Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

9666076342

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2015**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.
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.
No marks will be awarded for using brand names of software packages or hardware.

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.

The maximum number of marks is 75.

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

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Horses are entered for a horse race. A horse may have to carry a penalty weight in addition to the rider. This weight is added to the saddle. The penalty weight (if any) depends on the number of wins the horse has achieved in previous races.

The penalty weight is calculated as follows:

Number of previous wins	Penalty weight (kg)
0	0
1 or 2	4
Over 2	8

A program is to be written from the following structured English design.

```
1   INPUT name of horse
2   INPUT number of previous wins
3   CALCULATE penalty weight
4   STORE penalty weight
5   OUTPUT name of horse, penalty weight
```

(a) Complete the identifier table showing the variables needed to code the program.

Identifier	Data type	Description

[3]

(b) Line 3 in the algorithm above does not give the detail about how the race penalty weight is calculated; this step in the algorithm must be expressed in more detail.

(i) The algorithm above currently has five stages. One technique for program design is to further break down, where required, any stage to a level of detail from which the program code can be written.

Name this technique.

.....[1]

(ii) Write **pseudocode** for the given structured English design.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

2 **(a)** Two operators available in a programming language are `DIV` and `MOD`. They perform integer arithmetic as follows:

Expression	Explanation
`X DIV Y`	Computes the number of times `Y` divides into `X`
`X MOD Y`	Computes the remainder when `X` is divided by `Y`

Calculate the value of the variables shown for the following code fragments.

	Code	Variable
(i)	`NumberLeftOver ← 37 MOD 10`	`NumberLeftOver` ..... [1]

(ii)	`Quantity ← 208`
	`BoxSize ← 100`
	`NumberOfBoxes ← Quantity DIV BoxSize`
	`Temp ← (Quantity MOD BoxSize) + 1`
		`NumberOfBoxes` .....
		`Temp` ..... [2]

(b) Bank customers withdraw money from their account at a cash dispenser machine using their bank card. The machine operates as follows:

- it can dispense the following notes:
  - $50
  - $20
  - $10
- the maximum amount for a single withdrawal is $500

When a customer withdraws money, they enter the amount to withdraw. (This must be a multiple of $10).

The machine will always dispense the least possible number of notes.

A program is designed for the machine to process a withdrawal.

The following variables are used:

Identifier	Data type	Description
`Amount`	`INTEGER`	Amount to withdraw entered by the user
`FiftyDollar`	`INTEGER`	Number of $50 notes to dispense
`TwentyDollar`	`INTEGER`	Number of $20 notes to dispense
`TenDollar`	`INTEGER`	Number of $10 notes to dispense
`Temp`	`INTEGER`	Used in the calculation of the number of each note required

(i) The following four tests have been designed.

Complete the test data table showing the expected results with comments.

Input value	Output			Comment
Amount	**FiftyDollar**	**TwentyDollar**	**TenDollar**	
70	1	1	0	Least possible number of notes
85				
130				
600				

[3]

(ii) Complete the pseudocode.

```
INPUT .....
IF Amount > 500
   THEN
      OUTPUT "Refused – amount too large"
   ELSE
      .....
         THEN
            OUTPUT "Refused - not a multiple of $10"
         ELSE
            FiftyDollar ← Amount DIV 50
            Temp ← .....
            TwentyDollar ← .....
            Temp ← .....
            .....
      ENDIF
ENDIF
```

[5]

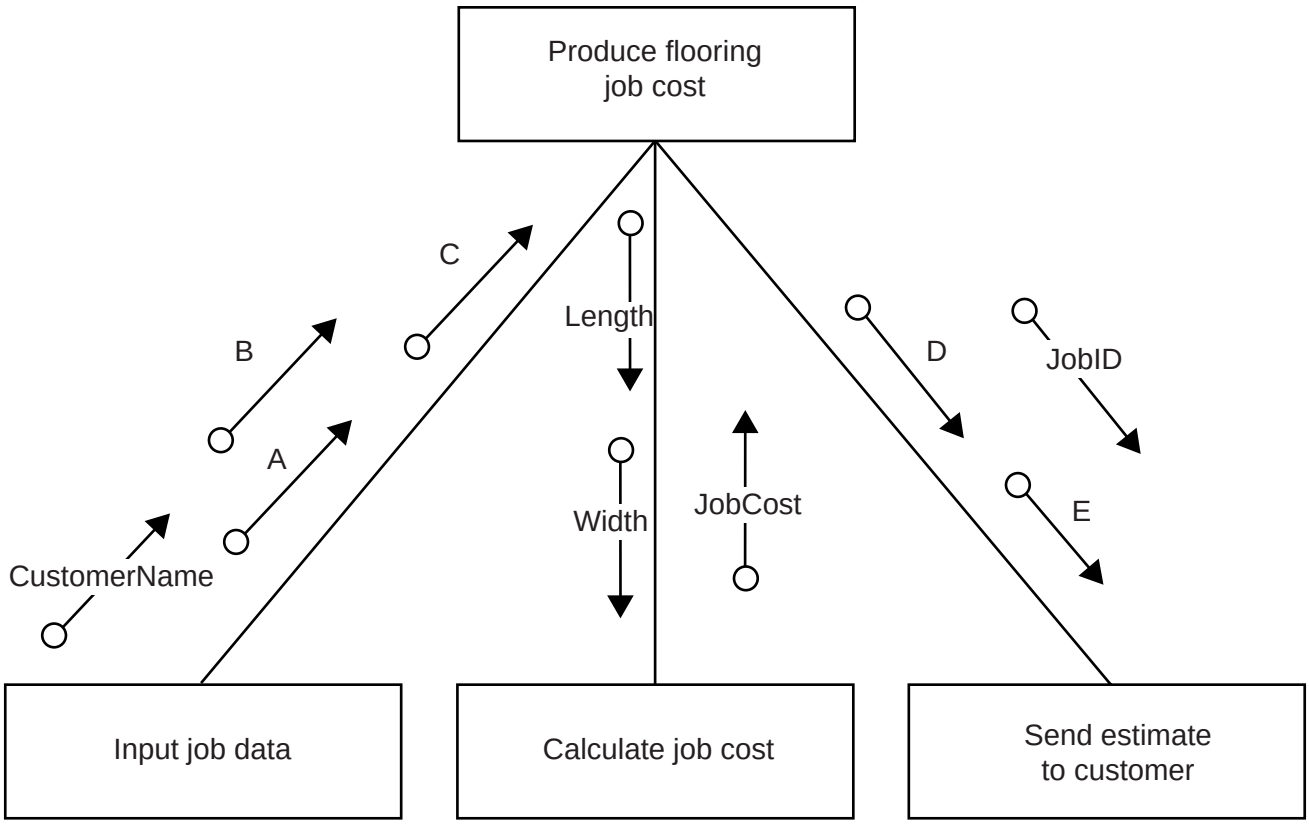
3 A flooring company provides for each customer an estimated price for a new job. Each job is given a Job ID.

The job cost is calculated from the length (nearest metre) and width (nearest metre) of the room.

The process for calculating the price is as follows:

- the floor area is calculated with 18% added to allow for wastage
- the job cost is calculated at $50 per square metre

The structure chart shows the modular design for a program to produce a new job cost.



(i) Give the data items corresponding to the labels A to E in the structure chart.

A .....

B .....

C .....

D .....

E .....

[5]

(ii) The procedure below is one of the modules shown on the structure chart.

Parameters can be passed ‘by value’ or ‘by reference’.

Complete the procedure header below showing for each parameter:

- its parameter passing mechanism
- its identifier
- its data type

```
PROCEDURE CalculateJobCost( ..........................................................................
                            ..........................................................................
                            ..........................................................................
                            .......................................................................... )
  JobCost ← (Length * Width * 1.18) * 50
ENDPROCEDURE
```

[5]

4 A programming language has the built-in function CONCAT defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )  RETURNS STRING
For example:  CONCAT("San", "Francisco") returns "SanFrancisco"
              CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.

(a) State the value returned by the following expressions.

If the expression is not properly formed, write ERROR.

(i) `CONCAT("Studio", 54)` ..... [1]

(ii) `CONCAT("parity", "error", "check")` ..... [1]

(iii) `CONCAT(CONCAT("Binary", "▼", "Coded"), "▼", "Decimal")`

▼ indicates a <Space> character

.....[2]

(b) A country has a number of banks. There are cash dispensers all over the country. Each bank is responsible for a number of dispensers.

- banks have a three digit code in the range 001 – 999
- each dispenser has a five digit code in the range 00001 – 99999

A text file, DISPENSERS, is to be created.

It has one line of text for each dispenser. For example: `00342▼007`.

This line in the file is the data for dispenser `00342` which belongs to bank `007`.

Incomplete pseudocode follows for the creation of the file DISPENSERS.

For the creation of the file, data is entered by the user at the keyboard.

(i) Complete the **pseudocode**.

```
OPENFILE ..... FOR WRITE

.....

   OUTPUT "Enter dispenser code (XXXXX to end)"
   INPUT DispenserCode
   IF DispenserCode <> "XXXXX"
      THEN
         OUTPUT "Enter bank code"
         INPUT BankCode
         LineString ← CONCAT(.....,"▼", BankCode)
         // now write the new line to the file

         .....

   ENDIF

UNTIL .....

.....

OUTPUT "DISPENSERS file now created"
```

[6]

(ii) No attempt has been made to validate the data entered by the user.

Describe **two** different types of validation check for the data entry.

1 .....

.....

2 .....

.....[2]

(iii) The programmer coded this algorithm above and the user successfully entered 15 dispenser records into the text file.

There is data for another 546 dispensers which needs to be added.

State the error that will occur if the user runs the program a second time for further data entry.

.....[1]

(iv) Give the 'file mode' available in the programming language which will be used to address this issue.

.....[1]

(c) The complete data file is created with the structure shown.

A new program is to be written to search the file.

The program will:

- input a bank code
- output a list of all the dispensers which belong to this bank
- output the total number of dispensers for this bank

```
00001▼007
00002▼001
00003▼002
00004▼003
00005▼101
00006▼004
00007▼004
```

```
00024▼002
00025▼003
00026▼007
00027▼007
00028▼102
```

```
99867▼013
```

An example of a run of the program is shown:

```
Enter bank code 007
00001
00011
00022
00026
00027

There are 5 dispensers for this bank
```

Write the **program code**. Do not attempt to include any validation checks.

Visual Basic and Pascal: You should include the declaration statements for variables.

Python: You should show a comment statement for each variable used with its data type.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[10]

5 A firm employs workers who assemble amplifiers. Each member of staff works an agreed number of hours each day.

The firm records the number of completed amplifiers made by each employee each day.

Management monitor the performance of all its workers.

Production data was collected for 3 workers over 4 days.

	Daily hours worked
Worker 1	5
Worker 2	10
Worker 3	10

Production data

	Worker 1	**Worker 2**	**Worker 3**
Day 1	10	20	9
Day 2	11	16	11
Day 3	10	24	13
Day 4	14	20	17

A program is to be written to process the production data.

(a) The production data is to be stored in a 2-dimensional array `ProductionData`, declared as follows:

```
DECLARE ProductionData ARRAY[1:4, 1:3] : INTEGER
```

(i) Describe **two** features of an array.

1 .....

.....

2 .....

.....[2]

(ii) Give the value of `ProductionData[3, 2]`.

.....[1]

(iii) Describe the information produced by the expression:

```
ProductionData[2, 1] + ProductionData[2, 2] + ProductionData[2, 3]
```

.....

.....[2]

(b) Complete the trace table for the pseudocode algorithm below.

```
FOR WorkerNum ← 1 TO 3
   WorkerTotal[WorkerNum] ← 0
ENDFOR

FOR WorkerNum ← 1 TO 3
   FOR DayNum ← 1 TO 4
      WorkerTotal[WorkerNum] ← WorkerTotal[WorkerNum] +
                                   ProductionData[DayNum, WorkerNum]
   ENDFOR
ENDFOR

FOR WorkerNum ← 1 TO 3
   WorkerAverage ← WorkerTotal[WorkerNum]/
                                    (4 * DailyHoursWorked[WorkerNum])
   IF WorkerAverage < 2
      THEN
         OUTPUT “Investigate“, WorkerNum
   ENDIF
ENDFOR
```

WorkerNum	DayNum	WorkerAverage	OUTPUT

WorkerTotal		
1	2	3

[8]

(c) An experienced programmer suggests that the pseudocode would be best implemented as a procedure `AnalyseProductionData`.

Assume that both arrays, `DailyHoursWorked` and `ProductionData`, are available to the procedure from the main program and they are of the appropriate size.

```
PROCEDURE AnalyseProductionData(NumDays : INTEGER, NumWorkers : INTEGER)

   DECLARE .....

   DECLARE .....

   DECLARE .....

   DECLARE .....

   FOR WorkerNum ← 1 TO 3
      WorkerTotal[WorkerNum] ← 0
   ENDFOR

   FOR WorkerNum ← 1 TO 3
      FOR DayNum ← 1 TO 4
         WorkerTotal[WorkerNum] ← WorkerTotal[WorkerNum] +
                                        ProductionData[DayNum, WorkerNum]
      ENDFOR
   ENDFOR

   FOR WorkerNum ← 1 TO 3
      WorkerAverage ← WorkerTotal[WorkerNum]/
                                       (4 * DailyHoursWorked [WorkerNum])
      IF WorkerAverage < 2
         THEN
            OUTPUT "Investigate", WorkerNum
      ENDIF
   ENDFOR

ENDPROCEDURE
```

(i) Complete the declaration statements showing the local variables. [4]

(ii) The original pseudocode has been 'pasted' under the procedure header.

Circle all the places in the original pseudocode where changes will need to be made.

Write the changes which need to be made next to each circle. [3]

(iii) Write the statement for a procedure call which processes data for 7 days for 13 workers.

.....[1]

BLANK PAGE

BLANK PAGE