Exercise questions and answers, Algorithm

Questions

Q1: By definition, algorithms need to define a terminating process. What are some examples to algorithms which are non-terminating processes?

Q2: Why do we use pseudocode?

Q3: Is the term *algorithm* specific to computer science?

Q4: What are the basic principles of problem solving? (Hint: Defined by G.Polya)

Q5: What is a loop? Give some examples to commands we use to represent loops?

Q6: What is the difference between *break* and *continue*?

Q7: Give two examples to naming conventions. Give these examples by using the convention itself.

Q8: What is time complexity of an algorithm? How is it expressed?

Q9: What is brute-force algorithm? What is a disadvantage of it?

Q10: What is recursion and why is it used?

Answers

A1: **Monitoring** vital signs of a patient or **maintaining** the altitude of an aircraft.

A2: It describes the steps of the algorithm in plain language. If we use a programming language syntax, it is possible that some people may not understand them. That is why the algorithm is defined in plain language. Also, sometimes the algorithm can be large enough so that it would be hard to be shown with flowcharts.

A3: No. This term was found way before computers. It actually gives a recipe on how to do things. A recipe on how to cook can also be an algorithm. It is not limited to computer science.

A4: 1. Understand the problem. 2. Get an idea of how an algorithmic function might solve the problem. 3. Formulate the algorithm and represent it as a program. 4. Evaluate the program for accuracy and its potential as a tool for solving other problems.

A5: A loop is an implementation of repetition. While and for commands are use to implement loops.

A6: When the program sees break command, it will get out of the loop. In continue, it will not run the next command but will start with the next iteration.

A7: camelCase, snake_case, PascalCase, kebab-case.

A8: It is defined by the time it consumes. There are three: best, average or worst. We usually deal with *worst case complexity* because that is the longest time it will take. It is expressed by Big-Oh notation. An example is O(n).

A9: It is a method where the solution is found by trying every possible scenario. That is why, it can take too much time, it is an *inefficient* algorithm. It relies on computing power, not the algorithm quality.

A10: Recursion is a method of solving a computational problem where the solution depends on solutions of smaller instances of the same problem. This is implemented as a function calling itself. It is generally use when writing iterative code is very complex. Usually recursion do not provide any time or space complexity benefits.