CAMBRIDGE
International Education

Cambridge IGCSE™ (9–1)

COMPUTER SCIENCE **0984/02**

Paper 2 Algorithms and Programming **For examination from 2029**

MARK SCHEME

Maximum Mark: 75

Specimen

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

General Marking Principles

All examiners must apply these marking principles when marking candidate responses.

1	Examiners must award marks for each response in line with: • the specific **content** defined in the mark scheme • the specific **skills** defined in the mark scheme or in the level descriptions • the **standard of response** required as exemplified by the standardisation scripts.
2	Examiners must award **whole marks** (not half marks, or other fractions).
3	Examiners must award marks **positively**. This means that: • marks are not deducted for errors or omissions • marks are awarded for correct/valid answers as defined in the mark scheme and also for **other valid answers** which go beyond the scope of the syllabus and mark scheme referring to your team leader as appropriate • the quality of spelling, punctuation and grammar are judged **only** when the mark scheme indicates that these features are **specifically assessed** in the question. However, the meaning of all responses must be unambiguous.
4	Examiners must **consistently** apply the requirements of the mark scheme and any rules for what to do when candidates have not followed instructions correctly.
5	Examiners must use the **full range of marks** defined in the mark scheme, unless limited by the quality of the candidate responses seen.
6	Examiners must award marks according to the mark scheme and **must not** award marks with grade thresholds or grade descriptions in mind.

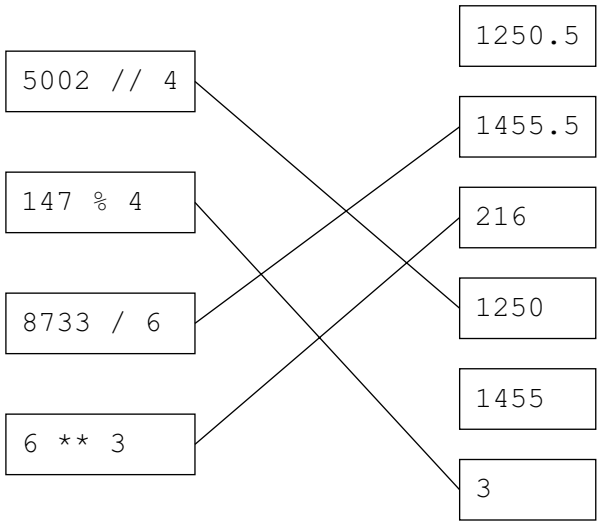
Mark scheme abbreviations

/ separates alternative words / phrases within a marking point
// separates alternative answers within a marking point
underline actual word given must be used by candidate (grammatical variants accepted)
max. indicates the maximum number of marks that can be awarded
( ) the word / phrase in brackets is not required, but sets the context

Note: No marks are awarded for using brand names of software packages or hardware.

Question	Answer	Marks	Guidance
1	C	**1**	

Question	Answer	Marks	Guidance
2	B	**1**	

Question	Answer	Marks	Guidance
3	**One** mark for each correct line, **max. four** **use of arithmetic operator** **result**  	**4**	

Question	Answer	Marks	Guidance
4(a)	Verification is used to check that data has been copied from one medium to another without introducing errors // Ensure by comparison that two copies of the data are the same/haven't changed.	**1**	

Question	Answer	Marks	Guidance
4(b)	**One** mark per mark point • Data is entered twice and the two values compared • If the two entries match, the entry is accepted. // If the two entries do not match, re-entry is requested.	**2**	

Question	Answer	Marks	Guidance
5(a)	To check that the data entered matches a predefined pattern or structure. Do not accept checks the format as an answer	**1**	
5(b)(i)	Length (check)	**1**	

5(b)(ii)	**One** mark per mark point, **max. four**
	
	• Initial input
	• Use of appropriate iteration
	• Correct use of len() for string length
	• Correctly applied re-input
	• Messages used for initial input, re-input and final acceptance message
	
	Example program
	
	`passcode = input('Enter a 6 character string ')`
	`while len(passcode) != 6:`
	`  passcode = input('The string must have 6 characters ')`
	`print('String accepted')`
	
	OR
	
	`while True:`
	`  passcode = input("Please enter a string of six characters ")`
	`  if len(passcode) == 6:`
	`  print("String accepted")`
	`  break`
	`  else:`
	`  print("Passcode must be 6 characters, try again.")`
		4	

Question	Answer	Marks	Guidance
6(a)	**One** mark per mark point (Grey shaded parts given in question) • correct line 1 … • correct line 2 … • … correct line 3 **Example answer** [68] [50] [20] [0] [100] [45] [10] [70] line 1: [68, 50] [20, 0] [100, 45] [70, 10] line 2: [68, 50 20, 0] [100, 70, 45, 10] line 3: [100, 70, 68, 50, 45, 20, 10, 0]	**3**	
6(b)	**One** mark per correct statement, **max. four** For example: • Find the mid-point of the list of 7 elements • Compare the value at the mid-point/53 with 98 • Discard the right-hand part because all the elements are less than 98 • Find the mid-point of the remaining 3 element list • Compare the value at the mid-point/98 with 98 • The value has been found.	**4**	

Question	Answer	Marks	Guidance

7(a)	**One** mark per mark point (do not award initialise count as Python does not require this)
	
	• Line 02 / `highest = 2000`
	should be `highest = 0`
	
	• Line 04 / `while count in range(1000):`
	should be `for count in range(1000):`
	
	• Line 07 / `total = total + count`
	should be `total = total + numbers[count]`
	
	• Line 12 / `print('The average is ', highest / 1000)`
	should be `print('The average is ', total / 1000)`
	
	Corrected algorithm
	
	`01 numbers = [0] *1000`
	`02 highest = 0`
	`03 total = 0`
	`04 for count in range(1000):`
	`05  print('Please input integer number ', count + 1)`
	`06  numbers[count] = int(input())`
	`07  total = total + numbers[count]`
	`08  if numbers[count] > highest:`
	`09  highest = numbers[count]`
	`10 print('The highest number is ', highest)`
	`11 print('The total is ', total)`
	`12 print('The average is ', total / 1000)`
		4	

Question	Answer	Marks	Guidance
7(b)	**One** mark per mark point • Print statement with correct calculation of average, or use of a variable average. • Correct use of round() function set to two decimal places. For example: `print (round(total / 1000, 2))`	**2**	

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>8(a)</td><td>

One mark per mark point

- Correct `g` column
- Correct `s` column
- Correct `t` column
- Correct `a` column
- Correct `limit`, `i` and `value` columns
- Correct output column

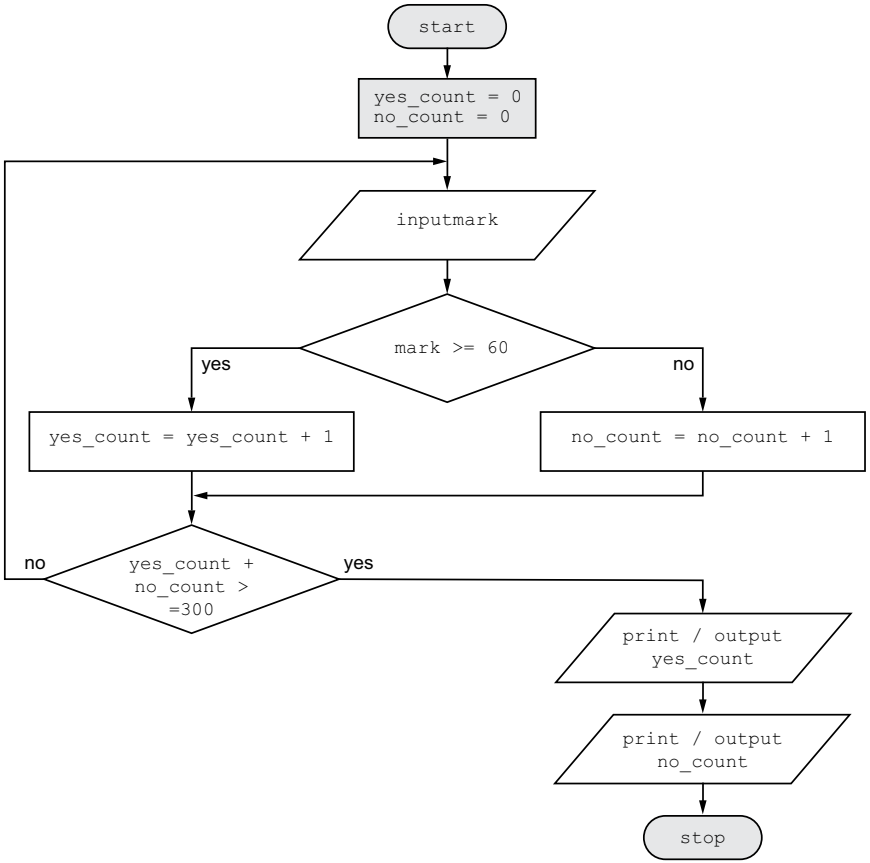
<table>
<tr><th>g</th><th>s</th><th>t</th><th>a</th><th>limit</th><th>i</th><th>value</th><th>output</th></tr>
<tr><td>–5000</td><td>5000</td><td>0</td><td>0</td><td>6</td><td>0</td><td>12</td><td></td></tr>
<tr><td>12</td><td></td><td>12</td><td></td><td></td><td>1</td><td>–50</td><td></td></tr>
<tr><td></td><td>–50</td><td>–38</td><td></td><td></td><td>2</td><td>1200</td><td></td></tr>
<tr><td>1200</td><td></td><td>1162</td><td></td><td></td><td>3</td><td>0</td><td></td></tr>
<tr><td></td><td></td><td>1162</td><td></td><td></td><td>4</td><td>–120</td><td></td></tr>
<tr><td></td><td>–120</td><td>1042</td><td></td><td></td><td>5</td><td>17</td><td></td></tr>
<tr><td></td><td></td><td>1059</td><td>176.5</td><td></td><td></td><td></td><td>g = 1200 s = –120</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>t = 1059 a = 176.5</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

</td><td>**6**</td><td></td></tr>
<tr><td>8(b)</td><td>

One mark per mark point

- Any **two** from finds/outputs the greatest, smallest, total and average of a set of numbers
- All **four** of finds/outputs the greatest, smallest, total and average of a set of numbers.

</td><td>**2**</td><td></td></tr>
</table>

Question	Answer	Marks	Guidance
9	**One** mark per mark point (Grey shaded parts given in question) • input mark • correct check that mark is greater than or equal to 60 • counting `yes_count` values • counting `no_count` values • checking for 300 entries (= 300 or >=300) • output of both counter values For example:  	**6**	

Question	Answer	Marks	Guidance
10(a)	**One** mark for an advantage of global variables, **max. one** • global variables can be accessed throughout the program • global variables do not need to be redefined in different modules **One** mark for an advantage of local variables, **max. one** • local variables can only be used in one subroutine which prevents accidental altering of data in other subroutines • memory used for local variables is removed once a subroutine has finished running, leading to memory efficiency.	**2**	
10(b)	**One** mark per correct statement, **max. three** For example: • Procedures can be used as basic building blocks when organising a program, • … they enable complex tasks to be broken down into smaller, more manageable tasks. // … they provide modularity. • Programs are more readable when they have modularity. • Enables collaboration between teams to construct / debug / testing large programs. • Procedures enable modular testing of large programs. • Procedures allow code to be reused within the same program or different programs.	**3**	

Question	Answer	Marks	Guidance

11	**One** mark per mark point
	
	• Open `'Quote.txt'` for read
	• Output the contents of `'Quote.txt'`
	• Convert the contents of `'Quote.txt'` to uppercase
	• Output the uppercase version of the file contents
	• Close `'Quote.txt'` // Use `with open … as`
	
	Example programs
	
	`x = open('Quote.txt')`
	`text = x.read()`
	`y = text.upper()`
	`print(y)`
	`x.close()`
	
	OR
	
	`x = open('Quote.txt')`
	`y = (x.read())`
	`print(y, y.upper())`
	`x.close()`
	
	OR
	
	`with open('Quote.txt', 'r') as file:`
	`  x = file.read()`
	`  print(x)`
	`  print(x.upper())`
		5	

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>12(a)</td><td>One mark for correct field

• <code>SubjectID</code>

One mark for correct purpose of a foreign key

• To establish / enforce a link between data in two tables.</td><td>2</td><td></td></tr>
<tr><td>12(b)</td><td>One mark for one correct text field and one correct integer field correctly identified (Grey shaded parts given in question)
Ignore table name <code>STUDENT</code>, <code>SUBJECT</code>

<table>
<tr><th>data type</th><th>field</th></tr>
<tr><td>text</td><td><code>STUDENT.StudentID //
STUDENT.StudentName //
STUDENT.TutorGroup //
STUDENT.SubjectID //
SUBJECT.SubjectID //
SUBJECT.SubjectName //</code></td></tr>
<tr><td>integer</td><td><code>SUBJECT.PracticalExams //
SUBJECT.TheoryExams //
SUBJECT.MaxClassSize //</code></td></tr>
</table>
</td><td>1</td><td></td></tr>
</table>

Question	Answer	Marks	Guidance

12(c)	**One** mark per correct line (Bold parts given in question)
	
	Correct answers:
	
	`SELECT StudentName, TutorGroup`
	`FROM STUDENT`
	`INNER JOIN SUBJECT`
	`ON STUDENT.SubjectID = SUBJECT.SubjectID`
	`WHERE SUBJECT.SubjectName = "Computer Science";`
	
	OR
	
	`SELECT StudentName, TutorGroup`
	`FROM SUBJECT`
	`INNER JOIN STUDENT`
	`ON SUBJECT.SubjectID = STUDENT.SubjectID`
	`WHERE SUBJECT.SubjectName = "Computer Science";`
		5	

Question	Answer	Marks	Guidance

13(a)	**One** mark per mark point
	
	• `random_numbers` initialised with [0] * 1 000 000
	• `count_store[index][0]` initialised with integers 1 to 6 inclusive **and** `count_store[index][1]` initialised with 0 * 6
	
	Example program
	
	`random_numbers = [0] * 1000000`
	`count_store = [[0 for i in range(2)]for j in range(6)]`
	`for index in range(6):`
	`  count_store[index][0] = index + 1`
	`  count_store[index][1] = 0`
		2	

Question	Answer	Marks	Guidance

13(b)	**One** mark per mark point
	
	• `import random` applied
	• Appropriate iteration used to enable 1 000 000 numbers to be generated
	• Appropriate random number method used to generate an integer between 1 and 6 inclusive
	
	Example program
	
	`import random`
	`for index in range(1000000):`
	`  random_numbers[index] = random.randint(1,6)`
		3	

13(c)	**One** mark per mark point
	
	• Appropriate iteration used (6 times)
	• Correct use of `.count()` with `random_numbers` list
	• Correct value being counted `count_store[index][0]`
	• Correct assignment to `count_store[index][1]`
	
	Example program
	
	`for index in range(6):`
	`  count_store[index][1] = random_numbers.count(count_store[index][0])`
		4	

Question	Answer	Marks	Guidance

13(d)	**One** mark per mark point
	
	• Bubble sort used with two loops
	• Correct limit for inner loop (5)
	• Correct use of selection to compare two adjacent elements using < for descending order
	• Swapping of elements in `count_store[index][1]` if they are not in order
	• Swapping of elements in `count_store[index][0]` to maintain data integrity
	• All data from `count_store` printed with appropriate message included.
	
	Example program
	
	`swap = True`
	`while swap:`
	`    swap = False`
	`    for index in range (5):`
	`        if count_store[index][1] < count_store[index + 1][1]:`
	`            temp1 = count_store[index][0]`
	`            temp2 = count_store[index][1]`
	`            count_store[index][0] = count_store[index + 1][0]`
	`            count_store[index][1] = count_store[index + 1][1]`
	`            count_store[index + 1][0] = temp1`
	`            count_store[index + 1][1] = temp2`
	`            swap = True`
	`for index in range(6):`
	`    print ('The integer', count_store[index][0], 'was generated'`
	`          , count_store[index][1], 'times')`
		6	

BLANK PAGE