Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME			
CENTRE NUMBER		CANDIDATE NUMBER	

8637779197

COMPUTING **9691/21**

Paper 2 **May/June 2016**

2 hours

Candidates answer on the Question Paper.

No additional materials are required.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name on all the work you hand in.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

This document consists of **14** printed pages and **2** blank pages.

DC (NF/SG) 114981/3

CAMBRIDGE
International Examinations

[Turn over

1 Zara wants to write a program to print a set of name labels for a group of students.

Each label will show the name surrounded by a border. There is one gap line above and below the name line. This gap line only has the border symbol at the start and end. There are two spaces between the border symbol and the name.

Here is a label example:

```
@@@@@@@@@@
@        @
@  FRED  @
@        @
@@@@@@@@@@
```

The user will choose the border symbol for the set of labels.

Zara writes pseudocode for her program:

```
INPUT Symbol
REPEAT
   INPUT Name
   LabelWidth ← LENGTH(Name) + 6
   CALL PrintTop2Lines(Symbol, LabelWidth)
   CALL PrintNameLine(Symbol, Name)
   CALL PrintBottom2Lines(Symbol, LabelWidth)
UNTIL Name = "NoName"
```

(a) Write **program code** for the procedure

```
PrintNameLine(Symbol, Name)
```

Programming language ....

Code ....

....

....

....

....

....

....

.... [4]

Zara uses top-down design and refines her solution:

```
PROCEDURE PrintTop2Lines(Symbol : CHAR, LabelWidth : INTEGER)
   CALL PrintGapLine(Symbol, LabelWidth)
   CALL PrintSymbolLine(Symbol, LabelWidth)
ENDPROCEDURE

PROCEDURE PrintBottom2Lines(Symbol : CHAR, LabelWidth : INTEGER)
   CALL PrintSymbolLine(Symbol, LabelWidth)
   CALL PrintGapLine(Symbol, LabelWidth)
ENDPROCEDURE
```

(b) Write **program code** for the procedures below.

Programming language ....

```
PrintSymbolLine(Symbol : CHAR, LabelWidth : INTEGER)
```

....

....

....

....

....

....

....

....

```
PrintGapLine(Symbol : CHAR, LabelWidth : INTEGER)
```

....

....

....

....

....

....

....

.... [6]

(c) Zara now decides that the name labels should all be of the same width, regardless of the length of the name.

She amends her top-level design:

```
CONSTANT LabelWidth = 20
INPUT Symbol
REPEAT
   INPUT Name
   CALL PrintTop2Lines(Symbol, LabelWidth)
   CALL PrintNameLine(Symbol, Name, LabelWidth)
   CALL PrintBottom2Lines(Symbol, LabelWidth)
UNTIL Name = "NoName"
```

The name is to be centred within the label.

(i) Rewrite the procedure `PrintNameLine`.

Your solution should allow for the case when the name has an odd number of characters.

Programming language ....

Code ....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [7]

(ii) Describe what happens when a name consists of 18 or more characters.

....

....

....

.... [2]

(iii) Suggest a solution to the problem that you described in **part (c)(ii)**.

....

....

....

.... [2]

(d) Zara used top-down design to develop her modular solution.

Give **two** benefits of a modular solution.

1 ....

....

2 ....

.... [2]

(e) The pseudocode Zara wrote in **part (c)** has some features that make it easier to read and understand.

State **three** such features.

1 ....

....

2 ....

....

3 ....

.... [3]

2 A city has a metro line with five stations. The transport department wants to introduce a ticket price calculator for its website. The transport department gives the following specification to a programmer.

Tickets cost 25 cents for each kilometre (km) travelled.

The user chooses the origin and destination of their journey from a list:

Station number	**Station name**
1	Airport
2	Industrial Centre
3	City Centre
4	University
5	Sports Stadium

Journeys can start and end at any of the following stations. Travel can be in either direction.

The distances between adjacent stations are:

Between stations	**Distance (km)**
1 and 2	4
2 and 3	2
3 and 4	1
4 and 5	3

To use the ticket price calculator, a user clicks on the 'Ticket Price Calculator' button. This calls the procedure `TicketCalculator`. The user inputs the numbers of their chosen start and end stations. The procedure calculates and displays the ticket price.

(a) Write **program code** to declare and initialise a one-dimensional array `Interval` to store the four distances between the stations.

Programming Language ....

Code ....

....

....

....

....

....

.... [2]

(b) Carefully choose **four** sets of test data to test the procedure `TicketCalculator`. Explain the reason for your choice of test data. Your reason for each test should be different.

Only choose journeys where:

- station numbers are valid
- origin and destination station numbers are different

One set of test data has been done for you.

Test number	**Station number**		**Reason for choice**
	Origin	**Destination**	
1	1	5	Tests longest journey, starting at first station, finishing at last station
2			
3			
4			
5			

[4]

(c) Use the flowchart opposite to write **program code** for the procedure `TicketCalculator`.

Programming language ....

Code ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

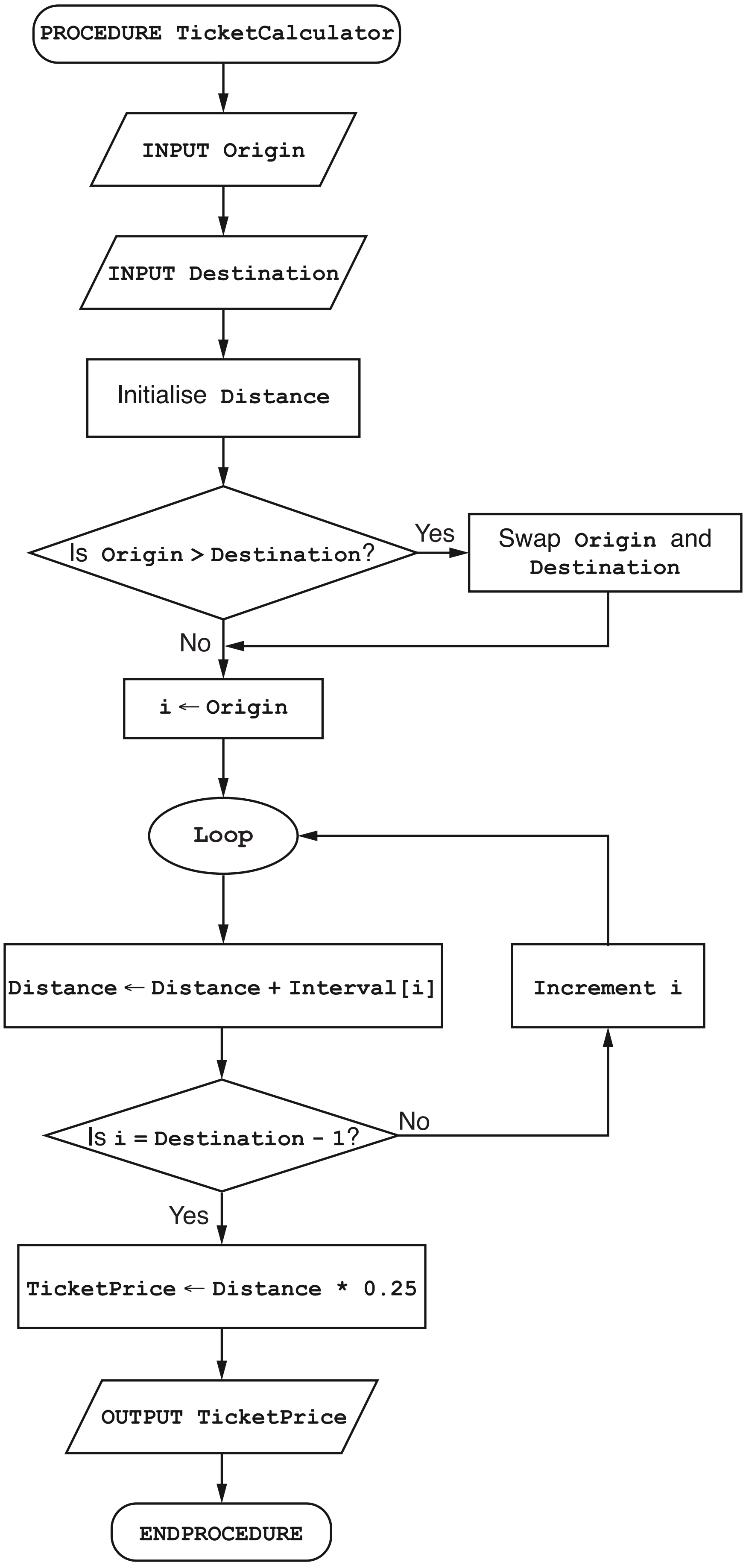
....

....

....

....

.... [10]



PROCEDURE TicketCalculator
INPUT Origin
INPUT Destination
Initialise Distance
Is Origin > Destination?
Yes
Swap Origin and Destination
No
i ← Origin
Loop
Distance ← Distance + Interval[i]
Increment i
Is i = Destination - 1?
No
Yes
TicketPrice ← Distance * 0.25
OUTPUT TicketPrice
ENDPROCEDURE

3 Alessio has been asked to write a computerised table booking system for a restaurant. The restaurant has 12 tables, each seating between 2 and 8 people.

The restaurant manager wants the following facilities:

- take a new booking
- cancel a booking
- report on which table(s) are available

As a first attempt, Alessio designs his program to take bookings for one evening only. Each table is only booked once per evening.

He writes the pseudocode for his main program:

```
CALL Initialisation
EndProgram ← FALSE
REPEAT
   CALL DisplayOptions
   INPUT Option
   CASE OF Option
      1: CALL TakeBooking
      2: CALL CancelBooking
      3: CALL AvailableTablesReport
      4: EndProgram ← TRUE
      OTHERWISE OUTPUT "Error - invalid input"
   ENDCASE
UNTIL EndProgram = TRUE
```

Each booking is for a number of customers (size of the group). Each table has a number of seats. The procedure `Initialisation` will set up arrays `TableSeats`, `Booked` and `GroupSize` as follows:

TableNumber	**TableSeats**		**Booked**		**GroupSize**
1	2	1	FALSE	1	0
2	2	2	FALSE	2	0
3	2	3	FALSE	3	0
4	2	4	FALSE	4	0
5	4	5	FALSE	5	0
6	4	6	FALSE	6	0
7	4	7	FALSE	7	0
8	4	8	FALSE	8	0
9	6	9	FALSE	9	0
10	6	10	FALSE	10	0
11	8	11	FALSE	11	0
12	8	12	FALSE	12	0

(a) When a booking enquiry is made, the number of customers is given.

Complete the pseudocode for the procedure `TakeBooking`:

```
PROCEDURE TakeBooking
   DECLARE ..................................., ................................... : INTEGER
   DECLARE ........................................................................ : BOOLEAN
   INPUT NumberOfCustomers
   // initialise search for a suitable table
   Found ← FALSE
   TableNumber = ...........................................................................
   REPEAT      // find a table with enough seats
      TableNumber ← TableNumber + 1
      IF TableSeats[.........................] ............................................
            AND Booked[.........................] ......................................
         THEN
            Found ← TRUE
      ENDIF
   UNTIL ...................................... OR ........................................
   IF Found = FALSE
      THEN   // no tables left with enough seats
         ...............................................................................
      ELSE   // make the booking
         Booked[.........................] .............................................
         GroupSize[.........................] ..........................................
         OUTPUT "Table number booked: ", TableNumber
   ENDIF
ENDPROCEDURE
```

[11]

(b) To cancel a booking, the table number is quoted.

Complete the pseudocode for the procedure `CancelBooking`.

```
PROCEDURE CancelBooking

   ..........................................................................

   INPUT TableNumber

   IF ......................................................................

      THEN

         OUTPUT "Error - this table is not booked"

      ELSE                               // cancel booking

         Booked[..........................] ...................................


         GroupSize[..........................] ...............................

         OUTPUT "Booking cancelled"

   ENDIF

ENDPROCEDURE
```

[4]

(c) The restaurant manager wants a report to show all available tables, each with its number of seats.

Write **pseudocode** for the procedure `AvailableTablesReport`.

....

....

....

....

....

....

....

....

....

.... [4]

(d) The manager wants to store the name and telephone number of the customer making the booking. At the time the booking is made, the manager also wants to collect a deposit ($5 per person). The program will record the deposit paid.

Alessio decides that a record structure would be more suitable for his program.

(i) Write **program code** to define a booking record with identifier `BookingType` and the fields `TableSeats`, `Booked`, `GroupSize`, `CustomerName`, `CustomerTelNumber`, `AmountDepositPaid`.

Programming language ....

Code ....

....

....

....

....

....

....

....

....

.... [5]

(ii) Each restaurant table will have its data stored in its own booking record. Alessio decides to use an array of records.

Write **program code** to declare the array `TableBookings` for the 12 table records. Use the record structure you defined in **part (d)(i)**.

Programming language ....

Code ....

....

.... [3]

(e) At the end of the day, the computer is shut down. Before the program stops, the array data from `TableBookings` must be saved to a new sequential file `TableBookings.DAT`

Write **program code** for the procedure `SaveToFile`.

Programming language ....

Code ....

....

....

....

....

....

....

....

....

....

....

.... [6]

BLANK PAGE

BLANK PAGE