

# Answers to Student's Book

Exam-style questions and sample answers have been written by the authors. References to assessment and/or assessment preparation are the publisher's interpretation of the syllabus requirements and may not fully reflect the approach of Cambridge Assessment International Education. Cambridge International recommends that teachers consider using a range of teaching and learning resources in preparing learners for assessment, based on their own professional judgement of their students' needs.

Cambridge Assessment International Education bears no responsibility for the example answers to questions taken from its past question papers which are contained in this publication.

The questions, example answers, marks awarded and/or comments that appear in this digital material were written by the author(s). In examination, the way marks would be awarded to answers like these may be different.

## 7 Algorithm design and problem solving

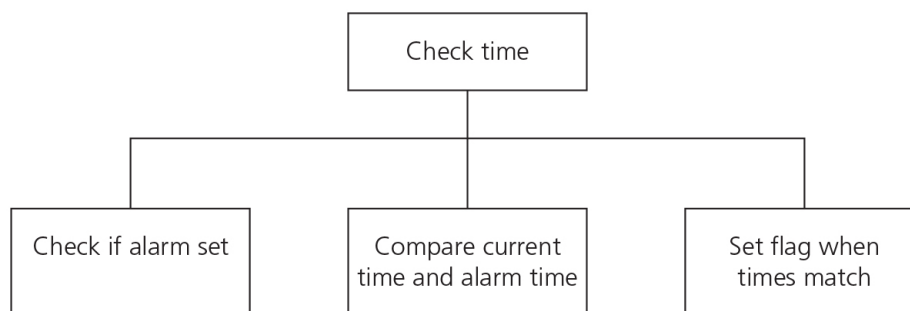
### Answers to activities

#### 7.1

For example, for a step tracking app:

- Input            a 3-axis accelerometer senses my steps
- Process        totals the steps for each day
- Output        the total number of steps taken so far
- Storage        the number of steps taken each day

#### 7.2



#### 7.3

```

Amount  100
TotalPrice  350
Discount  0.2
FinalPrice  280
Name  "Nikki"
Message  "Hello Nikki"
  
```

#### 7.4

```

OUTPUT "Please enter a mark "
INPUT Mark
  
```

```

IF Mark < 0 OR Mark > 20
    THEN
        OUTPUT "Invalid Mark"
    ELSE
        IF Mark > 9
            THEN
                OUTPUT "Pass"
            ELSE
                OUTPUT "Fail"
        ENDIF
    ENDIF
ENDIF

```

## 7.5

```

CASE OF Day
    1 : OUTPUT "Sunday"
    2 : OUTPUT "Monday"
    3 : OUTPUT "Tuesday"
    4 : OUTPUT "Wednesday"
    5 : OUTPUT "Thursday"
    6 : OUTPUT "Friday"
    7 : OUTPUT "Saturday"
    OTHERWISE OUTPUT "Error in day number"
ENDCASE

```

## 7.6

Purpose: Display the largest number

Processes:

- Input two numbers
- Compare the two numbers
- Display the largest number with a suitable message

Output if 7 and 18 input:            18 is the largest

## 7.7

- 1 Telephone number – presence check (to ensure something is entered), length check (to check the number is not too long or short), or format check (to make sure number matches the style of national telephone numbers).

Pupil's name – presence check (to ensure something is entered) and length check (i.e. between 2 and 30 letters, to check the name is of a reasonable length).

Part number – presence check (to ensure something is entered) and format check (to make sure data matches the character/number style XXX999).

- 2 OUTPUT "Please enter age of child "

```
INPUT ChildAge
```

```
OUTPUT "Please enter height of child in centimetres"
```

```
INPUT ChildHeight
```

```

IF ChildAge < 8 OR ChildAge > 11 OR ChildHeight < 110 OR
  ChildHeight > 150
THEN
  IF ChildAge < 8
    THEN
      OUTPUT "Too young"
    ENDIF
  IF ChildAge > 11
    THEN
      OUTPUT "Too old"
    ENDIF
  IF ChildHeight < 110
    THEN
      OUTPUT "Too small"
    ENDIF
  IF ChildHeight > 150
    THEN
      OUTPUT "Too tall"
    ENDIF
  ELSE
    OUTPUT "Child can ride"
  ENDIF
3 OUTPUT "Please enter your password "
REPEAT
  INPUT Password
  IF LENGTH(Password) > 12 OR LENGTH(Password) < 8
    THEN
      IF LENGTH(Password) > 12
        THEN
          OUTPUT "Password is too long, please re-enter "
        ELSE
          OUTPUT "Password is too short, please re-enter "
        ENDIF
      ENDIF
    ENDIF
  UNTIL LENGTH(Password) <= 12 AND LENGTH(Password) >= 8

```

**7.8**

Telephone number – the number needs to be the one that is required as well as having the necessary characteristics.

Pupil's name – the name needs to be the one that is associated with the pupil.

Part number – a part number could have the required format but not be for the part intended.

**7.9**

For example:

Normal test data: 7, 34, 55, 67, 74, 83, 21, 45, 72, 64; Expected result: 54.2

(Other values acceptable.)

**7.10**

For example:

Erroneous test data: -7, 1001; Expected result: both values rejected

(Other values acceptable.)

**7.11**

**1** Boundary test data for 100 is: 100, 101; Expected result: 100 accepted, 101 rejected

**2** Normal test data: 2, 14, 15, 6, 7, 13, 20, 4, 10, 9; Expected result: 10

Normal test data: 17, 13, 15, 15, 14, 16, 2, 8, 7, 6; Expected result: 11.3

Erroneous test data: -7, 23; Expected result: both values rejected

Boundary test data: -1, 0 and 20, 21

Expected results: -1 rejected, 0 accepted and 20 accepted, 21 rejected

**7.12**

Answer in chapter (page 285).

## 7.13

| A  | B  | C   | X  | OUTPUT                |
|----|----|-----|----|-----------------------|
| 0  | 0  | 100 |    | Enter your ten values |
| 1  | 4  | 4   | 4  |                       |
| 2  | 8  |     | 8  |                       |
| 3  | 19 |     | 19 |                       |
| 4  |    |     | 17 |                       |
| 5  |    | 3   | 3  |                       |
| 6  |    |     | 11 |                       |
| 7  |    |     | 6  |                       |
| 8  |    | 1   | 1  |                       |
| 9  |    |     | 13 |                       |
| 10 |    |     | 9  | 19 1                  |

7.14

| First | Last | Index | Swap  | Temp | Temperature |     |     |     |     |     |     |     |
|-------|------|-------|-------|------|-------------|-----|-----|-----|-----|-----|-----|-----|
|       |      |       |       |      | [1]         | [2] | [3] | [4] | [5] | [6] | [7] | [8] |
| 1     | 8    | 1     | FALSE |      | 35          | 31  | 32  | 36  | 39  | 37  | 42  | 38  |
|       |      |       | TRUE  | 35   | 31          | 35  |     |     |     |     |     |     |
|       |      | 2     | TRUE  | 35   |             | 32  | 35  |     |     |     |     |     |
|       |      | 3     |       |      |             |     |     |     |     |     |     |     |
|       |      | 4     |       |      |             |     |     |     |     |     |     |     |
|       |      | 5     | 39    | 39   |             |     |     |     | 37  | 39  |     |     |
|       |      | 6     |       |      |             |     |     |     |     |     |     |     |
|       |      | 7     |       | 42   |             |     |     |     |     |     | 38  | 42  |
|       | 7    | 1     | FALSE |      |             |     |     |     |     |     |     |     |
|       |      | 2     |       |      |             |     |     |     |     |     |     |     |
|       |      | 3     |       |      |             |     |     |     |     |     |     |     |
|       |      | 4     |       |      |             |     |     |     |     |     |     |     |
|       |      | 5     |       |      |             |     |     |     |     |     |     |     |
|       |      | 6     | TRUE  | 39   |             |     |     |     |     | 38  | 39  |     |
|       | 6    | 1     | FALSE |      |             |     |     |     |     |     |     |     |
|       |      | 2     |       |      |             |     |     |     |     |     |     |     |
|       |      | 3     |       |      |             |     |     |     |     |     |     |     |
|       |      | 4     |       |      |             |     |     |     |     |     |     |     |
|       |      | 5     |       |      |             |     |     |     |     |     |     |     |
|       | 5    |       |       |      |             |     |     |     |     |     |     |     |

## 7.15

Sample data: -1, -7, -11, -19, -3, -20, -9, -4, -6, -8

| A  | B | C   | X   | OUTPUT |
|----|---|-----|-----|--------|
| 0  | 0 | 100 |     |        |
| 1  |   |     | -1  |        |
| 2  |   | -7  | -7  |        |
| 3  |   | -11 | -11 |        |
| 4  |   | -19 | -19 |        |
| 5  |   |     | -3  |        |
| 6  |   | -20 | -20 |        |
| 7  |   |     | -9  |        |
| 8  |   |     | -4  |        |
| 9  |   |     | -6  |        |
| 10 | 0 |     | -3  |        |
|    |   |     |     | 0 -20  |

The highest value is -1 not zero.

## 7.16

-1000001(or any number less than this) and 1000001(or any number greater than this) will fail.

## 7.17

```

A ← 0
OUTPUT "Enter your ten values"
INPUT X
B ← X
C ← X
REPEAT
  INPUT X
  IF X > B
    THEN
      B ← X
  ENDIF
  IF X < C
    THEN
      C ← X
  ENDIF
  A ← A + 1
UNTIL A = 9
OUTPUT B, C

```

| A | B     | C   | X     | OUTPUT     |
|---|-------|-----|-------|------------|
| 0 | -97   | -97 | -97   |            |
| 1 | 12390 |     | 12390 |            |
| 2 |       |     | 0     |            |
| 3 |       |     | 77    |            |
| 4 |       |     | 359   |            |
| 5 |       |     | -2    |            |
| 6 |       |     | -89   |            |
| 7 |       |     | 5000  |            |
| 8 |       |     | 21    |            |
| 9 |       |     | 67    |            |
|   |       |     |       | 12390, -97 |

**7.18**

0, 26

Expected results rejected – erroneous data

1, 25

Expected results: Your tickets cost 20

Your tickets cost 400 – **extreme data**

9, 10

Expected results: Your tickets cost 180

Your tickets cost 180 – **normal data**

19, 20

Expected results: Your tickets cost 342

Your tickets cost 320 – **normal data**

Extra test data:

0, 1

Expected results: 0 rejected

1 Your tickets cost 20 – **boundary data lower bound**

26, 25

Expected results: 26 rejected

25 Your tickets cost 400 – **boundary data upper bound**

-30, 200

Expected results rejected – erroneous data

## 7.19

```

1  Change FOR Test ← 1 to 4  // outer loop for the tests
   to FOR Test ← 1 to 2  // outer loop for the tests
   remove 3 : SubjectName ← "English" and 4 : SubjectName ← "IT"
   change FOR StudentNumber ← 1 TO 600
   to FOR StudentNumber ← 1 TO 5

   change SubjectAverage ← SubjectTotal / 600
   to SubjectAverage ← SubjectTotal / 5
   change OverallAverage ← OverallTotal / 2400
   to OverallAverage ← OverallTotal / 10

```

The code should now look like this:

```

// initialisation of overall counters
OverallHighest ← 0
OverallLowest ← 100
OverallTotal ← 0
FOR Test ← 1 to 2  // outer loop for the tests
// initialisation of subject counters
  SubjectHighest ← 0
  SubjectLowest ← 100
  SubjectTotal ← 0
  CASE OF Test
    1 : SubjectName ← "Maths"
    2 : SubjectName ← "Science"
  ENDCASE
  FOR StudentNumber ← 1 TO 5  // inner loop for the students
    REPEAT
      OUTPUT "Enter Student", StudentNumber, "'s mark for ", SubjectName
      INPUT Mark
    UNTIL Mark < 101 AND Mark > -1
    IF Mark < OverallLowest THEN OverallLowest ← Mark
    IF Mark < SubjectLowest THEN SubjectLowest ← Mark
    IF Mark > OverallHighest THEN OverallHighest ← Mark
    IF Mark > SubjectHighest THEN SubjectHighest ← Mark
    OverallTotal ← OverallTotal + Mark
    SubjectTotal ← SubjectTotal + Mark
  NEXT StudentNumber
  SubjectAverage ← SubjectTotal / 5
  OUTPUT SubjectName
  OUTPUT "Average mark is ", SubjectAverage
  OUTPUT "Highest Mark is ", SubjectHighest
  OUTPUT "Lowest Mark is ", SubjectLowest
NEXT Test
OverallAverage ← OverallTotal / 10
OUTPUT "Overall Average is ", OverallAverage
OUTPUT "Overall Highest Mark is ", OverallHighest
OUTPUT "Overall Lowest Mark is ", OverallLowest

```

Test data (sample – yours might be different): 1, 99, -1, 101, 20, 80, 30, 70, 40, 60, 50, 50

2 Trace table:

| Test | Student Number | Mark | Subject Name | Subject |      |     |       | Overall |      |     |       | Output                                                                               |
|------|----------------|------|--------------|---------|------|-----|-------|---------|------|-----|-------|--------------------------------------------------------------------------------------|
|      |                |      |              | Av      | High | Low | Total | Av      | High | Low | Total |                                                                                      |
|      |                |      |              |         |      |     |       |         | 0    | 100 | 0     |                                                                                      |
| 1    |                |      |              |         | 0    | 100 | 0     |         |      |     |       |                                                                                      |
|      | 1              | 1    |              |         | 1    | 1   | 1     |         | 1    | 1   | 1     | Enter student 1's mark for Maths                                                     |
|      | 2              | 99   |              |         | 99   |     | 100   |         | 99   |     | 100   | Enter student 2's mark for Maths                                                     |
|      | 3              | -1   |              |         |      |     |       |         |      |     |       | Enter student 3's mark for Maths                                                     |
|      |                | 101  |              |         |      |     |       |         |      |     |       | Enter student 3's mark for Maths                                                     |
|      |                | 20   |              |         |      |     | 120   |         |      |     | 120   | Enter student 3's mark for Maths                                                     |
|      | 4              | 80   |              |         |      |     | 200   |         |      |     | 200   | Enter student 4's mark for Maths                                                     |
|      | 5              | 30   |              |         |      |     | 230   |         |      |     | 230   | Enter student 5's mark for Maths                                                     |
|      |                |      |              | 46      |      |     |       |         |      |     |       | Maths<br>Average mark is 46<br>Highest mark is 99<br>Lowest mark is 1                |
| 2    |                |      |              |         | 0    | 100 | 0     |         |      |     |       |                                                                                      |
|      |                | 1    | 70           |         | 70   | 70  | 70    |         |      |     | 300   | Enter student 1's mark for English                                                   |
|      |                | 2    | 40           |         |      | 40  | 110   |         |      |     | 340   | Enter student 2's mark for English                                                   |
|      |                | 3    | 60           |         |      | 60  | 170   |         |      |     | 400   | Enter student 3's mark for English                                                   |
|      |                | 4    | 50           |         |      | 50  | 220   |         |      |     | 450   | Enter student 4's mark for English                                                   |
|      |                | 5    | 50           |         |      | 50  | 270   |         |      |     | 500   | Enter student 5's mark for English                                                   |
|      |                |      |              | 54      |      |     |       |         |      |     |       | English<br>Average mark is 54<br>Highest mark is 70<br>Lowest mark is 40             |
|      |                |      |              |         |      |     |       | 50      |      |     |       | Overall average mark is 50<br>Overall highest mark is 99<br>Overall lowest mark is 1 |

## 7.20

1 Total  $\leftarrow$  0

FOR Counter  $\leftarrow$  1 TO 10

REPEAT

    OUTPUT "Enter positive number "

    INPUT Number

UNTIL Number > 0

    Total  $\leftarrow$  Total + Number

NEXT Counter

Average  $\leftarrow$  Total / 10

OUTPUT "Average is ", Average

OUTPUT "Total is ", Total

2 Total  $\leftarrow$  0

Counter  $\leftarrow$  0

REPEAT

    REPEAT

        OUTPUT "Enter positive number, -1 to finish "

        INPUT Number

    UNTIL Number > 0 OR Number = -1

    IF Number <> -1

        THEN

            Total  $\leftarrow$  Total + Number

            Counter  $\leftarrow$  Counter + 1

        ENDIF

UNTIL Number = -1

Average  $\leftarrow$  Total / Counter

OUTPUT "Average is ", Average

OUTPUT "Total is ", Total

3 For 1, I chose a FOR ... NEXT loop because it has a fixed number of iterations.

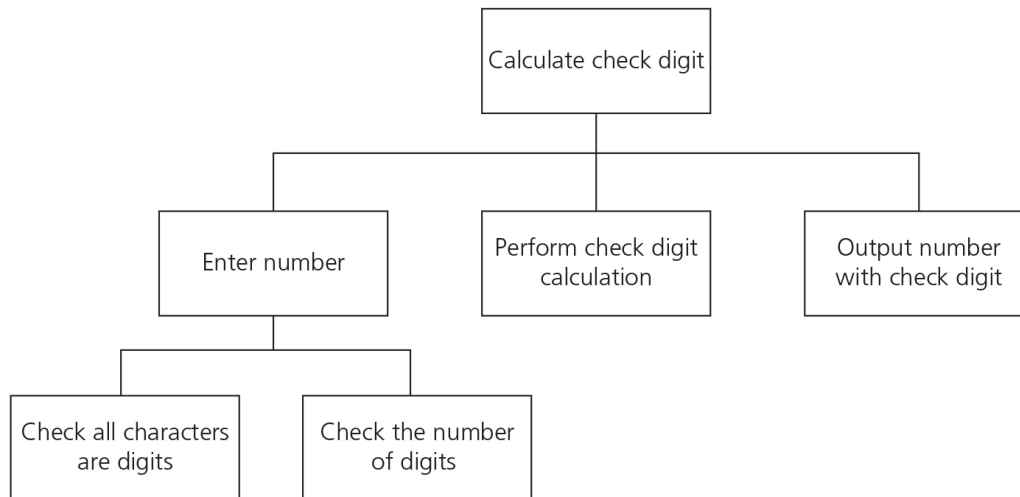
For 2, I chose a REPEAT ... UNTIL loop because there must be at least one number in the list but the number of iterations is unknown.

### Answers to exam-style questions

*Answers to questions 8 and 9 in this section are example answers for past paper questions. The details of the relevant past paper can be found with the corresponding exam-style question in the Student's Book.*

- 1
  - Inputs – the data used by the system that needs to be entered while the system is active
  - Processes – the tasks that need to be performed using the input data and any other previously stored data
  - Outputs – information that needs to be displayed or printed for the users of the system
  - Storage – data that needs to be stored in files on an appropriate media for use in the future

2



3 a Diners – range check 2 to 12 inclusive, presence check and type check of integer

Bill – range check 10 to 500 inclusive, presence check and type check of real

b Test data (sample – yours might be different):

- Normal data: 5 diners and a bill of \$90.00  
Expected result: \$18.00
- Normal data: 8 diners and a bill of \$69.00  
Expected result: \$8.63

c Test data (sample – yours might be different):

- Abnormal data: 15 diners and a bill of \$5.00  
Expected result: both values rejected

d • Boundary data for lower bounds 1 diner and 2 diners and bills of \$9.99 and \$10.00  
Expected result: 1 and \$9.99 both values rejected, 2 and \$10.00 \$5.00

- Boundary data for upper bounds 12 diners and 13 diners and bills of \$500.00 and \$500.01  
Expected result: 13 and \$500.01 both values rejected, 12 and \$500.00 \$41.67

4 Validation – the automated checking by a program that data is reasonable before it is accepted into a computer system

Verification – checking that data has been accurately copied from another source and input into a computer or transferred from one part of a computer system to another

5 **Name:**

Validation checks:

- presence to ensure name is entered
- length check to ensure there are not too many letters

Verification check – yes to ensure that the name entered is the one intended and there are no typing errors

**Date of birth:**

Validation checks:

- presence to ensure date of birth is entered
- format check to ensure that all the components of the data are in range
- range check to ensure that the value for the date is realistic

Verification check – yes to ensure that the date entered is the one intended and there are no typing errors

**Password:**

Validation checks:

- presence to ensure password is entered
- length check to ensure there are the appropriate number of characters
- format check for any special rules, for example at least one digit

Verification check – yes to ensure that the password entered is the one intended and there are no typing errors

**Phone number:**

Validation checks:

- presence to ensure phone number is entered
- length check to ensure there are the appropriate number of digits/characters
- format check for any special rules, for example must start with +44

Verification check – yes to ensure that the phone number entered is the one intended and there are no typing errors

**6 a**

| Counter | Accept | Reject | Length | Breadth | Size | OUTPUT |
|---------|--------|--------|--------|---------|------|--------|
| 0       | 0      | 0      |        |         |      |        |
| 1       | 1      |        | 15     | 10      | 150  |        |
| 2       | 2      |        | 20     | 17      | 340  |        |
| 3       |        | 1      | 32     | 10      |      |        |
| 4       |        | 2      | 30     | 35      |      |        |
| 5       | 3      |        | 30     | 15      | 450  |        |
| 6       |        | 3      | 30     | 28      | 840  |        |
| 7       |        | 4      | 25     | 25      | 625  |        |
| 8       | 4      |        | 20     | 15      | 300  |        |
| 9       |        | 5      | 40     | 20      | 800  |        |
| 10      | 5      |        | 12     | 10      | 120  | 5 5    |

- b**
- input length and breadth of 10 parcels
  - check length and breadth less than 30 for each parcel
  - calculate size and check size less than 600 for each parcel
  - increment the number of parcels accepted or the number of parcels rejected
  - output number of parcels accepted and number of parcels rejected
- c** Length and breadth must be less than or equal to 30. Size must be less than or equal to 600.
- 7 a**
- Error 1: Counter  $\leftarrow$  1 should be Total  $\leftarrow$  0
  - Error 2: UNTIL Number < 0 should be UNTIL Number > 0
  - Error 3: Total  $\leftarrow$  Total + Counter should be Total  $\leftarrow$  Total + Number
  - Error 4: Counter  $\leftarrow$  Counter + 1 is not required
  - Error 5: OUTPUT Total should be after the end of the loop
  - Error 6: NEXT Number should be NEXT Counter

**b**

```

Total  $\leftarrow$  0
FOR Counter  $\leftarrow$  1 TO 10
    REPEAT
        OUTPUT "Enter a positive whole number "
        INPUT Number
    UNTIL Number > 0
    Total  $\leftarrow$  Total + Number
NEXT Counter
OUTPUT Total

```

- c** Test data: 1, 0, 67, 10, 22, -10, 3, 7, 2, 8, 1, 9 (sample – yours might be different).

| Counter | Number | Total | OUTPUT |
|---------|--------|-------|--------|
|         |        | 0     |        |
| 1       | 1      | 1     |        |
| 2       | 0      |       |        |
|         | 67     | 68    |        |
| 3       | 10     | 78    |        |
| 4       | 22     | 100   |        |
| 5       | -10    |       |        |
|         | 3      | 103   |        |
| 6       | 7      | 110   |        |
| 7       | 2      | 112   |        |
| 8       | 8      | 120   |        |
| 9       | 1      | 121   |        |
| 10      | 9      | 130   | 130    |

- d** Test data: 1, 0, 67, 10, 22, -10, 3, 7, 2, 8, 1, 9 (sample – yours might be different).  
 1 extreme, 0 erroneous, 67 normal, 10 normal, 22 normal, -10 erroneous, 3 normal, 7 normal, 2 normal, 8 normal, 1 normal and 9 normal.

**8 a**

| Number1 | Number2 | Sign | Answer | OUTPUT |
|---------|---------|------|--------|--------|
| 5       | 7       | +    | 12     | 12     |
| 6       | 2       | -    | 4      | 4      |
| 4       | 3       | *    | 12     | 12     |
| 7       | 8       | ?    | 0      |        |
| 0       | 0       | /    | (0)    |        |

- b** CASE Sign OF  
     '+' : Answer ← Number1 + Number2  
     '-' : Answer ← Number1 - Number2  
     '\*' : Answer ← Number1 \* Number2  
     '/' : Answer ← Number1 / Number2  
     OTHERWISE Answer ← 0  
 ENDCASE

- 9 a** To ensure no changes are made on input, because the details do not have fixed values and there is no clear set of rules that can be used for validation.  
 The programmer could ask the contributor to type in each detail twice and then check that both values are equal. If they are not equal, then the input should be rejected.
- b** Email – check for @ to ensure that it is an email address. Password – length check to ensure sufficient characters have been entered.