Cambridge Assessment International Education

Cambridge IGCSE™ (9–1)

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

COMPUTER SCIENCE **0984/22**

Paper 2 Problem-solving and Programming **October/November 2022**

1 hour 45 minutes

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS

- Answer **all** questions.
- **Do not attempt Tasks 1, 2 and 3** in the copy of the pre-release material on page 2; these are for information only.
- Use a black or dark blue pen. You may use an HB pencil for any diagrams or graphs.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- Calculators must **not** be used in this paper.

INFORMATION

- The total mark for this paper is 50.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **16** pages. Any blank pages are indicated.

DC (CJ/CB) 315858

[Turn over

Section A

You are advised to spend no longer than 40 minutes answering this section.

Here is a copy of the pre-release material.

DO NOT attempt Tasks 1, 2 and 3 now.

Use the pre-release material and your experience from attempting the following tasks before the examination to answer Question 1.

Pre-release Material

An organisation has a visitor car park with 20 car parking spaces numbered 1 to 20. Car park spaces can be booked by visitors up to two weeks before the date they are needed, as long as a space is available. Visitors request a car parking space by stating the day in the two-week period in which it is required. They give the licence number of the car to be parked and their name. The next available space, beginning at space 1, is allocated and the given data and booking are stored. A system is required to record the car park bookings.

Write and test a program or programs for the visitor car park booking system to work for a static period of two weeks:

- Your program or programs must include appropriate prompts for the entry of data. Data must be validated on entry.
- All outputs, including error messages, need to be set out clearly and understandably.
- All variables, constants and other identifiers must have meaningful names.

You will need to complete these **three** tasks. Each task must be fully tested.

Task 1 – setting up the booking system
Set up suitable data structures to store the car licence numbers and names of visitors who have booked car parking spaces. The data structures should have sufficient capacity to store data for each of the 20 parking spaces for a static period of two weeks. Allow a visitor to request a parking space on any day within the two-week period by entering a number between 1 and 14, inclusive. The system will check that there are spaces available on the day requested, and if so, will ask the visitor to enter their name and car licence number. This data will be stored in the data structures representing the first available parking space for the day requested. The visitor will be told the number of their parking space.

At the end of the two-week period, allow all of the data to be deleted ready for the next two-week period.

Task 2 – adding accessible parking spaces
The visitor car park booking system is to be re-designed to offer accessible parking. Spaces 1 to 5 are named accessible spaces. Spaces 6 to 20 are named general spaces.

Extend your program in **Task 1** so that:

- when a visitor requests a parking space, they are additionally asked if they need an accessible space
  - if so, they are allocated the first available space beginning at space 1 and finishing at space 20
  - if **not**, they are allocated the first available space beginning at space 20 and finishing at space 6.

The system must work so that visitors requiring accessible parking may be allocated any of the 20 spaces, but visitors who do **not** need accessible parking may only be allocated general spaces.

Task 3 – working out car park usage statistics
Extend the program to enable the following statistics to be counted and output on request:

- The number of accessible spaces used on any of the 14 days.
- The number of general spaces used on any of the 14 days.
- The total number of spaces used on any of the 14 days.
- The number of accessible spaces used in the whole 14-day period.
- The number of general spaces used in the whole 14-day period.
- The total number of spaces used in the whole 14-day period.

1 All variables, constants and other identifiers must have meaningful names.

(a) (i) Identify **one** constant you could have used for **Task 1** and state its value.

Constant .....

Value .....
[2]

(ii) Identify **one** array you could have used in **Task 1** and describe its use.

Array .....

Use .....

.....

.....
[2]

(b) Describe how your program could validate the input for the day number within the two-week period to make sure an appropriate value is entered (part of **Task 1**).

You must include programming statements as part of your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(c) Write an algorithm to show how your program allocates a parking space if the visitor requires accessible parking (part of **Task 2**), including all relevant input and output, using pseudocode, programming statements or a flowchart.

[6]

(d) Explain how your program in **Task 2** could be altered so that bookings can be made for any time over a four-week period instead of the current two-week period. Any programming statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

(e) Explain how your program records and outputs the number of accessible and general parking spaces used for the two-week period (part of **Task 3**). Any programming statements used in your answer must be fully explained.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

Section B

2 Draw a line to connect each programming concept to the most appropriate description.

Programming concept	Description
counting	carrying out an action multiple times within a loop structure
repetition	adding together the numbers in a list of numbers
selection	tracking the number of iterations a program has performed in a loop
sequence	branching off to take a course of action depending on the answer to a question
totalling	a set of statements to be executed in order

[4]

3 Describe the use of verification on input of data when entering a list of items in stock into a database. Explain why verification is necessary.

.....

.....

.....

.....

.....

..... [3]

4 Describe **one** type of test data that must be used to test if a program accepts valid input data.

.....

.....

.....

..... [2]

5 This pseudocode should allow 500 marks to be entered into the algorithm. If the mark is 80 or greater it is stored in an array for higher marks. If the mark is less than 80, but greater than or equal to 50 it is stored in an array for middle marks. The remaining marks are stored in an array for lower marks. The results from the algorithm are displayed at the end.

```
01 HighList ← 0
02 MidList ← 0
03 LowList ← 0
04 MarksEntry ← 0
05 REPEAT
06   INPUT Mark
07   IF Mark >= 80
08     THEN
09       Higher[HighList] ← MarksEntry
10       HighList ← HighList + 1
11     ELSE
12       IF Mark >= 50
13         THEN
14           Middle[MidList] ← Mark
15           MidList ← MidList
16         ELSE
17           Lower[HighList] ← Mark
18           LowList ← LowList + 1
19       ENDIF
20   ENDIF
21   MarksEntry ← MarksEntry + 1
22 NEXT MarksEntry = 500
23 OUTPUT "You entered ", HighList, " higher marks"
24 OUTPUT "You entered ", MidList, " middle marks"
25 OUTPUT "You entered ", LowList, " lower marks"
```

(a) Identify the **four** errors in the pseudocode and suggest a correction for each error.

Error 1 .....

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

Error 4 .....

Correction .....

.....

[4]

(b) The corrected algorithm needs to be changed so that any number of marks may be entered and the algorithm runs until the user tells it to stop.

Write the new pseudocode statements that would be needed to achieve this and state where in the algorithm they would be placed.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

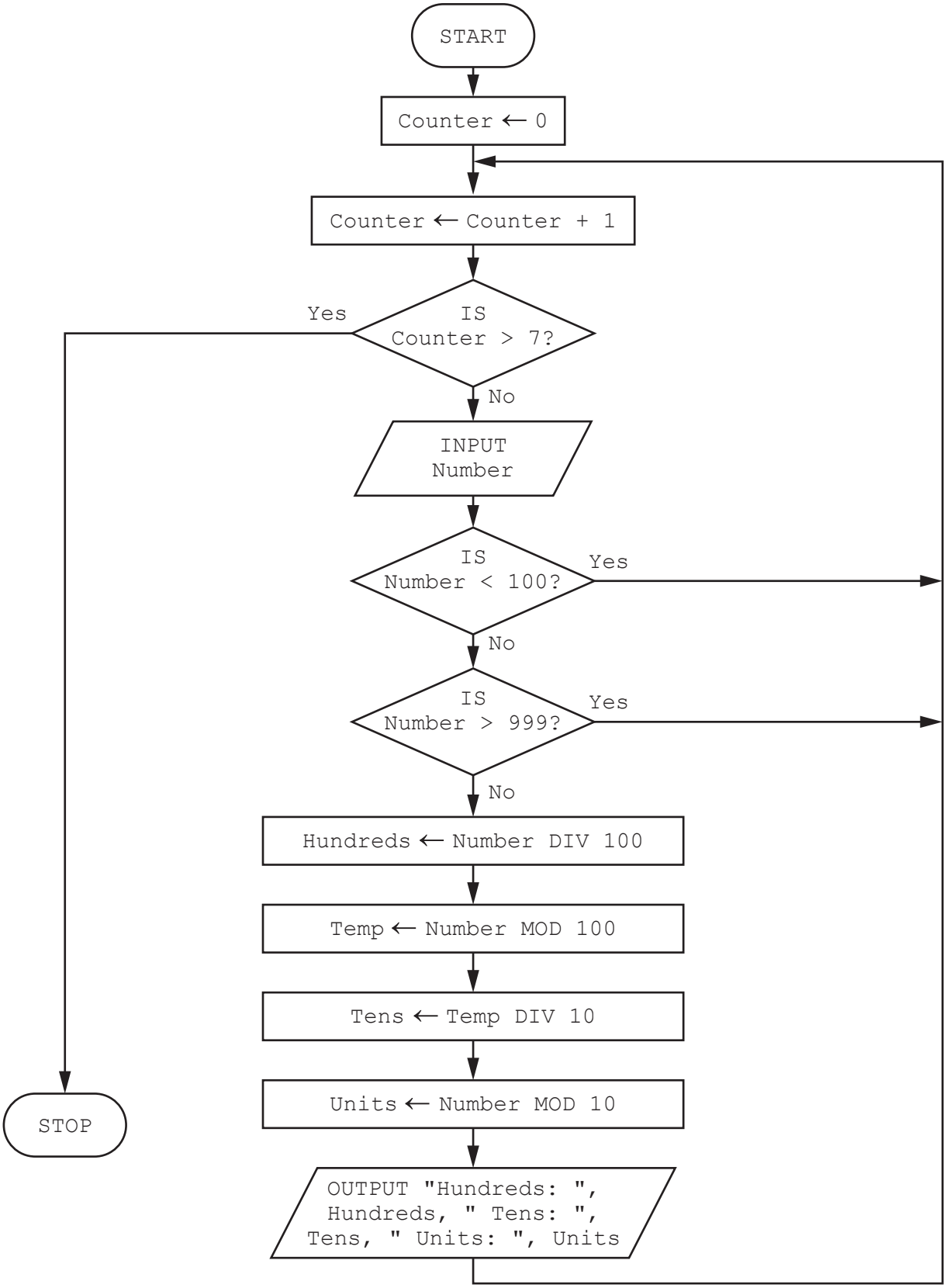
.....

..... [4]

6 This flowchart represents an algorithm to divide three-digit numbers into hundreds, tens and units.

The pre-defined function `DIV` gives the value of the result of integer division, for example `Y = 9 DIV 4` gives the value `Y = 2`

The pre-defined function `MOD` gives the value of the remainder of integer division, for example `R = 9 MOD 4` gives the value `R = 1`



Complete the trace table for the algorithm using this input data:

97, 876, 4320, 606, 9875, 42, 124

Counter	Number	Hundreds	Temp	Tens	Units	OUTPUT

[5]

7 A school uses a database table, ASSESS, to keep a record of the internal assessments and the number of candidates for each of the subjects in its curriculum.

SubjectCode	SubjectName	Exams	Practicals	Candidates
COMP	Computer Science	2	1	200
INFO	Information Technology	1	2	200
MATH	Mathematics	3	0	350
PHYS	Physics	2	1	120
CHEM	Chemistry	2	1	120
BIOL	Biology	2	1	200
GEOG	Geography	2	0	200
HIST	History	2	0	250
GEOL	Geology	2	0	80
PHED	Physical Education	1	2	350
FREN	French	2	2	120
ENGL	English	2	2	350

This database only allows the data types:

- text
- number
- currency
- Boolean.

(a) (i) State the most appropriate data type for the fields SubjectCode and Exams.

SubjectCode .....

Exams .....

[1]

(ii) State **one** reason why the Candidates field could **not** be of the Boolean data type.

.....

..... [1]

(b) Show the output given by the query-by-example grid.

Field:	SubjectName	Practicals	Candidates		
Table:	ASSESS	ASSESS	ASSESS		
Sort:	Ascending				
Show:	☑	☐	☑	☐	☐
Criteria:		<1			
or:					

.....

.....

.....

.....

.....

..... [3]

(c) Complete the query-by-example grid to output the subjects with fewer than 150 candidates. Display only the SubjectCode, SubjectName and Candidates fields in order of the number of candidates from largest to smallest.

Field:					
Table:					
Sort:					
Show:	☐	☐	☐	☐	☐
Criteria:					
or:					

[3]

BLANK PAGE

BLANK PAGE

BLANK PAGE