Past-paper walk-through

Programming Basics ▪ 11.1

eXercise

Questions in this set

- 9618/21 N25 Q1 ▪ 10 marks
- 9618/22 N25 Q1 ▪ 11 marks
- 9618/21 J25 Q2 ▪ 10 marks
- 9618/21 N24 Q1 ▪ 9 marks
- 9618/23 N24 Q1 ▪ 10 marks
- 9618/23 N24 Q6 ▪ 6 marks
- 9618/21 J24 Q1 ▪ 12 marks
- 9618/22 J24 Q1 ▪ 11 marks

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 1

9618/21 N25 ▪ Q1 ▪ 10 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program is being developed to meet a particular customer requirement.
(a) The program contains these variables:

Variable	Data type
MyChar	CHAR
MyString	STRING
MyInt	INTEGER
MyDOB	DATE

Question 1

Complete the table by filling in the gaps using functions and/or operators from the **insert**. Each expression must be valid.

Expression
`MyString <- 'X' ___ MyString`
`MyChar <- ___ ("ABCD", ___ , ___ )`
`MyString <- ___ ( ___ (MyDOB))`
`MyInt <- ___ ( ___ (MyString) / 2)`

Question 1

[4]

Question 1

Test description	Test method
carried out as soon as a program module has been coded	alpha
carried out as program modules are being combined	
completed by the developers without referring to the code	
completed by the customer	

[3]

Question 1

Expression
MyString ← 'X' ..... MyString
MyChar ← ..... ("ABCD", ... ,.....)
MyString ← ..... (..... (MyDOB))
MyInt ← ..... (..... (MyString) / 2)

[4]

Question 1

Variable	Data type
MyChar	CHAR
MyString	STRING
MyInt	INTEGER
MyDOB	DATE

Question 1

9618/21 N25 ▪ Q1 ▪ 10 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program is being developed to meet a particular customer requirement.
(b) Different test methods will be used at different stages of the program development.
Complete the table by identifying the test method that matches the test description.
The first row has been completed for you.

Test description	Test method
carried out as soon as a program module has been coded	alpha
carried out as program modules are being combined	___
completed by the developers without referring to the code	___
completed by the customer	___

Question 1

[3]

Question 1

9618/21 N25 ▪ Q1 ▪ 10 marks

Refer to the **insert** for the list of pseudocode functions and operators.

A program is being developed to meet a particular customer requirement.

(c) During the alpha testing stage, an Integrated Development Environment (IDE) is used to help locate an error that has been identified. The IDE report window feature is used to examine the values assigned to variables.

Explain how **two** other IDE features are used together with the report window feature to help locate the error. **[3]**

Solution 1

(a) Worked steps

There is one question to ask on every row, and it is always the same one: **what type does the variable on the left need, and does the expression on the right produce that type?** An expression is "valid" here only when the two match.
The declared types are the whole of the data:

- `MyChar` is `CHAR` — exactly one character.
- `MyString` is `STRING` — any number of characters.
- `MyInt` is `INTEGER` — a whole number.
- `MyDOB` is `DATE`.

Solution 1

Now take the rows one at a time.

- `MyString <- 'X' & MyString` — `&` is the concatenation operator, and joining a character onto a string gives a string. Anything that returns a `STRING` earns this mark.
- `MyChar <- MID("ABCD", 1, 1)` — `MID(string, start, length)` cuts a piece out of a string. The **length argument must be 1**, because only a one-character result can go into a `CHAR`. The start position may be 1, 2, 3 or 4; all four are inside `"ABCD"`.
- `MyString <- NUM_TO_STR(MONTH(MyDOB))` — the date functions `DAY`, `MONTH`, `YEAR` and `DAYINDEX` all return an `INTEGER`, which cannot be assigned to a `STRING` directly, so `NUM_TO_STR` converts it. Any one of the four date functions is accepted.
- `MyInt <- INT(LENGTH(MyString) / 2)` — `LENGTH` returns an `INTEGER`, but `/` produces a `REAL`, so the result must be put back through `INT` before it can go into `MyInt`.

Solution 1

The two easy marks to lose are both type mistakes: dropping `NUM_TO_STR` on row three, and dropping `INT` on row four because `LENGTH` "already returns an integer" — the division is what changes the type.

Mark scheme

Expression
`MyString ← 'X' & MyString`
`MyChar ← MID ("ABCD", 1 / 2 / 3 / 4 , 1)`
`MyString ← NUM_TO_STR (DAY / MONTH / YEAR / DAYINDEX (MyDOB))`
`MyInt ← INT (LENGTH(MyString) / 2)`

Solution 1

One mark per row

Solution 1

(b) Mark scheme

Test description	Test method
carried out as soon as a program module has been coded	**alpha**
carried out as program modules are being combined	**integration**
completed by the developers without referring to the code	**black-box**
completed by the customer	**acceptance** / beta

Solution 1

One mark per row (2 to 4)

Solution 1

(c)

Mark scheme

One mark per point:

1. reference to a **breakpoint**
2. reference to **single stepping**
3. Correct explanation of **both** e.g. to stop program execution at specific point / line / statement / instruction and to execute one line / statement / instruction at a time to (locate an/the error)

One mark for each bulleted point

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(a) The table contains pseudocode examples.

Each example may contain statements that relate to one or more of:

- selection
- iteration (repetition)
- subroutines (procedures or functions).

Complete the table by placing **one or more** ticks (✓) in each row.

Question 1

Pseudocode example	Selection	Iteration	Subroutine
IF Status = FALSE THEN FOR Count <- 1 TO 20 CALL Reset(Count) NEXT Count ENDIF			
OTHERWISE : Status <- TRUE			
WHILE AllDone() = TRUE			
30 : NextChar <- 'X'			

[4]

Question 1

Pseudocode example	Selection	Iteration	Subroutine

`IF Status = FALSE THEN`
`  FOR Count ← 1 TO 20`
`  CALL Reset(Count)`
`  NEXT Count`
`ENDIF`			

`OTHERWISE : Status ← TRUE`			
`WHILE AllDone() = TRUE`			
`30 : NextChar ← 'X'`			

[4]

Question 1

Expression	Evaluates to
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) − 2, 1)`	

Question 1

Variable	Example data value	Data type
`Result`	`5.42`	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	`15/11/2009`	

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(b) Complete the table by giving the appropriate data type.

Variable	**Example data value**	**Data type**
`Result`	5.42	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	15/11/2009	

Question 1

[3]

Question 1

9618/22 N25 ▪ Q1 ▪ 11 marks

Refer to the **insert** for the list of pseudocode functions and operators.

(c) Evaluate each expression in the table by using the data values shown in (b).
Write 'ERROR' if the expression contains an error.

Expression	**Evaluates to**
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) - 2, 1)`	

Question 1

[4]

Solution 1

(a) Worked steps

Judge each extract by what its statements **do**, and remember a single extract can be more than one thing.

Extract	Selection	Iteration	Subroutine
`IF Status = FALSE THEN … FOR Count <- 1 TO 20 … CALL Reset(Count) … NEXT Count … ENDIF`	✓	✓	✓
`OTHERWISE : Status <- TRUE`	✓		
`WHILE AllDone() = TRUE`		✓	✓
`30 : NextChar <- 'X'`	✓		

Solution 1

The rows that carry the marks are the ones doing more than one thing:

- The first is all three at once: an IF, a FOR, and a CALL.
- `WHILE AllDone() = TRUE` loops **and** calls a function in its condition — a function call is a subroutine call wherever it appears.
- Both `OTHERWISE` and `30 :` are branches of a CASE statement, so both are selection.

Solution 1

Mark scheme

- Pseudocode example
- Selection
- Iteration
- Subroutine
- IF Status = FALSE THEN
- FOR Count <- 1 TO 20
- CALL Reset(Count)
- NEXT Count
- ENDIF
- Π
- Π
- Π

Solution 1

- OTHERWISE : Status <- TRUE
- Π
- WHILE AllDone() = TRUE
- Π
- Π
- 30 : NextChar <- 'X'
- Π
- One mark per row

Solution 1

(b) Worked steps

Pick the type from the **example value**, not from the variable's name.

Variable	Example value	Data type
`Result`	`5.42`	`REAL`
`MonthLetter`	`"JFMAMJJASOND"`	`STRING`
`Birthday`	`15/11/2009`	`DATE`

Solution 1

- `5.42` has a fractional part, so REAL, not INTEGER.
- `MonthLetter` sounds like a single character, but the value shown is twelve of them — STRING. This is the row the question is testing.
- A date has its own type in this syllabus.

Solution 1

Mark scheme

- Variable
- Example data value
- Data type
- Result
- 5.42
- REAL
- MonthLetter
- "JFMAMJJASOND"
- STRING
- Birthday
- 15/11/2009
- DATE

Solution 1

- One mark per row

Solution 1

(c) Mark scheme

- Expression
- Evaluates to
- INT(Result) + 1 > 6
- FALSE
- NUM_TO_STR(LENGTH(MonthLetter))
- "12"
- NUM_TO_STR(Result + "3.2")
- ERROR

Solution 1

- MID(MonthLetter, MONTH(Birthday) — 2, 1)
- "S"
- One mark per row
- 4
- October/November
- 2025

Question 2

9618/21 J25 ▪ Q2 ▪ 10 marks

A program is being developed to calculate the pay of employees working for a company.
A function `CalculateBonus()` calculates bonus pay based on the value of sales.
Bonus pay is calculated as shown in the table.

Value of sales (in dollars)	**Bonus pay (in dollars)**
below 2000	0
between 2000 and 4000 inclusive	10
above 4000	100

Question 2

A flowchart for the function `CalculateBonus()` has been designed.

(a)(i) Explain why this error occurs. **[1]**

(a)(ii) Explain how this error could be corrected. **[1]**

(b)(i) State a value that could be replaced by a constant in the function `CalculateBonus()` **[1]**

(b)(ii) Explain how the use of constants helps to reduce the risk of programming errors. **[2]**

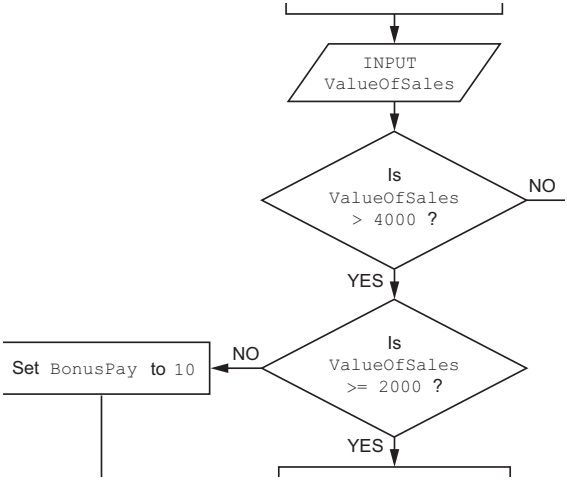
(b)(iii) One other way that can reduce the risk of errors when writing the program is the use of library routines.

Explain how library routines can reduce the risk of programming errors. **[1]**

(c) All the logic errors have been corrected, and the program has been coded.

Identify and describe **two** other types of error that the program could contain. **[4]**

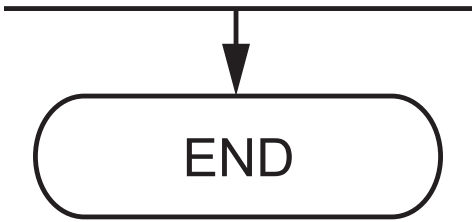
Question 2


INPUT
ValueOfSales
Is
ValueOfSales
> 4000 ?
NO
YES
Is
ValueOfSales
>= 2000 ?
NO
Set BonusPay to 10
YES

Question 2

Value of sales (in dollars)	**Bonus pay (in dollars)**
below 2000	0
between 2000 and 4000 inclusive	10
above 4000	100

Question 2



Solution 2

(a)(i)

Mark scheme

- MP1
- The two decision boxes are in the wrong order //
- The first decision symbol is wrong
- MP2
- The first decision should check for values greater than or equal to 2000
- Max 1

Solution 2

(a)(ii) Mark scheme

- Explaining how to correct the flowchart.
- MP1
- Swap around the two decision boxes
- MP2
- The first decision box should check for values greater than or equal to 2000 and the second decision box should check for values above 4000
- Max 1

Solution 2

(b)(i)

Mark scheme

- 2000 / 4000 / 10 / 100

Solution 2

(b)(ii) Mark scheme

- MP1
- The value cannot be changed // avoids being different in two places
- MP2
- Makes the program easier to understand
- MP3
- Avoids repeatedly writing the same value throughout the program
- MP4
- A change to the value requires a change in only one statement

Solution 2

- Max 2

Solution 2

(b)(iii)

Mark scheme

- MP1
- (The code is) tried and tested so error free
- MP2
- Provides functionality / code that the programmer may find difficult to code themselves
- Max 1

Solution 2

(c) Mark scheme

- MP1
- Syntax (error)
- MP2
- Rules of programming language // the language grammar was not followed
- MP3
- Run-time (error)
- MP4
- The program performs an illegal operation

Solution 2

- Mark as follows:
- One mark for naming each type of error and one mark for the description

Question 1

9618/21 N24 ▪ Q1 ▪ 9 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program will calculate the tax payable based on the cost of an item.
Calculations will occur at many places in the program and these involve the use of one of three tax rates.
Tax rate values represent a percentage. For example, a tax rate value of 5.23 represents 5.23
Tax rate values are used at several places within the program. One example is given in pseudocode as follows:
HighRate <- FALSE CASE OF ItemCost <= 50 : TaxRate <- 3.75 // tax rate of 3.75<= 200 : TaxRate <- 5.23 // tax rate of 5.23> 200 : TaxRate <- 6.25 // tax rate of 6.25HighRate <- TRUE ENDCASE TaxPayable <- ItemCost * TaxRate // tax payable

Question 1

(a) The pseudocode contains a logical error.
Identify the error **and** suggest a correction. **[2]**

Question 1

Variable name	Data type
`HighRate`	
`TaxPayable`	

Question 1

9618/21 N24 ▪ Q1 ▪ 9 marks

Refer to the **insert** for the list of pseudocode functions and operators.
A program will calculate the tax payable based on the cost of an item.
Calculations will occur at many places in the program and these involve the use of one of three tax rates.
Tax rate values represent a percentage. For example, a tax rate value of 5.23 represents 5.23
Tax rate values are used at several places within the program. One example is given in pseudocode as follows:

```
HighRate <- FALSE CASE OF ItemCost <= 50 : TaxRate <- 3.75 // tax rate of 3.75<= 200 : TaxRate <- 5.23 // tax rate of 5.23> 200 : TaxRate <- 6.25 // tax rate of 6.25HighRate <- TRUE ENDCASE TaxPayable <- ItemCost * TaxRate // tax payable
```

(b)(i) Identify a more appropriate way of representing the tax rate values in the final program. **[1]**

Question 1

(b)(ii) Describe the benefits of your answer to part (b)(i) with reference to this program. **[3]**