Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/22**

Paper 2 Written Paper **October/November 2017**

MARK SCHEME

Maximum Mark: 75

Published

Question	Answer	Marks
1(a)(i)	<table><tr><th>Data value</th><th>Data type</th></tr><tr><td>`FALSE`</td><td>**`BOOLEAN`**</td></tr><tr><td>`03/03/2013`</td><td>**`DATE // DATETIME`**</td></tr><tr><td>`35`</td><td>**`INTEGER`**</td></tr><tr><td>`"INTEGER"`</td><td>**`STRING`**</td></tr><tr><td>`3.5`</td><td>**`REAL`**</td></tr><tr><td>`"35"`</td><td>**`STRING`**</td></tr></table> One mark for each data type Mark first data type given in each case	6
1(a)(ii)	1D Array // 1D List	1
1(a)(iii)	Ability to recognise: • selection statement • iteration statement • assignment statements • data declarations / structures / data types / use of variables or objects • modular structure / functions / procedures / subroutines • subroutine parameters • Specific types of statement e.g. Input, Output, File operations • Code format • Operators Mark as follows: Any two from above, or valid alternative Accept by example	2
1(b)(i)	<table><tr><th>Data</th></tr><tr><td></td></tr><tr><td>67 // 0100 0011 // 043h</td></tr><tr><td>65 // 0100 0001 // 041h</td></tr><tr><td>71 // 0100 0111 // 047h</td></tr><tr><td>69 // 0100 0101 // 045h</td></tr><tr><td></td></tr></table> One mark for 67 and 65 One mark for 71 and 69 Accept binary, denary or hex values (hex must be clearly indicated) Max one mark if blank cell anywhere in sequence Ignore any data values before or after the four characters	2

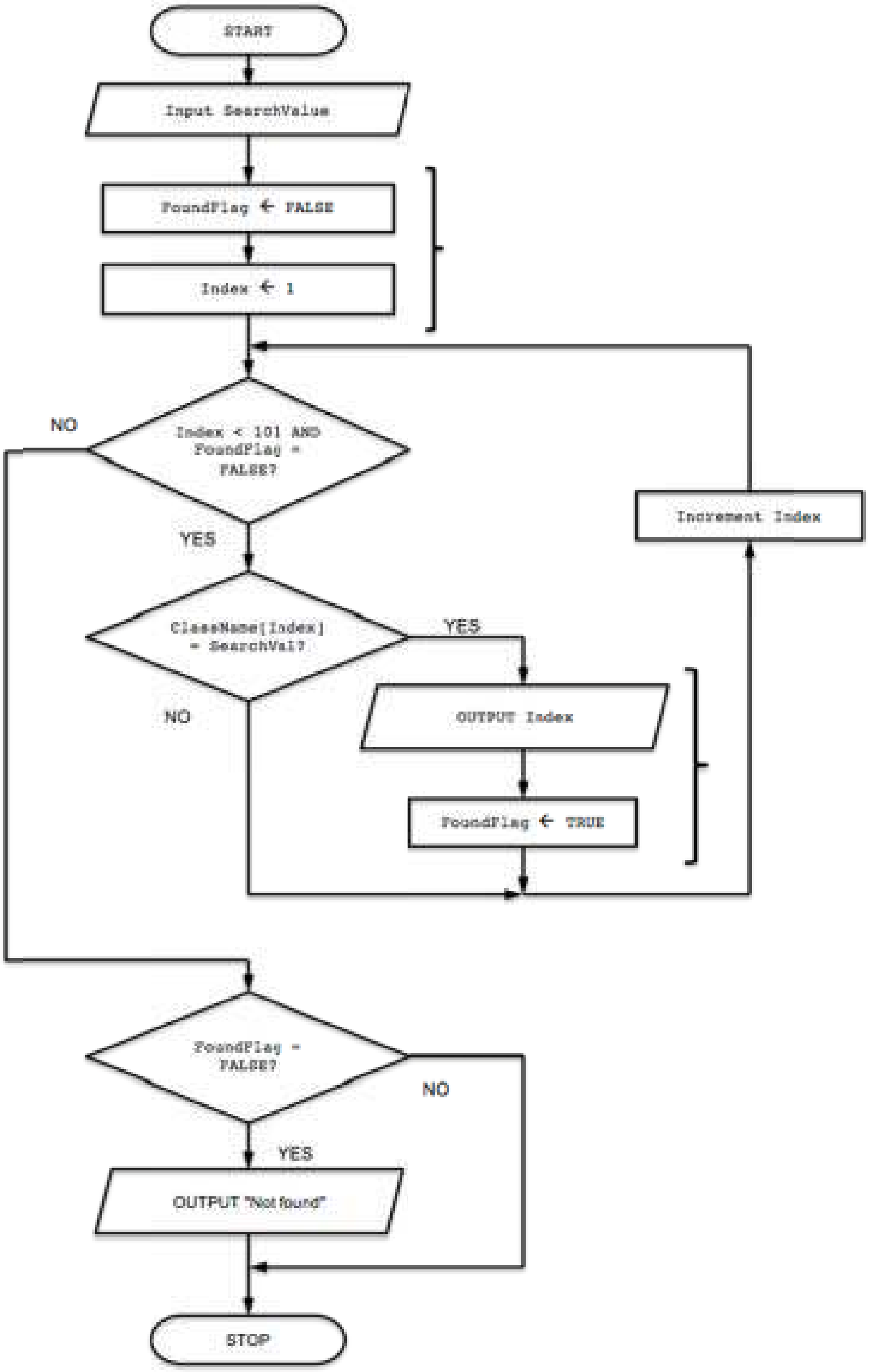
Question	Answer	Marks
1(b)(ii)	• A value representing the number of characters ... ... stored at beginning of string OR • Terminator / special character ... ... stored to indicate the end of string One mark for each phrase or equivalent.	2
1(c)	Explanation includes: • to pass values to/from the subroutine • to produce re-useable code • to avoid global variables • to allow recursion One mark per answer	**Max 3**
1(d)(i)	27: `MyGrade` assigned the value "`Fail`" 101: Output the text "`Invalid Value Entered`" Ignore minor spelling mistakes	2

1(d)(ii)	
	```
	IF MyMark >= 75 AND MyMark <=100
	   THEN
	      MyGrade ← "Distinction"
	   ELSE

	      IF MyMark >= 35 AND MyMark <=74
	         THEN
	            MyGrade ← "Pass"
	         ELSE

	            IF MyMark >= 0 AND MyMark <=34
	               THEN
	                  MyGrade ← "Fail"
	               ELSE

	               OUTPUT "Invalid value entered"
	            ENDIF
	      ENDIF
	ENDIF
	```
	One mark for each of:
	
	• One correct range test
	• 'IF' equivalent (nested or not) to three CASE range tests...
	• ... with three corresponding assignments
	• Equivalent of CASE `OTHERWISE` with corresponding `OUTPUT` statement
	• Matching (three) `ENDIF`s (Or one if `ELSIF`s used)
	
	Max 4 if solution doesn't work under all circumstances // is not functionally equivalent to CASE
		5

Question	Answer	Marks
2(a)	Mark as follows: • To search for a given value in the array • and output the position in the array if the search value is found • if search value not found then output "Not found"	**Max 2**
2(b)	 START Input SearchValue FoundFlag ← FALSE Index ← 1 NO Index < 101 AND FoundFlag = FALSE? Increment Index YES ClassName[Index] = SearchVal? YES NO OUTPUT Index FoundFlag ← TRUE FoundFlag = FALSE? NO YES OUTPUT "Not found" STOP	**9**

Question	Answer	Marks
2(b)	Mark as follows: • One mark for `START` **and** `STOP` / `END` • One mark for each bracketed pair • One mark for each of other labelled symbol (decision box shape must be correct) • Allow F/T from incorrect decision symbol Full marks should be awarded for functionally equivalent solutions.	

Question	Answer	Marks
3	see table below; One mark for each correct row	**Max 8**

Line number	Error	Correction
01	Wrong procedure name – "SortArray"	PROCEDURE ArraySort
02	Wrong data type - `CHAR`	`DECLARE Temp:` **`STRING`**
03	Variables undefined	`DECLARE FirstID, SecondID, I, J : INTEGER`
04	Wrong 'Value2' of 100	`FOR I ←` **`1 TO 99`**
05	Wrong range	`FOR J ←` **`1 TO (100 - I)`**
06/07	Wrong function - `MODULUS`	Replace `MODULUS` with `TONUM`: `FirstID ←` **`TONUM`**`(LEFT(Product[J],`
06/07	Wrong value of 6	Should be 4: `FirstID ← TONUM(LEFT(Product[J],`
10	Assigning wrong value to `Temp`	`Temp ←` **`Product[J]`**
11	Assigning wrong value to `Product[I]`	**`Product[J]`** `← Product[J + 1]`

13/14	Lines reversed	`13 ENDIF`
		`14 ENDFOR`

One mark for each correct row

Question	Answer	Marks

4(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.
	
	`PROCEDURE TestRandom (Repetitions : INTEGER)`
	`  DECLARE Frequency : ARRAY [1 : 10] OF INTEGER`
	`  DECLARE Expected : REAL / INTEGER //allow either`
	`  DECLARE NextRandom : INTEGER`
	`  DECLARE N : INTEGER`
	
	`  FOR N ← 1 TO 10`
	`     Frequency[N] ← 0`
	`  ENDFOR`
	
	`  Expected ← INT(Repetitions / 10)`
	
	`  CALL RANDOMIZE()  //Set random seed`
	
	`  FOR N ← 1 TO Repetitions`
	`     NextRandom ← INT(RND() * 10) + 1`
	`     Frequency[NextRandom] ← Frequency[NextRandom] + 1`
	`  ENDFOR`
	
	`  OUTPUT "The expected frequency is " & Expected`
	
	`  OUTPUT "Number    Frequency    Difference"`
	
	`  FOR N ← 1 TO 10`
	`     OUTPUT N & "    " & Frequency[N] & "    " & Frequency[N] – Expected`
	`  ENDFOR`
	`ENDPROCEDURE`
	
	Mark as follows:
	1. Procedure heading (including parameter)
	2. Array declaration – 10 or 11 elements
	3. Array declaration – data type
	4. Variable declaration for a loop counter (integer) or expected frequency (integer or real)
	5. Variable declaration for next random value
	
	(For Python solutions, mark points 1 to 4 may be gained by suitable comments)
	
	6. Initialise all elements of array
	7. To set all elements to zero
	8. Calculate expected frequency
		16

Question	Answer	Marks
4(a)	9. Loop to generate required number of random values 10. Use of relevant `RANDOM()` function **in a loop** 11. Generate random integer value in the range 1 to 10 **in a loop** 12. Increment (array) element **in a loop** 13. Output expected frequency message **not in any loop** 14. Output column header text 15. (Loop to) output each row 16. ... including three correct values (spaces optional)	
4(b)	• Single-stepping – to allow program statements to be executed one at a time • Breakpoints – to pause / stop the program at a specific line / statement • Variable / expression watch window – to monitor the value of variables / expressions as the program is run One mark for each Feature (text as above or equivalent) + 1 for meaningful explanation of use in context.	**6**
4(c)	Program is probably working correctly if: • Header is present giving frequency as 20 • Column headers are present • All rows are present (1 to 10) • The difference is calculated correctly • Output is formatted correctly • Total differences should be zero • Sum of Frequencies should be 200	**Max 2**

Question	Answer	Marks

5	
	```
	PROCEDURE RemoveDetails
	  DECLARE FileLine: STRING
	  DECLARE MemberToDelete: STRING

	  OPENFILE "EmailDetails.txt" FOR READ
	  OPENFILE "NewEmailDetails.txt" FOR WRITE

	  INPUT MembershipNumber
	  WHILE NOT EOF("EmailDetails.txt")
	     READFILE "EmailDetails.txt", FileLine
	     IF LEFT(FileLine, 4) <> MembershipNumber
	        THEN
	           WRITEFILE "NewEmailDetails.txt", FileLine
	     ENDIF
	  ENDWHILE

	  CLOSEFILE "EmailDetails.txt"
	  CLOSEFILE "NewEmailDetails.txt"

	ENDPROCEDURE
	```
	Mark as follows:
	1. Procedure declaration and end. No parameters.
	2. Variable declaration of `STRING` for variable `FileLine` (or similar)
	3. Input the `MembershipNumber` of the person who has left
	4. Open `EmailDetails` for READ
	5. Open `NewEmailDetails` for `WRITE`
	6. Correct loop checking for `EOF(EmailDetails)`
	7. Reading a line from `EmailDetails.txt` **in a loop**
	8. Correct check for `MemberToDelete` **in a loop**
	9. Writing a line to `NewEmailDetails.txt` **in a loop**
	10. Closing both files (not in a loop)
		Max 9

Appendix - Program Code Example Solutions

Q4 (a): Visual Basic

```
Dim random As New Random()

Sub TestRandom(ByVal repetitions As Integer)
   Dim randinrange As Integer
   Dim i As Integer
   Dim num(1 To 10) As Integer
   Dim freq As Integer
   Dim difference As Integer

   freq = repetitions / 10   'calculate expected frequency

   For i = 1 To 10           'initialise array to store total frequencies
      num(i) = 0
   Next i

   For i = 1 To repetitions 'generate random numbers & increment 
appropriate freq
      randinrange = random.Next(1, 11)
      num(randinrange) = num(randinrange) + 1
   Next i

   Console.WriteLine("The expected frequency is " & freq)   'report header
   Console.WriteLine("Number   Frequency    Difference")    'column headers

   For i = 1 To 10   'calc & display difference between expected and actual 
freq
      difference = num(i) - freq
      Console.WriteLine(i & "       " & num(i) & "        " & difference)
   Next i

End Sub
```

Other possible ways of calculating a random number in VB include:

```
   randinrange = CInt(Math.Floor((upperbound - lowerbound + 1) * Rnd())) + 
lowerbound

   randinrange = math.round((Rnd()*9)+1)

   randinrange = CInt(Math.Ceiling(Rnd() * 9
```

Q4 (a): Pascal

```
procedure TestRandom(var Repetitions : integer);
   var
      Frequency : array[1..10] :  integer;
      Expected, NextRandom, N  : integer;

   begin
      Expected := Round(Repetitions/10);
      for N := 1 to 10 do
         Frequency[N] := 0;

      for N := 1 to Repetitions do
      begin
         NextRandom := random(10)+1;
         Frequency[NextRandom] := Frequency[NextRandom]+1;
      end;

      writeln ('The expected frequency is ', Expected);
      writeln ('Number Frequency Difference');

      for N := 1 to 10 do
         writeln ('   ',N,'      ',Frequency[N],'       ',Frequency[N]-
Expected);

   end;
```

Q4 (a): Python

```
# frequency is an array from 1 to 10 of type integer;
# nextNumber is an integer which stores the created random number
# expected is an integer which stores the expected frequency of each number

def TestRandom (repetitions):
   import random
   frequency = [0 for i in range(1,11)]  # initialise each frequency count
to 0

   expected = repetitions / 10

   for i in range(1, repetitions + 1):
      nextNumber = random.randint(1,10)
      frequency[nextNumber] = frequency[nextNumber]+ 1

   print ("The expected frequency is ",  expected)
   print("   Number  Frequency  Difference")

   for i in range(1,11):
     print ("    ", i, "    ", frequency[i],"    ", frequency[i] -
expected)
```

Alternative:

```
def TestRandom (repetitions):
   expected = repetitions / 10     ## initialised as real/integer
                                   ## NextRandom and N defined as integers
   frequency =[0,0,0,0,0,0,0,0,0,0,0]  ## defined as an array and
initialised to zero

   for n in range (0,repetitions):
      nextNumber = randint(1, 10)
      frequency[nextNumber] += 1

   print ('The expected frequency is ', expected)

   print ('Number  Frequency  Difference')
   for n in range (1, 11):
     print (n,'      ',frequency[n],'          ',frequency[n] - expected)
```

Alternative:

```
   frequency =[0]*11        ## alternate way to initialise array to zero
   frequency =[]            ## empty array/list
```

Alternative:

```
    for n in range (1,11):
       frequency[n-1] = 0       ##alternate way to initialise array to zero
```

Alternative:

```
for n in range (0,11):  ##alternate way to initialise array to zero
    frequency.append(0)
```