# GCSE COMPUTER SCIENCE 8520/CA/CB/CC/CD/CE

Sample NEA Task (Task 1- Area Trainer)

V1.1

For candidates entering for the XXXX examination.
To be issued to candidates at the start of the final academic year of their course of study.

**Instructions**

- Evidence for assessment must include a complete listing of all program code, and a report describing the design of the solution, any features of the coded solution which are not evident from the listing, the testing and any potential enhancements and refinements to the solution.

**Information**

- The assessment is designed to be completed in 20 hours.
- The assessment period is not required to be continuous.
- There are restrictions on when and where students can work on this problem. Please see the Teachers' Notes which accompany this task for more information about these restrictions.
- Students may need to use the Internet to research certain parts of the problem. This must be within the 20 hours.
- Submission may be paper-based or electronic using CD/DVD.
- Students will need to complete and sign a Candidate Record Form which declares that the work is their own. This must be countersigned by the teacher and a member of the senior leadership team at your school or college.

Students must use one of the following programming languages:

- C#
- Java
- Pascal/Delphi
- Python
- VB.Net.

Centres will be asked to indicate their programming language choice(s) for each cohort at the start of the course.

## **Problem:** Area Trainer

The aim of this activity is to produce a computer program which will help students in year 9 practise working out the area of shapes.

The teacher wants a way of recording when each student has used the program and so students must login to the program when they run it. The program must work in the following way:

1. A student registers by choosing:
    a. a username
    b. and creating a password, of appropriate strength.
2. Once logged in, a menu of shapes is displayed from which the student can choose the shape they wish to practise. Shapes could include a triangle, rectangle, or circle.
3. The dimensions of the chosen shape are displayed, along with four possible values for the area. One of the possible values is the correct answer, and the other three are incorrect.
4. The student must choose **one** value for the area.
5. The scoring must work as follows:
    a. If the student chooses the correct answer they score 2 points
    b. If they choose one of the incorrect answers, a feedback comment is displayed and they have another chance
    c. If the student chooses the correct answer this time, they score 1 point
    d. If they choose an incorrect answer again, a feedback comment is displayed together with the correct answer, and the option of answering another question, or quitting the trainer.
6. The student's total score for the session should be recorded in an external file along with the scores from previous sessions.
7. When quitting the trainer, the student's total score for the current session is displayed and they are offered the following options:
    a. starting again
    b. looking at the record of their scores from their previous sessions
    c. closing the program.

## Notes:

The area, *a*, of rectangle, which is *l* centimetres long by *w* centimetres wide is

$$a = l * w$$

The area, *a*, of a triangle, with a base of length *b* centimetres and a height of *h* centimetres is

$$a = {}^{1}\!/_{2}\, b * h$$

The area, *a*, of a circle of radius r centimetres is

$$a = \pi r^2$$

Students are free to use other shapes if they wish. It is acceptable for teachers to provide the area formulae for shapes if students require them.

When creating a solution students should demonstrate their use of the following programming skills where appropriate:

a. Validation of user inputs
b. The use of random numbers
c. File input/output to handle the storage of usernames and passwords and of scores.
d. Considering password strength
e. Use of a graphical display (although this is not essential)
f. Interactive program control
g. Well-designed, structured code using subroutines and interfaces
h. Error handling techniques.