MODULE O ROBUST PROGRAMS · O1

# Defensive design: anticipating misuse

Original practice questions · 21 marks · ~25 minutes · OCR J277 spec 2.3.1

Name _____ Date _____

**Instructions.** Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. All code on this sheet is written in OCR Exam Reference Language. Where you write code, you may use OCR Exam Reference Language or a high-level programming language that you have studied. The total for this worksheet is **21 marks**.

## 1

**Total: 4 marks**

Anticipating misuse is one of the defensive design considerations a programmer should think about.

**(a)** State what is meant by **anticipating misuse** when designing a program. **[2]**

**(b)** A council's staff portal asks each member of staff for a password. Someone may try to get into another person's account by guessing the password over and over again.
Describe **one** way the programmer could design the program to anticipate this misuse. **[2]**

**2** **Total: 6 marks**

A bike hire kiosk at a railway station runs a program. The customer is asked to enter:
- the number of hours of hire: a whole number from 1 to 8
- the type of bike, chosen from a menu: 1 = city bike, 2 = electric bike, 3 = tandem
- their surname, so the booking can be found later.

**(a)** Complete the table. For each input, give **one** example of an invalid value that a customer might enter and that the program should anticipate. **[3]**

| Input | Example of an invalid value |
|---|---|
| Number of hours of hire | |
| Type of bike (menu choice) | |
| Surname | |

**(b)** Describe how the program should deal with an invalid value when a customer enters one. **[3]**

**3** **Total: 6 marks**

An environmental agency measures the depth of a river in metres at several points. This program works out the average depth.

```
01 total = 0
02 count = int(input("How many readings?"))
03 for i = 1 to count
04     depth = float(input("Enter depth in metres"))
05     total = total + depth
06 next i
07 average = total / count
08 print("Average depth: " + str(average))
```

**(a)** State the value of `count` that would make line 07 stop the program with an error. **[1]**

**(b)** Give **one other** value that a user could enter for `count` that the program does not anticipate. **[1]**

**(c)** Write code to replace line 02. Your code must: **[4]**

- ask the user how many readings there are
- if the value is less than 1, output an error message and ask again
- keep asking, as many times as needed, until a value of 1 or more is entered.

You may assume that the user always enters a whole number.

**4** **Total: 5 marks**

A snack vending machine has six slots, numbered 1 to 6 on the front of the machine. The names of the snacks are stored in an array called `snacks`. The array is 0-indexed, so the snack in slot 1 is stored in `snacks[0]`.

```
01 array snacks = ["Crisps", "Apple", "Flapjack", "Nuts", "Raisins", "Popcorn"]
02 slot = int(input("Enter slot number 1 to 6"))
03 print("You chose " + snacks[slot - 1])
```

**(a)** A customer enters **7**. Explain why line 03 would cause an error. **[2]**

**(b)** Write code to replace line 03 so that the program: **[3]**

- outputs the name of the chosen snack only if the slot number is from 1 to 6
- outputs an error message for any other slot number.

**Total for this worksheet: 21 marks**