Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/22**

Paper 2 Written Paper **May/June 2019**

MARK SCHEME

Maximum Mark: 75

Published

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1: Marks must be awarded in line with: • the specific content of the mark scheme or the generic level descriptors for the question • the specific skills defined in the mark scheme or in the generic level descriptors for the question • the standard of response required by a candidate as exemplified by the standardisation scripts.
GENERIC MARKING PRINCIPLE 2: Marks awarded are always **whole marks** (not half marks, or other fractions).
GENERIC MARKING PRINCIPLE 3: Marks must be awarded **positively**: • marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate • marks are awarded when candidates clearly demonstrate what they know and can do • marks are not deducted for errors • marks are not deducted for omissions • answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.
GENERIC MARKING PRINCIPLE 4: Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.
GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks
1(a)(i)	• a sequence of steps / stages / instructions • to implement a task // solution to a problem Allow alternatives to sequence providing meaning is clear.	2
1(a)(ii)	**Input:** • e.g. `INPUT MyVar` // `READFILE MyFile, MyString` **Process:** • e.g. `NextChar ← 'X'` // `Count ← Count + 1` Output:: • e.g. `OUTPUT "Hello World"` // `WRITEFILE "LogFile.txt", SomeData` Mark as follows: One mark for each stage (**Input and Process**) One mark for each pseudocode example	5
1(b)(i)	see table below No quotes for row 1 Quotes (single or double) for row 5	5
1(b)(ii)	see table below One mark per data type	5

1(b)(i)

Expression	Evaluates to
`STRING_TO_NUM(RIGHT(ID, 3))`	**234.0 / 234**
`INT(Height * Children)`	**11**
`IsMarried AND Married < 31/12/1999`	**TRUE**
`LENGTH(ID & NUM_TO_STRING(Height))`	**8**
`MID((ID, INT(Height) - Children, 2)`	**"23"**

1(b)(ii)

Variable	Data type
`Married`	DATE
`ID`	STRING
`MiddleInitial`	CHAR
`Height`	REAL
`IsMarried`	BOOLEAN

Question	Answer	Marks
2(a)(i)	• To make a more manageable / understandable solution • To support modular design	**1**
2(a)(ii)	• Allows the subroutine to be called from many / multiple places • Subroutine may be (independently) tested and debugged • If the task changes the change needs to be made only once • Reduces unnecessary duplication / program lines • Allows teams to work on different parts of the solution	**3**
2(a)(iii)	**Type of subroutine**: Function **Justification**: It returns a value // assigns a value to variable `Answer` One mark for type One mark for justification	**2**
2(b)	• An editor is used to produce / write / modify the source code / program / high-level language code **OR by example:** An editor provides (features such as) context-sensitive prompts / dynamic syntax checking etc. • A translator (compiler) is used to translate / convert the source code / program / high-level language code into object code / machine code / an executable file. **OR** A translator (interpreter) is used to translate the source code / program / high-level language code line by line • A debugger is used to test the program / detect errors (and correct errors) in the program. One mark per bullet point	**3**

Question	Answer	Marks
2(c)	**Control structure**: A (pre-) <u>conditional</u> loop **Function of code**: • Check if `Result` is less than 20 and If true, calls `ResetSensor` with parameter value 3… • … and assign the value returned by `GetSensor` with parameter value 3 to `Result` • Loop until `Result >= 20` **OR** **Control structure**: A selection // conditional statement **Function of code**: • Check if `Result` is less than 20 and If true, calls `ResetSensor` with parameter value 3... • … and assign the value returned by `GetSensor` with parameter value 3 to `Result` One mark for control structure, maximum two for function Function of code marks independent of answer to control structure	**3**

Question	Answer	Marks
3(a)(i)	<u>`PROCEDURE SubA`</u> <u>`(A : STRING, B : INTEGER,`</u> <u>`BYREF C : CHAR)`</u> One mark for each underlined part Ignore `BYVAL` for parameter A and/or parameter B Parameter order / names not important but must be correct data types	**3**
3(a)(ii)	<u>`Function SubB`</u> <u>`(D : STRING, E : INTEGER)`</u> <u>`RETURNS BOOLEAN`</u> One mark for each underlined part Ignore `BYVAL` for parameter D and/or parameter E Parameter order / names not important but must be correct data types	**3**
3(b)	• Selection • Iteration • Sequence One mark per bullet to max. 2	**2**

Question	Answer	Marks
4(a)(i)		5

Index	NextChar	Selected	NewValue	NewString
		0		"0"
1	'1'			"01"
2	'2'			"012"
3	'∇'		12	
		12		
				"0"
4	'3'			"03"
5	'4'			"034"
6	'∇'		34	
		34		
				"0"
7	'5'			"05"
8	'∇'		5	
				"0"
9	'∇'		0	
				"0"
10	'3'			"03"
11	'9'			"039"

One mark for each column.

If no mark for columns, award one mark for initialisation of `Selected` to 0 **and** `Newstring` to '0' (single or double quotes).

Question	Answer	Marks
4(a)(ii)	34	1

Question	Answer	Marks
4(b)(i)	• The final value (in the string) is the largest value (39) and is not considered // the final comparison with variable `Selected` is not made • The loop terminates at the end of the string (the character 9) // there wasn't a final space / non-numeric digit One mark per bullet.	**2**
4(b)(ii)	• Check the (final) value of `NewString` **after** the loop... • ...and see if it is greater than `Selected` (repeat the existing conditional clause) **OR** • Amend the algorithm to add a space character / non-numeric character to the end of the string... • ...before the `FOR` loop / at the start of the function One mark per bullet point Accept alternative workable solution	**2**

Question	Answer	Marks
5(a)	One mark for each of: • Open the file • Set a count to zero • Loop until end of file // no more lines to read • Increment the count each time a line is read in a loop Maximum 3 marks	**3**

5(b)	`PROCEDURE CountLines(FileName : STRING)`
	
	`   DECLARE NumLines : INTEGER`
	`   DECLARE Dummy : STRING`
	
	`   NumLines ← 0`
	
	`   OPENFILE FileName FOR READ`
	
	`   WHILE NOT EOF(FileName)`
	`      READFILE FileName, Dummy`
	`      NumLines ← NumLines + 1`
	`   ENDWHILE`
	
	`   CLOSEFILE FileName`
	
	`   OUTPUT "Number of lines in the file : ", NumLines`
	
	`ENDPROCEDURE`
	
	One mark for each of the following:
	
	1. Procedure header and end, including parameter
	2. Declaration **and** initialisation of a local `INTEGER` to count lines (e.g. `NumLines`)
	3. `OPEN` file in read mode **and** `CLOSE` file
	4. `WHILE` loop stopping when `EOF(FileName)`
	5. Read a line from the file **and** increment `NumLines` **in a loop**
	6. Output a message plus the `NumLines` **outside a loop**
		6

Question	Answer	Marks

6(a)	'Pseudocode' solution included here for development and clarification of mark scheme.
	
	Programming language example solutions appear in the Appendix.
	
	`FUNCTION GetInfo() RETURNS STRING`
	
	`   DECLARE ID : STRING`
	`   DECLARE PreferredName : STRING`
	`   DECLARE Valid : BOOLEAN`
	
	`   Valid ← FALSE`
	
	`   WHILE Valid = FALSE`
	`      OUTPUT "Please Enter a valid ID"`
	`      INPUT ID`
	
	`      IF LENGTH(ID) = 5 AND LEFT(ID, 1) >= 'A'__`
	`                        AND LEFT(ID, 1) <= 'Z'__`
	`                        AND ISNUM(RIGHT(ID, 4))`
	`         THEN`
	`            Valid ← TRUE`
	`      ENDIF`
	
	`   ENDWHILE`
	
	`   OUTPUT "Please enter preferred name"`
	`   INPUT PreferredName`
	`   RETURN ID & '*' & PreferredName`
	
	`ENDFUNCTION`
	
	One mark for each of the following:
	
	1. Function header and end (where appropriate)
	2. Local variables used are declared (commented in python)
	3. Prompt **and** input for ID (until valid) **and** preferred name
	4. Conditional loop repeating while `ID` is invalid
	5. test length **in a loop**
	6. test first character **in a loop**
	7. test last four characters **in a loop**
	8. Concatenate using correct separator character **and** return resulting string
		8

Question	Answer	Marks

6(b)	'Pseudocode' solution included here for development and clarification of mark scheme.
	
	Programming language example solutions appear in the Appendix.
	
	`PROCEDURE TopLevel()`
	
	`    DECLARE Response : CHAR`
	`    DECLARE InputData : STRING`
	`    DECLARE Success : BOOLEAN`
	
	`    Response ← 'Y'`
	
	`    WHILE Response = 'Y'`
	`        InputData ← GetInfo()`
	`        IF LEFT(InputData,1) < 'N'`
	`            THEN`
	`                Success ← WriteInfo(InputData, "File1.txt")`
	`            ELSE`
	`                Success ← WriteInfo(InputData, "File2.txt")`
	`        ENDIF`
	`        IF NOT Success`
	`            THEN`
	`                Response ← 'N'`
	`            ELSE`
	`                OUTPUT "Enter details for another student? (Y/N)"`
	`                INPUT Response`
	`        ENDIF`
	
	`    ENDWHILE`
	
	`ENDPROCEDURE`
	
	One mark for each of the following:
	
	1. Procedure header and end
	2. Conditional loop terminated with user input
	3. call to `GetInfo()` **in a loop**
	4. check first character of returned `UserID` value **in a loop**
	5. call(s) to `WriteInfo`() in both cases **…**
	6. … with two `STRING` parameters **in a loop**
	7. exit procedure if `WriteInfo()` unsuccessful **in a loop**
	8. if `WriteInfo()` successful, prompt and check input to repeat / exit **in a loop**
		8

6(c)	`FUNCTION WriteInfo (FileData : STRING, Filename : STRING) RETURNS BOOLEAN` One mark per underlined section	**3**

*** End of Mark Scheme – example program code solutions follow ***

Program Code Example Solutions

Q6 (a): Visual Basic

```
Function GetInfo() As String
   Dim ID As String = ""
   Dim PreferredName As String = ""
   Dim Valid As Boolean = False
   While Valid = False
      Console.Write("Please enter a valid ID : ")
      ID = Console.ReadLine()
      If Len(ID) = 5 And Left(ID, 1) >= "A" And Left(ID, 1) <= "Z" __
         And IsNumeric(Right(ID, 4)) Then
         Valid = True
      End If
   End While
   Console.Write("Please enter preferred name : ")
   PreferredName = Console.ReadLine()
   Return ID & "*" & PreferredName
End Function
```

Alternative:

```
Function GetInfo() As String

   Dim ID As String
   Dim PreferredName As String
   Dim Valid As Boolean
   Dim Number As String
   Dim Size As Integer
   Dim i As Integer

   Valid = False

   While Valid = False
      Console.WriteLine("Please Enter a valid ID")
      ID = Console.ReadLine()
      Size = Len(ID)

      If (Size = 5) And ((Left(ID, 1) >= "A") And (Left(ID, 1) <= "Z"))
Then
         Valid = True
         For i = 2 To 5
            Number = Mid(ID, i, 1)
            If (Number < "0") Or (Number > "9") Then
               Valid = False
            End If
         Next
      End If
   End While

   Console.WriteLine("Please enter preferred name")
   PreferredName = Console.ReadLine()
   Return (ID & "*" & PreferredName)
End Function
```

Q6 (a): Pascal

```
function GetInfo() : String;
var
   ID : String;
   PreferredName : String;
   Valid : Boolean;
   Value, Code : Integer;
begin
   Valid := false;
   while not Valid do
   begin
      Write('Please enter a valid ID : ');
      Readln(ID);
      if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') then
         Valid := true;
      Val(Copy(ID, 2, 4), Value, Code);
      if Code <> 0 then
         Valid := false;
   end;
   Write('Please enter preferred name : ');
   Readln(PreferredName);
   GetInfo :=  ID + '*' + PreferredName;
end;
```

Free Pascal

```
function GetInfo() : String;
var
   ID : String;
   PreferredName : String;
   Valid : Boolean;
   Value, Code : Integer;
begin
   Valid := false;
   while not Valid do
   begin
      Write('Please enter a valid ID : ');
      Readln(ID);
      if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') __
         and (IsNumber(SubStr(ID, 2, 4))) then
         Valid := true;
   end;
   Write('Please enter preferred name : ');
   Readln(PreferredName);
   result :=  ID + '*' + PreferredName;
end;
```

Q6 (a): Python

```
def GetInfo() :
   ID = ""                  # string variable
   PreferredName = ""       # string variable
   Valid = False            # Boolean variable
   while not Valid :
      ID = input("Please enter a valid ID : ")
      if len(ID) == 5 and ID[0] >= "A" and ID[0] <= "Z" and
ID[1:].isnumeric() :
         Valid = True
   PreferredName = input("Please enter preferred name : ")
   return ID + "*" + PreferredName
```

Q6 (b): Visual Basic

```
Sub TopLevel()
   Dim Response As String = "Y"
   Dim InputData As String = ""
   Dim Success As Boolean = True
   While Response = "Y"
      InputData = GetInfo()
      If Left(InputData, 1) < "N" Then
         Success = WriteInfo(InputData, "File1.txt")
      Else
         Success = WriteInfo(InputData, "file2.txt")
      End If
      If Not Success Then
         Response = "N"
      Else
         Console.Write("Enter details for another student? Y/N ")
         Response = Console.ReadLine()
      End If
   End While
End Sub
```

Q6 (b): Pascal

```
procedure TopLevel();
var
   Response : Char;
   InputData : String;
   Success : Boolean;
begin
   Response := 'Y';
   while Response = 'Y' do
   begin
      InputData := GetInfo();
      if InputData[1] < 'N' then
         Success := WriteInfo(InputData, 'File1.txt')
      else
         Success := WriteInfo(InputData, 'File2.txt');
      if not Success then
         Response := 'N'
      else
      begin
         Write('Enter details for another student? (Y/N) ');
         Readln(Response);
      end;

   end;
end;
```

Q6 (b): Python

```
def TopLevel() :
   Response = "Y"            # string/character variable
   InputData = ""            # string variable
   Success = True            # Boolean variable
   while Response == "Y" :
      InputData = GetInfo()
      if InputData[0] < "N" :
         Success = WriteInfo(InputData, "File1.txt")
      else :
         Success = WriteInfo(InputData, "File2.txt")
      if not Success :
         Response = "Y"
      else :
         Response = input("Enter details for another student? (Y/N) ")
```