

## **Cambridge International Examinations**

Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME			
CENTRE NUMBER		CANDIDATE NUMBER	
COMPUTERS	SCIENCE		9608/41
Paper 4 Furth	er Problem-solving and Programming Skills	Oct	ober/November 2017
			2 hours
Candidates an	swer on the Question Paper.		
No Additional I	Materials are required.		

## **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 calculators allowed.

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.



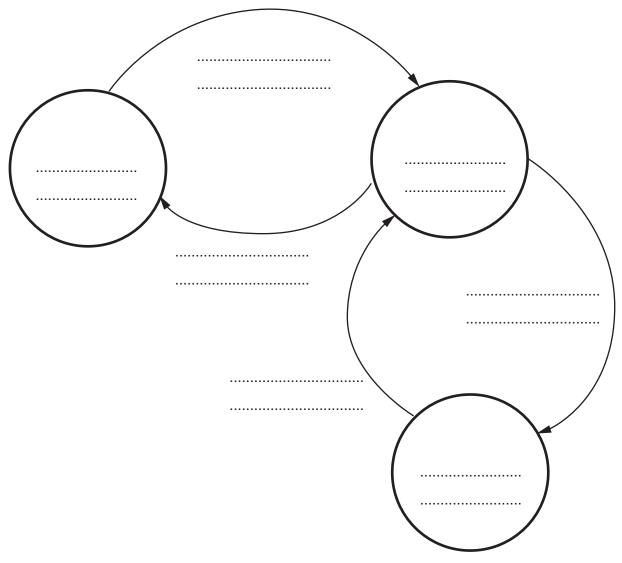
**1** A greenhouse has a window that automatically opens and closes depending on the internal temperature.

If the temperature rises above 20 °C, the window half opens. If the temperature rises above 30 °C, the window fully opens. If the temperature drops below 25 °C, the window returns to being half open. If the temperature drops below 15 °C, the window fully closes.

The window has three possible states: Closed, Half Open and Fully Open.

Current state	Event	Next state		
Closed	Temperature rises above 20 °C	Half Open		
Half Open	Temperature drops below 15°C	Closed		
Half Open	Temperature rises above 30 °C	Fully Open		
Fully Open	Temperature drops below 25 °C	Half Open		

Complete the state-transition diagram for the window:



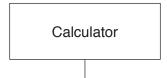
2	(a)	(i)	State how repetition is shown in a Jackson Structured Programming (JSP) structure diagram.
			[1]
		(ii)	State how selection is shown in a JSP structure diagram.
			[1]

**(b)** A simple calculator is to be created.

The calculator is to be used as follows:

- User inputs 2 numbers (x and y).
- User inputs an operator (+, -, \* or /).
- The calculator computes the answer.
- The calculator displays the answer.

Draw a JSP diagram for the calculator. The first element is provided.



3 A declarative programming language is used to represent the following knowledge base:

```
01 person(jane).
02 person(ahmed).
03 person(caroline).
04 person(stuart).
05 food(chocolate).
06 food(sushi).
07 food(pizza).
08 food(chilli).
09 likes(jane, pizza).
10 likes (ahmed, chocolate).
11 likes(ahmed, pizza).
12 likes(jane, chilli).
13 likes(stuart, sushi).
14 dislikes(stuart, chocolate).
15 dislikes(jane, sushi).
16 dislikes(caroline, pizza).
```

These clauses have the following meanings:

Clause	Explanation				
01	Jane is a person				
0.5	Chocolate is a food				
09	Jane likes pizza				
14	Stuart dislikes (does not like) chocolate				

(a) Mimi is a person who likes chocolate but does not like sushi or lettuce.

Write additional clauses to represent this information.

17	
18	
19	
20	
21	
	[5]

(b)	Using the variable PersonName, the goal:
	likes(PersonName, pizza).
	returns:
	PersonName = jane, ahmed.
	Write the result that is returned by the goal:
	likes(ahmed, FoodItem).
	FoodItem =
	[2]
(c)	B might like A, if B is a person, A is a food and B does not dislike A.
	Write this as a rule.
	might_like()
	IF
	[6]

4 The following table shows part of the instruction set for a processor. The processor has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Fundanskian			
Op code	Operand	Explanation			
LDM	#n	Immediate addressing. Load the number n to ACC.			
LDD	<address></address>	Direct addressing. Load the contents of the location at the given address to ACC.			
LDI	<address></address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.			
LDX	<address></address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.</address>			
LDR	#n	Immediate addressing. Load the number n to IX.			
STO	<address></address>	Store the contents of ACC at the given address.			
STX	<address></address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.</address>			
ADD	<address></address>	Add the contents of the given address to the ACC.			
INC	<register></register>	Add 1 to the contents of the register (ACC or IX).			
DEC	<register></register>	Subtract 1 from the contents of the register (ACC or IX).			
JMP	<address></address>	Jump to the given address.			
CMP	<address></address>	Compare the contents of ACC with the contents of <address>.</address>			
CMP	#n	Compare the contents of ACC with number n.			
JPE	<address></address>	Following a compare instruction, jump to <address> if the compare was True.</address>			
JPN	<address></address>	Following a compare instruction, jump to <address> if the compare was False.</address>			
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.			
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.			
IN		Key in a character and store its ASCII value in ACC.			
OUT		Output to the screen the character whose ASCII value is stored in ACC.			
END		Return control to the operating system.			

(a) A program stores a letter. The user is allowed nine attempts to guess the stored letter. The program outputs "?" and the user guesses a letter. If the user guesses the letter, the program outputs "\*".

The following is pseudocode for this program.

```
REPEAT

OUTPUT '?'

INPUT GUESS

IF GUESS = LETTERTOGUESS

THEN

OUTPUT '*'

BREAK

ELSE

ATTEMPTS 	— ATTEMPTS + 1

ENDIF

UNTIL ATTEMPTS = 9
```

Write this program. Use the op codes from the instruction set provided.

Label	Op code	Operand	Comment
START:	LDM	#63	// load ASCII value for '?'
			// OUTPUT '?'
			// input GUESS
			// compare with stored letter
			// if correct guess, go to GUESSED
			// increment ATTEMPTS
			// is ATTEMPTS = 9 ?
			// if out of guesses, go to ENDP
			// go back to beginning of loop
GUESSED:	LDM	#42	// load ASCII for '*'
			// OUTPUT '*'
ENDP:	END		// end program
ATTEMPTS:		0	
LETTERTOGUESS:		'a'	

[11]

**(b)** Five numbers are stored, starting in the location labelled NUMBERS. A program is needed to multiply each of the numbers by 4 and store them back in their original location.

Write this program. Use the op codes from the instruction set on the opposite page.

Label	Op code	Operand	Comment					
START:			// initialise the Index Register					
			// load the value from NUMBERS					
			// multiply by 4					
			// store the new value in NUMBERS					
			// increment the Index Register					
			// increment COUNT					
			// is COUNT = 5 ?					
			// repeat for next number					
ENDP:	END							
COUNT:	(	)						
NUMBERS:	22							
	13							
	ī	5						
	4	6						
	1	2						

[10]

Instruction		Evalenation		
Op code	Operand	- Explanation		
LDM	#n	Immediate addressing. Load the number n to ACC.		
LDD	<address></address>	Direct addressing. Load the contents of the location at the given address to ACC.		
LDI	<address></address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.		
LDX	<address></address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.</address>		
LDR	#n	Immediate addressing. Load the number n to IX.		
STO	<address></address>	Store the contents of ACC at the given address.		
STX	<address></address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.</address>		
ADD	<address></address>	Add the contents of the given address to the ACC.		
INC	<register></register>	Add 1 to the contents of the register (ACC or IX).		
DEC	<register></register>	Subtract 1 from the contents of the register (ACC or IX).		
JMP	<address></address>	Jump to the given address.		
CMP	<address></address>	Compare the contents of ACC with the contents of <address>.</address>		
CMP	#n	Compare the contents of ACC with number n.		
JPE	<address></address>	Following a compare instruction, jump to <address> if the compare was True.</address>		
JPN	<address></address>	Following a compare instruction, jump to <address> if the compare was False.</address>		
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.		
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.		
IN		Key in a character and store its ASCII value in ACC.		
OUT		Output to the screen the character whose ASCII value is stored in ACC.		
END		Return control to the operating system.		

Lar	ge de	evelopment projects require careful resource management.
(a)	(i)	Name an appropriate project management tool that helps the manager to work out the estimated length of time it takes for the project to complete.
		[1]
	(ii)	Explain how, during the planning stage of the project, the manager would use the tool you named in <b>part (a)(i)</b> .
		[3]
(b)	(i)	Different programmers have been writing independent modules. The modules now need to be combined to create the final system.
		Name the type of testing required at this stage.
		[1]
	(ii)	Name the final testing stage required before the system becomes operational.
		[1]

A programmer wants to create a computer simulation of animals searching for food in a desert. The desert is represented by a 40 by 40 grid. Each position in the grid is represented by a pair of coordinates. 'A' represents an animal and 'F' represents food. At the start of the simulation, the grid contains 5 animals and 1 food source.

The following is an example of part of the grid.

	0	1	2	3	4	 37	38	39
0	Α							
1			F					
2							Α	
3				Α				
38				Α		 Α		
39								

A timer is used. In each time interval, each animal randomly moves 0 or 1 position in a random direction. The program generates this movement by computing two random numbers, each of which can be -1, 0 or 1. The program adds the first random number to the across number and the second random number to the down number representing the animal's position.

## For example:

- if 0 and 1 are generated, the across value does not change, the down value increases by 1
- if -1 and 1 are generated, the across value decreases by 1, and the down value increases by 1.

Each animal has an individual score. If the animal moves to a position in the grid with food ('F'):

- the animal's score increases by 1
- the food disappears
- one new animal ('A') is randomly generated and added to the grid (to a maximum of 20 animals)
- one new food ('F') is randomly generated and added to the grid.

The simulation is to be implemented using object-oriented programming.

The programmer has designed two classes, Desert and Animal.

The Desert class consists of:

- attributes
  - o Grid
  - o StepCounter
  - O AnimalList
  - O NumberOfAnimals
- methods
  - o Constructor
  - o IncrementStepCounter
  - O GenerateFood
  - O DisplayGrid

The following table describes the attributes and methods for the  ${\tt Animal}$  class.

Identifier	Data type	Description
Constructor()		Instantiate an object of the Animal class  • Generate a pair of random numbers between 0 and 39.  • Place animal at that random position.  • Initialise the animal's score to 0.
EatFood()		<ul> <li>Delete the food.</li> <li>Increase the score of the animal that called the method.</li> <li>Call the GenerateFood method of the Desert class.</li> <li>Call the Constructor method of the Animal class.</li> </ul>
Move()		<ul> <li>Call the         GenerateChangeInCoordinate         method for each coordinate (across         or down number) of the animal's         position.</li> <li>Moves the animal to the new space.</li> <li>If there is food in the new position,         call the EatFood method.</li> </ul>
Score	INTEGER	Initialised to 0
Across	INTEGER	The across value, between 0 and 39
Down	INTEGER	The down value, between 0 and 39

(a)	) Write <b>program code</b> to declare the attributes and constructor for the Animal class.					
	You only need to write the set and get methods for the attribute Across.					
	Υοι	You should also write:				
	•	the constructor for the class set and get methods for the Across attribute only.				
	Programming language					
	Pro	Program code				

.....[6]

- (b) The Constructor method of the Desert class:
  - initialises an empty grid
  - creates 5 animal objects which are added to the AnimalList (an array of animal objects currently on the grid)
  - generates one food
  - sets the StepCounter to 0.

Write program code for the Constructor method.
Programming language
Program code
[E]

- (c) (i) The function GenerateChangeInCoordinate:
  - receives a coordinate (across or down number) as a parameter
  - checks whether the coordinate's value is at a boundary of the grid

Write program code for the GenerateChangeInCoordinate function.

• returns a random change (-1, 0 or 1) that will keep the animal's position within the grid.

Programming language
Program code
[4]

(ii) The Move method uses the GenerateChangeInCoordinate function to calculate the new Across and Down values for an animal. If there is food in the new position in the

	grid, the animal eats the food.
	Write program code for the Move method.
	Programming language
	Program code
	[4]
(d)	The programmer plans to add a graphic display to the program. The programmer will make use of a program library.
	Explain what is meant by a program library.
	[2]
	o reproduce items where third party owned meterial protected by convight is included has been cought and cleared where possible. Figure

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

Cambridge International Examinations is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of University of Cambridge Local Examinations Syndicate (UCLES), which is itself a department of the University of Cambridge.