Past-paper walk-through

User-defined data types ▪ 13.1

eXercise

Questions in this set

- 9618/31 N25 Q1 ▪ 6 marks
- 9618/33 N25 Q1 ▪ 8 marks
- 9618/41 N25 Q2 ▪ 30 marks
- 9618/31 J25 Q1 ▪ 6 marks
- 9618/32 J25 Q1 ▪ 6 marks
- 9618/33 J25 Q1 ▪ 6 marks
- 9618/31 N24 Q9 ▪ 6 marks
- 9618/32 N24 Q3 ▪ 7 marks
- 9618/33 N24 Q6 ▪ 7 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

9618/31 N25 ▪ Q1 ▪ 6 marks

The composite record data type, `ClubMember`, is defined in pseudocode as:
TYPE ClubMember DECLARE Code : INTEGER DECLARE LastName : STRING DECLARE FirstName : STRING DECLARE Telephone : STRING DECLARE JoinDate : DATE DECLARE Fees : REAL DECLARE FeesPaid : BOOLEAN ENDTYPE

(a)(i) Write the pseudocode statement to set up a variable for one record of the composite data type, `ClubMember`. **[1]**

(a)(ii) Write the pseudocode statements to assign the following values to the variable set up in part (a)(i):

- 984632 to `Code`
- `TRUE` to `FeesPaid`

Question 1

[2]

(b)(i) Write the pseudocode statement for the type declaration of `Activity` to hold the names of the available activities:
Badminton, Football, Golf, Snooker, Swimming, Tennis. **[2]**

(b)(ii) Write the new pseudocode statement required to update the declaration of `Choice` in the definition of `ClubMember`. **[1]**

Solution 1

(a)(i)

Worked steps

```
DECLARE Member1 : ClubMember
```

A composite type is used exactly like a built-in one: `DECLARE <name> : <type>`.

Mark scheme

- DECLARE Member1 : ClubMember

Solution 1

(a)(ii) Worked steps

Reach into a record with **dot notation**, one statement per field.

```
Member1.Code <- 984632
Member1.FeesPaid <- TRUE
```

- 984632 is a number, so no quotes.
- `TRUE` is a Boolean literal, so no quotes either — `"TRUE"` would be a string and loses the mark.

Solution 1

Mark scheme

- One mark for each correct answer
- Example answer
- Member1.Code <- 984632
- Member1.FeesPaid <- TRUE

Solution 1

(b)(i) Worked steps

An enumerated type lists its allowed values in brackets, and they are identifiers rather than strings.

```
TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming, Tennis)
```

Solution 1

Mark scheme

- One mark per mark point (Max 2)
- MP1
- TYPE Activity=
- MP2
- (Badminton, Football, Golf, Snooker, Swimming, Tennis)
- Example answer
- TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming,
- Tennis)

Solution 1

(b)(ii) Worked steps

```
DECLARE Choice : Activity
```

Solution 1

The field's type becomes the new enumerated type, so `Choice` can now only hold one of those six values — which is the whole reason for declaring it.

Mark scheme

- DECLARE Choice : Activity

Question 1

9618/33 N25 ▪ Q1 ▪ 8 marks

An enumerated data type, `Spectrum`, is required to hold the names of colours.

(a)(i) Write the **pseudocode** statement for the type declaration of `Spectrum` to hold the names of the colours available:

Red, Orange, Yellow, Green, Blue, Indigo, Violet. **[2]**

(a)(ii) Identify **two** characteristics of the list of values given in an enumerated data type declaration.

1. 2. **[2]**

Question 1

Field	Example data
ColourCode	XYZ12345
Colour	Red
Wavelength	650
Frequency	4.62
PrimaryColour	Yes

Question 1

9618/33 N25 ▪ Q1 ▪ 8 marks

An enumerated data type, `Spectrum`, is required to hold the names of colours.

(b) `ColourData` is a composite data type to store definitions of colours.

Write **pseudocode** statements to declare `ColourData` to hold the following fields:

Field	**Example data**
ColourCode	XYZ12345
Colour	Red
Wavelength	650
Frequency	4.62
PrimaryColour	Yes

Question 1

Use the most appropriate data type in each case, including the enumerated data type `Spectrum` from part a(i). **[4]**

Solution 1

(a)(i) Worked steps

```
TYPE Spectrum = (Red, Orange, Yellow, Green, Blue, Indigo, Violet)
```

Solution 1

Two marks, two halves: `TYPE <name> =` and the bracketed list. The values are identifiers, so no quotes — quoting them would make them strings and lose the point of an enumerated type, which is that only these seven values are possible. **Mark scheme**

- One mark per mark point (Max 2)
- MP1
- TYPE Spectrum =
- MP2
- (Red, Orange, Yellow, Green, Blue, Indigo, Violet)
- Example answer
- TYPE Spectrum = (Red, Orange, Yellow, Green, Blue,
- Indigo, Violet)

Solution 1

(a)(ii)

Mark scheme

- One mark per mark point (Max 2)
- MP1 the list is ordered/ordinal
- MP2 the list contains all possible values
- MP3 no duplicate values in list
- MP4 all values are the same data type

Solution 1

(b) Worked steps

```
TYPE ColourData
    DECLARE Name : Spectrum
    DECLARE Wavelength : REAL
    DECLARE Hexadecimal : STRING
    DECLARE Visible : BOOLEAN
    DECLARE Discovered : DATE
ENDTYPE
```

Solution 1

The four marks are placed deliberately, and each is easy to drop on its own:

- `TYPE … ENDTYPE` around the whole thing;
- `DECLARE` on **every** field, not just the first;
- the colour field typed as `Spectrum` — the type declared in part (a), which is the whole reason that part came first;
- sensible types for the rest: a wavelength is REAL, a hex code is a STRING (it contains letters), a yes/no field is BOOLEAN and a date has its own type.

Solution 1

Mark scheme

- One mark for TYPE ColourData and ENDTYPE correct
- One mark for correct use of DECLARE in all declarations
- One mark for correct use of Spectrum in declaration
- One mark for remaining four declarations correct (STRING, INTEGER,
- REAL, BOOLEAN)
- Example answer
- TYPE ColourData
- DECLARE ColourCode : STRING
- DECLARE Colour : Spectrum
- DECLARE Wavelength : INTEGER
- DECLARE Frequency : REAL
- DECLARE PrimaryColour : BOOLEAN
- ENDTYPE

Question 2

9618/41 N25 ▪ Q2 ▪ 30 marks

A program stores data about trains and train stations using Object-Oriented Programming (OOP). The class `Train` stores the data about the trains:

Train	
`TrainIDNumber : String`	stores the train ID number
`Route : Integer`	stores the route number the train is travelling
`Constructor()`	initialises `TrainIDNumber` and `Route` to the parameter values
`GetTrainIDNumber()`	returns the train ID number
`GetRoute()`	returns the route number the train is travelling

Question 2

(a)(i) Write program code to declare the class `Train` and its constructor.
Do **not** declare the other methods.
All attributes should be private.
Use your programming language appropriate constructor.
If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_N25**.
Copy and paste the program code into part **2(a)(i)** in the evidence document.

Question 2

[4]

(a)(ii) The methods `GetTrainIDNumber()` and `GetRoute()` return the appropriate attribute.

Write program code for `GetTrainIDNumber()` and `GetRoute()`

Save your program. Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[3]

Question 2

Station	
`StationID : String`	stores the station ID
`NumberPlatforms : Integer`	stores the number of platforms at the station
`Trains[0:9] : Train`	stores the trains currently at the station platforms
`NumberTrains : Integer`	stores the number of trains currently at the station platforms
`Constructor()`	initialises `StationID` and `NumberPlatforms` to the parameter values, initialises `Trains` to an empty array and `NumberTrains` to `0`
`GetTrains()`	returns a string containing data about the trains currently at the station platforms
`AddTrain()`	takes a `Train` parameter and stores it if there is a platform available; each platform can only have one train

Question 2

Train	
`TrainIDNumber : String`	stores the train ID number
`Route : Integer`	stores the route number the train is travelling
`Constructor()`	initialises `TrainIDNumber` and `Route` to the parameter values
`GetTrainIDNumber()`	returns the train ID number
`GetRoute()`	returns the route number the train is travelling

Question 2

train ID number	route
12ADV	134
33ART	20
9FKF	3
21VBC	24