▲Algorithms

► Activity 3: Choose the correct option:

Can you think of some everyday activities that **(1) involve / evolve** algorithms? Well, actually, they are countless; they simply go **(2) with / by** different names. Algorithms for cooking, for example, are commonly called *recipes*, for playing music are called *sheet music*, and for finding our way through a city we have never visited before are called *directions*.

It is **(3) evident / subsequent** from the above that an algorithm is a series of steps, or a **(4) final / finite** sequence of directions, which defines how a task is performed. The study of algorithms has existed since antiquity. Indeed, long before the emergence of the computer, mathematicians searched for algorithms in their **(5) attempt / release** to find a single set of instructions that described how all problems of a particular type could be solved. Examples of this early research include the long division algorithm for finding the quotient of two multiple-digit numbers or the algorithm discovered by the Ancient Greek mathematician Euclid (the world-famous *Euclidean algorithm*) for finding the greatest common divisor of two positive integers.

For more on Maths, see Unit 12.

The algorithm is the most **(6) dominant / fundamental** concept of computer science. In order for a computer to perform a task, an algorithm for this particular task must be discovered and represented in a form **(7) comparable / compatible** with the technology of the computer. A representation of an algorithm is called a *program*. The process of developing a program, encoding it, and inserting it into a computer in a language that it can understand is called *programming*, or *coding*. Programs, together with the algorithms they represent, are collectively referred to as **(8) *firmware / software***.

It goes without saying that a machine is capable of **(9) accomplishing / initiating** a particular task or performing other problem-solving operations only if an algorithm exists for that task or problem. Once an algorithm has been found, the performance of the task or the solution of the problem no longer requires an understanding of the **(10) geeks / principles** according to which the algorithm has been developed. To use the same example as above, we can follow the Euclidean algorithm to find the greatest common divisor of two positive integers without understanding why or how the algorithm works.

As mentioned above, the study of algorithms as a subject in mathematics started a long time before the development of today's computers. However, identifying the **(11) limitations / shortcomings** of algorithmic capabilities was **(12) solidified / modified** as a subject in mathematics in the 1930s, when Austrian logician Kurt Gödel published his incompleteness theorem, which states that in any mathematical theory, there will always be statements whose truth or **(13) transparency / falseness** cannot be established by algorithmic means. Gödel's theorem, which is widely considered one of the greatest intellectual achievements of modern times, shook the foundations of mathematics, and the study of algorithmic capabilities that ensued paved the way for the development of a field known today as *computer science*.

What are the most important **(14) suites / features** of an algorithm? First of all, unambiguity: a perfect algorithm should be unambiguous, which means that its instructions should be clear and precise. Secondly, finiteness: an algorithm should be finite, or, in other words, should have a limited **(15) number / bundle** of instructions. Last, but not least, effectiveness: each guideline in an algorithm should be adequate, as it will affect the overall process. Are you now ready to write one?