4 presents specific variations of several question types in the context of Automata theory.

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

The construction of such a table can be facilitated as an activity in the MTCS course for one of the topics included in the high school curriculum that the MTCS focuses on.

9.6 Question Preparation

The preparation of questions to be used in a CS class is not a simple task. Many considerations should be thought about, some of them are local to the specific lesson and class and others are more global and refer to the teaching unit, or even, further, to the entire curriculum. Those considerations are important and therefore, the pro-

cess of question preparation should be included in CS teacher preparation programs. Though an entire course can be dedicated to aspects related to question preparation in different stages of the teaching processes, we present here only the main stages of the preparation process of a question to be used in a CS class.

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.

The target of Activities 78 and 79 is to train the students in question design of different types. Additional activities about evaluation in general and about test preparation in particular are presented in Chap. 10.

Activity 78: Question Design, Individual Work or Group Work
All students/groups are directed to focus on one specific CS subject, such as, the loop or differences between indices and values of array cells. In addition, for each student/group, one type of question is assigned. Then, each student/group composes a question of that specific type related to the specified CS subject. The students can exchange the questions to get feedback, and based on it to improve the questions formulation.

In the next stage, all questions are merged into one document. Such a document highlights very clearly how a specific subject can be addressed by a variety of types of questions.

Further, the course instructor can facilitate a discussion about how to evaluate question difficulty, by posing questions such as: Can we definitely decide whether a question is simpler or more difficult than another question? What elements do determine question difficulty? Can we sort questions according to their difficulty level? Is question difficulty level connected directly to its type?

This discussion can follow by an activity in which for several of the presented questions, the students are asked to formulate two similar questions—one that is simpler and one that is harder. An interesting question that the students can be asked to reflect on is whether different levels of question difficulty require also different question types.

Activity 79: Test Design, Group Work

This activity can be facilitated in the context of any topic discussed in the course. The activity is presented in Table 9.5.

The students are divided into groups and are asked to compose a test on a specified CS subject taken from the high school curriculum that the MTCS course focuses on. Each question in the test should represent a different question type. While working on this activity, the student should relate to the different stages of question composition, to the variety of question types, and to the CS contents as well.

Table 9.5 Worksheet on test design

Worksheet—test design, group work
Compose a test on (conditions/loops/arrays/…) according to the following stages:
1. Determine which concepts should be tested
2. Decide what types of questions to include in the test. Associate the concepts to be checked to each type of question and roughly estimate the time required for a high school pupil to solve it
3. Distribute the work among the team members, so that each team member will compose one question
4. When all team members finish composing their questions, organize them into one test
5. Re-estimate the time required for a high school pupil to solve each question and check if the total time estimations fits the time framework of the test
6. Each team member solves the test and writes down his or her comments on each question
7. In your groups, discuss your comments and perform the needed changes in the test
8. Submit the test to the course web site

References

Biggs JB, Collis K (1982) Evaluating the quality of learning: the SOLO taxonomy. Academic Press, New York

Bloom BS, Engelhart MD, Furst EJ et al (1956) Taxonomy of educational objectives Handbook 1: cognitive domain. Longman Group Ltd, London

Boyer KE, Lahti W, Phillips R, Wallis MD, Vouk MA, Lester JC (2010) Principles of asking effective questions during student problem solving. Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10) ACM, New York, USA, pp 460–464

Hazzan O (2001) On the presentation of computer science problems. Inroads—the SIGCSE Bull 33(4):55–58

Jones KO, Harland J, Reid JM et al (2009) Relationship between examination questions and Bloom's taxonomy. In Proceedings of the 39th IEEE International Conference on Frontiers in Education Conference San Antonio, Texas, USA

Pomerantz J (2002) Question types in digital reference: an evaluation of question taxonomies. In Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Library. Portland, Oregon, USA

Ragonis N, Shilo G. (2013) What is it we are asking: Interpreting problem-solving questions in computer science and linguistics. In Proc. of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13). ACM, New York, USA, pp 189–194

Sanders K, Ahmadzadeh M, Clear T, Edwards SH, Goldweber M, Johnson C, Lister R, McCartney R, Patitsas E, Spacco J (2013) The canterbury questionbank: building a repository of multiple-choice CS1 and CS2 questions. In Proceedings of the ITiCSE working group report Conference on Innovation and Technical in Computer Science Education -working group reports (ITiCSE -WGR '13). ACM, New York, USA, pp 33–52

Thompson E, Luxton-R A, Whalley J et al (2008) Bloom's taxonomy for CS assessment. In Simon, Hamilton, M (Eds), Proc 10th Australas Comput Educ Conf (ACE 2008), vol 78. Wollongong, NSW, Australia. CRPIT, pp 155–162

Yang A., Wu J, Wang L (2008) Research and design of test question database management system based on the three-tier structure. WTOS 7(12):1473–1483