Past-paper walk-through

Program Design ▪ 12.2

eXercise

Questions in this set

- 9618/21 N25 Q7 ▪ 10 marks
- 9618/22 N25 Q7 ▪ 6 marks
- 9618/23 N25 Q7 ▪ 9 marks
- 9618/23 J25 Q6 ▪ 9 marks
- 9618/21 N24 Q7 ▪ 7 marks
- 9618/22 N24 Q7 ▪ 10 marks
- 9618/23 N24 Q7 ▪ 9 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 7

9618/21 N25 ▪ Q7 ▪ 10 marks

There are several different ways to express an algorithm during the design of a program.

(a) One part of the program contains an algorithm which is represented by a state-transition diagram.

The table shows the inputs, outputs and states for the algorithm:

Current state	Input	Output	Next state
S1	A2	X2	S3
S1	A1	X1	S2
S2	A4	X4	S5
S3	A1		S3
S3	A3	X3	S2
S3	A2	X4	S4
S4	A1	X1	S4
S4	A3		S2
S4	A4	X4	S5

Question 7

Complete the state-transition diagram to represent the information given in the table. **[5]**

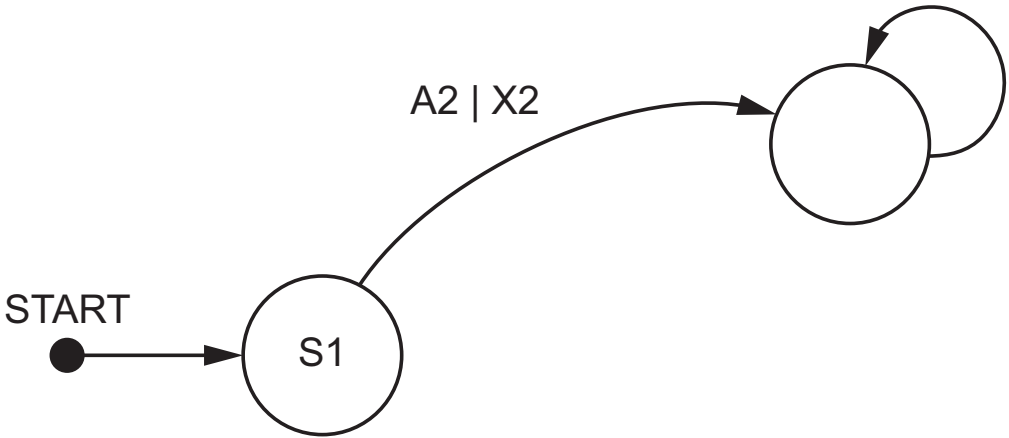
Question 7

Current state	Input	Output	Next state
S1	A2	X2	S3
S1	A1	X1	S2
S2	A4	X4	S5
S3	A1		S3
S3	A3	X3	S2
S3	A2	X4	S4
S4	A1	X1	S4
S4	A3		S2
S4	A4	X4	S5

Question 7

Pseudocode module header
`PROCEDURE Setup()`
`PROCEDURE Restart(H1 : STRING, C1 : INTEGER)`
`FUNCTION Modify(B1 : BOOLEAN) RETURNS INTEGER`
`PROCEDURE Final(T1 : INTEGER)`
`PROCEDURE Update(BYREF R2 : STRING)`
`FUNCTION Confirm() RETURNS BOOLEAN`

Question 7



Question 7

9618/21 N25 ▪ Q7 ▪ 10 marks

There are several different ways to express an algorithm during the design of a program.

(b) A structure chart is used to document a different part of the program.

This part of the program contains six modules:

Pseudocode module header
`PROCEDURE Setup()`
`PROCEDURE Restart(H1 : STRING, C1 : INTEGER)`
`FUNCTION Modify(B1 : BOOLEAN) RETURNS INTEGER`
`PROCEDURE Final(T1 : INTEGER)`
`PROCEDURE Update(BYREF R2 : STRING)`
`FUNCTION Confirm() RETURNS BOOLEAN`

Question 7

Module `Setup` will repeatedly call **three** of the modules.
Complete the structure chart to document the information given in the table. **[5]**

Solution 7

(a)

Mark scheme

One mark for each:

1. Label A1 | X1 added on line from S1 to state labelled S2
2. States S3, S4 and S5 labelled
3. Line from S2 to S5 with event label A4|X4
4. Lines from S3 with correct event labels including 'loop' label
5. Lines from S4 with correct event labels including 'loop'

Solution 7

(b) Worked steps

A structure chart says three things at once, and each is a separate mark: **who calls whom**, **what flows between them**, and **how the call repeats**.

- **Label every box** with its module name, and place them by level: the caller above, the modules it calls beneath it.
- Draw the **iteration arrow** — a curved arrow round the line of a call that happens repeatedly. A selection is marked with a diamond instead; using the wrong symbol loses the mark even when the structure is right.
- Show **parameters** as arrows down the calling line, labelled with the data passed — for example the value passed into `Restart`.
- Show **return values** as arrows up the line, labelled with what comes back — the value `Modify` returns, which is then passed on as a parameter to the next module.

Solution 7

Read the description sentence by sentence: each one is either a call, a piece of data, or a repetition, and each maps to exactly one thing on the chart.

Mark scheme

One mark per point:

1. All boxes correctly labelled
2. Iteration arrow
3. Parameter to Restart
4. Return value from Modify and parameter to Final
5. BYREF parameter to Update

Question 7

9618/22 N25 ▪ Q7 ▪ 6 marks

The structure chart shows part of a program design:

(a) Explain the meaning of the curved arrow symbol in this structure chart. **[2]**

(b) The program designer has noted that:

- Count is the number of elements in an array
- T1 is a value representing the number of elements in an array that have been modified
- Key is of type `STRING`

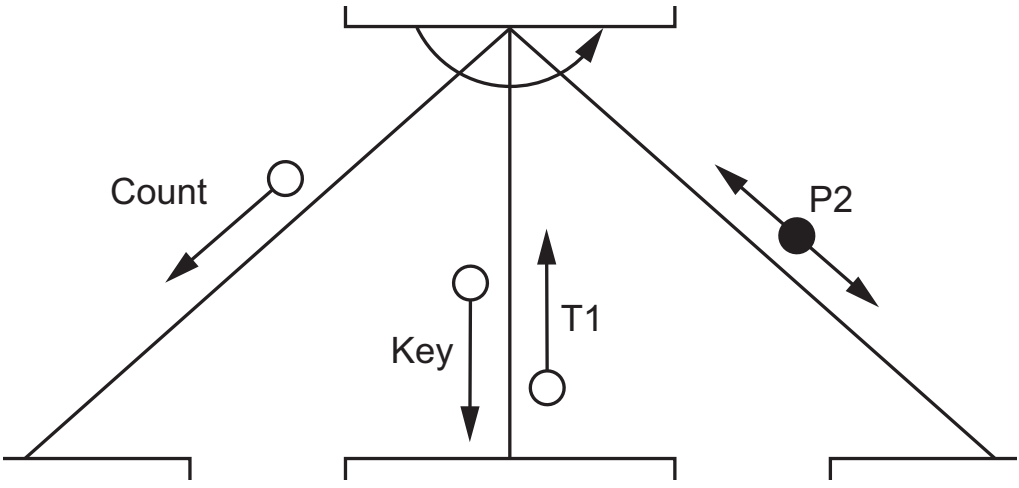
Write the pseudocode module headers for analyse, update and sort.

Analyse:

Update:

Sort: **[4]**

Question 7



Solution 7

(a)

Mark scheme

- Example explanation:
- The arrow means that Convert repeatedly calls Analyse then Update and then Sort (in that order)10
- MP1
- Reference to 'repetition'...'
- MP2
- Naming all four modules correctly and the sequence is stated or implied

Solution 7

(b) Worked steps

Every header is decided by three questions: does it return a value, what does it take, and does the caller need to see a change?

```
PROCEDURE Analyse(Count : INTEGER)
FUNCTION Update(Key : STRING) RETURNS INTEGER
PROCEDURE Sort(BYREF P2 : BOOLEAN)
```

Solution 7

- **PROCEDURE or FUNCTION** — a module that gives a value back is a FUNCTION and needs `RETURNS <type>`; one that just does something is a PROCEDURE. This is the first mark and the one most often lost.
- **Parameter names and types**, matching the description: a count is INTEGER, a key is STRING.
- **BYREF** where the module must change the caller's own variable. By default a parameter is passed by value, so the caller sees nothing; `BYREF` is the only way that change gets out, and here it is what the description asks for.

Solution 7

Mark scheme

- MP1
- PROCEDURE Analyse (Count : INTEGER)
- FUNCTION Update (Key : STRING) RETURNS INTEGER
- MP2
- MP3
- MP4
- PROCEDURE Sort (BYREF P2 : BOOLEAN)
- Mark as follows:
- One mark per underlined part
- 4
- October/November
- 2025

Question 7

9618/23 N25 ▪ Q7 ▪ 9 marks

There are several different ways to express an algorithm during the design of a program.

(a) One part of the program contains an algorithm which is represented by a state-transition diagram.

The table shows the inputs, outputs and states for the algorithm:

Current state	Input	Output	Next state
S1	A1		S2
S2	A2	X4	S3
S3	A1	X1	S3
S3	A2	X1	S3
S3	A3	X3	S4
S3	A4		S2
S4	A3	X4	S5
S4	A4	X4	S5
S4	A1		S2

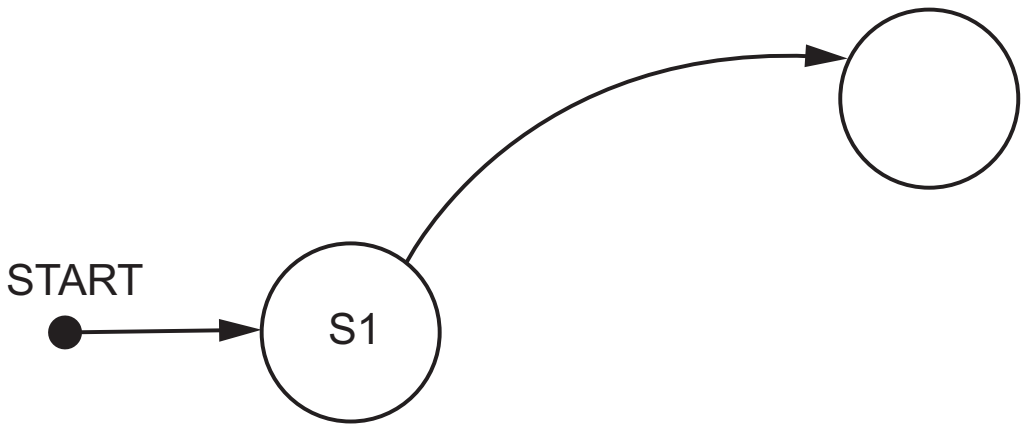
Question 7

Complete the state-transition diagram to represent the information given in the table: **[5]**

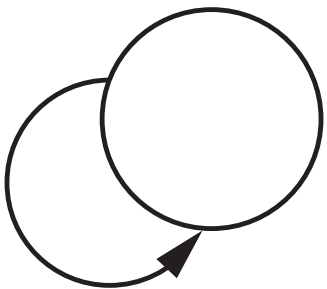
Question 7

Current state	Input	Output	Next state
S1	A1		S2
S2	A2	X4	S3
S3	A1	X1	S3
S3	A2	X1	S3
S3	A3	X3	S4
S3	A4		S2
S4	A3	X4	S5
S4	A4	X4	S5
S4	A1		S2

Question 7



Question 7



Question 7

9618/23 N25 ▪ Q7 ▪ 9 marks

There are several different ways to express an algorithm during the design of a program.

(b) A structure chart is used to document a different part of the program, made up of five modules.

Program notes:

- module `Setup` calls either module `Restart`, or module `Confirm`
- module `Confirm` takes a string as a parameter and returns an integer
- module `Modify` has no parameters and returns a Boolean
- module `Update` takes a string as a parameter
- module `Restart` repeatedly calls module `Update` followed by module `Modify`
- module `Restart` takes a string as a parameter that is passed by reference.

Draw a structure chart to represent the relationship between the **five** modules, including all parameters and return values. **[4]**

Solution 7

(a) Mark scheme

- One mark for each:
- 1
- S2 to S5 labelled
- 2
- Label A1 added to event from S1 to S2
- 3
- Event lines from S2 to S3 and S3 to S2 correctly labelled
- 4

Solution 7

- Events lines from S3 to S3 and event line from S3 to S4 all labelled correctly
- 5
- Event lines from S4 to S5 correctly and event line from S4 to S2 labelled correctly
- Max 4 for any additional events/lines
- 5
- October/November
- 2025

Solution 7

(b) Worked steps

A structure chart is read top-down, and the notes give you it one line at a time. Take them in order and turn each into a drawing rule.

The hierarchy. `Setup` is the only module nothing calls, so it is the root. It calls `Restart` and `Confirm`, and `Restart` calls `Update` and `Modify` — so the chart is three levels: `Setup`; then `Restart` and `Confirm`; then `Update` and `Modify` beneath `Restart`.

The arrows. Every parameter and return value is a labelled arrow on the connecting line, pointing the way the data travels.

- `Confirm` takes a string and returns an integer: an arrow **down** labelled with the string and one **up** labelled with the integer.
- `Update` takes a string: one arrow **down**.

Solution 7

- `Modify` has no parameters and returns a Boolean: one arrow **up**, and no arrow down.
- `Restart` takes a string: one arrow **down** from `Setup`.

The two annotations that carry their own marks.

- `Setup` calls *either* `Restart` *or* `Confirm` — that is **selection**, drawn as a diamond on the line above the two branches.
- `Restart` calls `Update` **followed by** `Modify`, repeatedly — that is **iteration**, drawn as a curved arrow round the lines to both of them, with `Update` to the left of `Modify` so the left-to-right order shows the sequence.

Solution 7

The four marks are for the hierarchy, the parameters, the return values, and the two control annotations. Neat boxes with no arrows score one of four, so label every line — and remember that a module with no parameters still gets its return arrow. **Mark scheme**

- One mark for each:
- 1
- All boxes correctly labelled and connected in correct hierarchy
- 2
- BYREF parameter to Restart, parameter to Confirm and return
- 3

Solution 7

- Diamond and loop symbol
- 4
- Parameter to Update and return from Modify
- 4
- October/November
- 2025

Question 6

9618/23 J25 ▪ Q6 ▪ 9 marks

Study the structure chart:
Some of the parameter data types are:

- R and V are of type `INTEGER`
- T is of type `REAL`
- U is of type `STRING`
- W is of type `BOOLEAN`.

Some of the modules in the structure chart are functions, others are procedures.

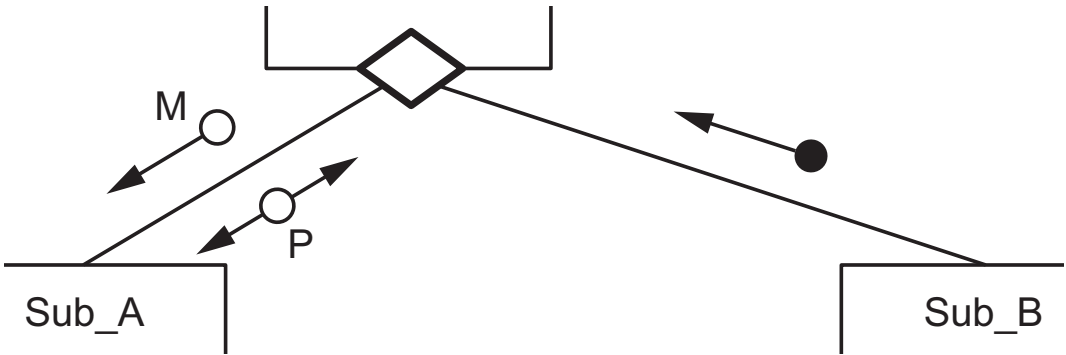
(a) Explain **one** difference between functions and procedures. **[1]**

Question 6

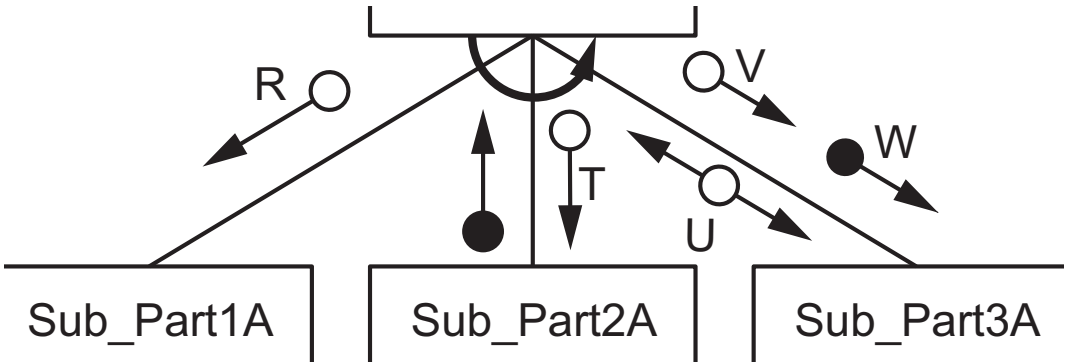
Symbol	Explanation
	....
	....

[3]

Question 6



Question 6



Question 6

9618/23 J25 ▪ Q6 ▪ 9 marks

Study the structure chart:
Some of the parameter data types are:

- R and V are of type `INTEGER`
- T is of type `REAL`
- U is of type `STRING`
- W is of type `BOOLEAN`.

Some of the modules in the structure chart are functions, others are procedures.

(b) Write the pseudocode to define the module headers:
Sub_Part1A
Sub_Part2A
Sub_Part3A **[5]**

Question 6

9618/23 J25 ▪ Q6 ▪ 9 marks

Study the structure chart:
Some of the parameter data types are:

- R and V are of type `INTEGER`
- T is of type `REAL`
- U is of type `STRING`
- W is of type `BOOLEAN`.

Some of the modules in the structure chart are functions, others are procedures.

Question 6

(c) The structure chart uses the two symbols shown.
Complete the table to explain what each symbol means and how the relevant modules in the structure chart are affected.

Symbol	Explanation
	
	

Question 6

[3]

Solution 6

(a)

Mark scheme

- MP1
- A function returns a (single) value and a procedure does not
- MP2
- A procedure can pass parameters by ByRef
- Max 1

Solution 6

(b) Worked steps

A structure chart's arrows **are** the parameter list. Read each module's arrows and turn them into a header, in three steps:

1. **Procedure or function?** An arrow pointing **up** out of the module is a return value, so it is a `FUNCTION … RETURNS <type>`. No upward arrow means a `PROCEDURE`.
2. **Which parameters?** Every arrow pointing **down** into the module is one parameter, in the order they appear.
3. **BYREF or by value?** A parameter that also has to be *changed* for the caller — an arrow both ways on the same identifier — is passed `BYREF`.

Solution 6

Then take each parameter's type from the list the question gives.

```
PROCEDURE Sub_Part1A(R : INTEGER)

FUNCTION  Sub_Part2A(T : REAL) RETURNS BOOLEAN

PROCEDURE Sub_Part3A(V : INTEGER, W : BOOLEAN, BYREF U : STRING)
```

Five marks in total, weighted towards the harder two: one for the first, two each for the second and third.

The two that decide most of them:

Solution 6

- **`RETURNS` versus `BYREF`.** `Sub_Part2A` sends a value back as its *result*, so it is a function. `Sub_Part3A` sends `U` back through its *parameter*, so `U` is BYREF and the module returns nothing. Both are "the caller gets something back", and the chart distinguishes them by whether the returning arrow is on a parameter line.
- **Parameter order.** Take it from the chart left to right; the header is only correct if the caller can match it.

Solution 6

Every parameter needs its type, and a function needs its return type. A header without types is not a pseudocode declaration. **Mark scheme**

- Sub_Part1A:
- PROCEDURE Sub_Part1A(R: INTEGER)
- 1 Mark
- Sub_Part2A:
- FUNCTION Sub_Part2A(T : REAL) RETURNS BOOLEAN
- 2 Marks
- Sub_Part3A:
- PROCEDURE Sub_Part3A(V: INTEGER, W: BOOLEAN, BYREF U :
- STRING)
- 2 Marks

Solution 6

(c) Mark scheme

- Symbol
- Explanation
- The module Main calls either Sub_A or Sub_B (which one is called is determined at run time)
- The module Sub_A will repeatedly call Sub_Part1A followed by Sub_Part2A then by Sub_Part3A
- MP1 Shape1: reference to selection
- MP2 Shape2: reference to repetition // iteration

Solution 6

- MP3
- Descption ($\times 2$) including correct use of all module names in both parts

Question 7

9618/21 N24 ▪ Q7 ▪ 7 marks

A program contains six modules with headers as follows:

Pseudocode module header
`PROCEDURE Connect()`
`FUNCTION Activate(H1 : STRING, Code : INTEGER) RETURNS BOOLEAN`
`FUNCTION Sync(T1 : BOOLEAN, S2 : REAL) RETURNS INTEGER`
`PROCEDURE Initialise(BYREF ID : INTEGER, BYVAL CC : INTEGER)`
`FUNCTION Reset(RA : STRING) RETURNS INTEGER`
`FUNCTION Enable(SA : INTEGER) RETURNS BOOLEAN`

Question 7

Module `Connect()` will call either `Activate()` or `Sync()`. This is decided at run-time.

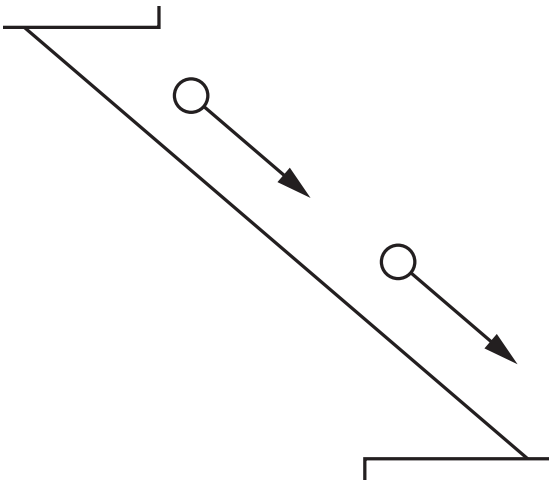
(a) Complete the structure chart for these modules. **[5]**

(b) Explain the meaning of the curved arrow symbol used in the diagram in part (a). **[2]**

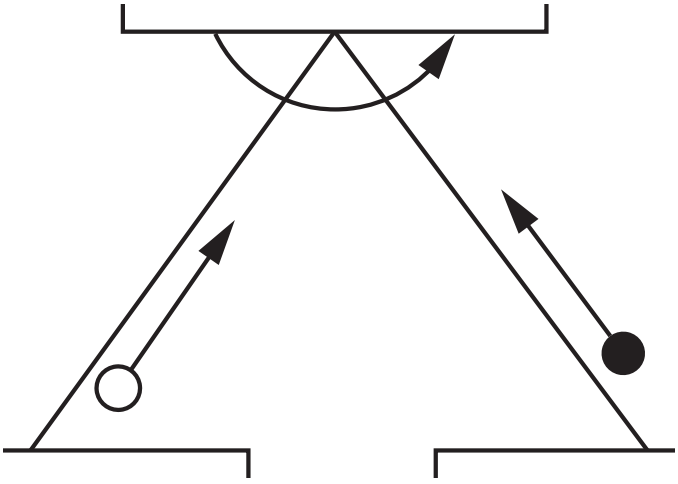
Question 7

Pseudocode module header
`PROCEDURE Connect()`
`FUNCTION Activate(H1 : STRING, Code : INTEGER) RETURNS BOOLEAN`
`FUNCTION Sync(T1 : BOOLEAN, S2 : REAL) RETURNS INTEGER`
`PROCEDURE Initialise(BYREF ID : INTEGER, BYVAL CC : INTEGER)`
`FUNCTION Reset(RA : STRING) RETURNS INTEGER`
`FUNCTION Enable(SA : INTEGER) RETURNS BOOLEAN`

Question 7



Question 7



Solution 7

(a) Mark scheme

- One mark per point:
- 1
- All module identified in correct hierarchy
- 2
- Parameters to both Sync and Activate and return values
- 3
- Parameters to both Reset and Enable
- 4

Solution 7

- Parameters to Initialise
- 5
- Decision diamond

Solution 7

(b) Mark scheme

- Explanation:
- Means that Sync repeatedly calls Reset and (then) Enable
- One mark for each point:
- MP1: reference to iteration
- MP2: naming all three modules correctly including correct sequence of calls
- 2
- October/November
- 2024

Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.
A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

Question 7

(a) Identify **three** items of customer information that will be used by the new module **and** justify your choices.

Item 1

Item 2

Item 3 **[3]**

Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.

A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

(b) Identify the computational thinking skill that you needed to use to answer part (a). **[1]**

Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.
A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

(c) It is decided to adopt a formal program development life cycle model for the development of the new module.

Question 7

The analysis of the new module is complete and the project moves on to the design stage. During this stage all the necessary algorithms and module designs will be defined.

State **three other** items that will be defined for the new module during the design stage.

1. 2. 3. **[3]**

Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.

A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

(d) Part of the coffee shop program contains three program modules as follows:

- Module `Init()` has no parameters and returns a Boolean.

Question 7

- Module `Reset()` takes a string as a parameter and returns an Integer.
- Module `Check()` repeatedly calls `Init()` followed by `Reset()`.

Draw a structure chart to represent the relationship between the **three** modules, including all parameters and return values. **[3]**

Solution 7

(a) Mark scheme

- ▪ Customer ID – to reference the other customer details
- ▪ Email address – to send the email
- ▪ Name – to personalise the email
- ▪ Date of last visit – to select which customers should receive an email
- ▪ Unique voucher code (or method of code generation) – to include in the email
- Mark as follows:
- One mark per item and justification
- Max 3 marks

Solution 7

(b)

Mark scheme

- Abstraction

Solution 7

(c) Worked steps

The question already names algorithms and module designs, so those two are off the table — "three **other** items" means three more. Think about what a programmer would still be unable to start coding without.

- **Data structures** — the arrays, records and files the program will use. Equivalently a **data dictionary** or **identifier table**, and the **validation rules** each field must satisfy.
- **Data-flow diagrams** or **state-transition diagrams** — how data moves between the parts of the system, and how the system changes state.
- The **user interface** — the screen layouts, and here also the **format of the email** the module sends.

Solution 7

One mark each. The trap is naming things from the wrong stage: a requirement specification belongs to **analysis**, and test data and test plans belong to **testing**. Both are common answers and neither scores.

A useful test before writing an item down: could a programmer produce this *before* knowing what the system must do (too early, that is analysis), or only *after* the code exists (too late, that is testing)? If neither, it is a design item. **Mark scheme**

- MP1 Data structures // data dictionary // identifier table(s) // validation rules
- MP2 Data-flow diagram // state-transition diagram
- MP3 User interface // Format for the email
- MP4 Testing method / Test plan / Test data / Trace tables

Solution 7

- MP5 Choice of email protocol to be used // Programming language to be used // Development environment
- MP6 Use of library routines // program to send the email
- Max 3 marks

Solution 7

(d) Worked steps

Turn each line of the specification into a feature of the chart, in order.
The hierarchy. `Check()` calls the other two, so it is the root, with `Init()` and `Reset()` below it. Nothing calls `Check()`, and neither of the others calls anything.
The arrows. Every parameter and return value is a labelled arrow on the line joining a module to its caller, pointing the way the data travels.

- `Init()` has no parameters and returns a Boolean: one arrow **up**, labelled Boolean, and no arrow down.
- `Reset()` takes a string and returns an integer: an arrow **down** labelled with the string and one **up** labelled with the integer.

Solution 7

The annotation. `Check()` calls them **repeatedly**, one after the other — that is **iteration**, drawn as a curved arrow looping round the lines to both modules. Put `Init()` to the left of `Reset()`, because in a structure chart left-to-right order is the order of execution and the specification says `Init()` comes first.
Three marks, three features: the hierarchy, the parameters and return values, and the iteration. Boxes with no arrows score one of the three, and a module with no parameters still needs its return arrow drawn. **Mark scheme**

- MP1
- Three boxes correctly labelled and correct hierarchy
- MP2

Solution 7

- Parameter and return values
- MP3
- Iteration arrow
- 3
- October/November
- 2024

Question 7

9618/23 N24 ▪ Q7 ▪ 9 marks

7 A coffee shop owner wants to introduce a computerised loyalty card system.

A programmer discusses the details of the system with the shop owner.

(a) Identify the stage of the program development life cycle that this discussion is part of **and** give a document that will be produced during this stage.

Question 7

Question 7

Question 7

[2]

(b) The shop will give each customer a loyalty card that displays a unique customer ID as a bar code. A customer will be able to present their card each time they make a purchase. The system will scan the bar code, calculate points, and add them to the customer's total. When the customer next makes a purchase and presents their card, they will have the option to exchange points for a discount.

The designer decides that this activity will be handled by a new module. Decomposition will be used to break the problem of designing the new module down into sub-problems (sub-modules).

Identify **four** sub-modules that could be used in the design of the new module **and** describe their use.

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

Question 7

[4]

Solution 7

(a) Worked steps

The programmer is **finding out what the system must do** by talking to the person who wants it. That is **analysis** — the first stage, before any design decision is made. One mark for the stage, one for a document it produces:

- A **requirement specification**
- A **definition of system objectives**
- A **list of problems with the existing system**
- **Survey results**
- A **feasibility study**

Solution 7

The distinction being tested is between analysis and design, and the test is simple: analysis asks **what** the system must do, design decides **how** it will do it. A discussion with the shop owner about how the loyalty scheme should work is entirely the first. A test plan and a program specification are the usual wrong answers. Both come later, once the requirements exist. **Mark scheme**

- One mark per answer:
- MP1 Analysis
- MP2 Named document, examples include:
- ▪
- A requirement specification

Solution 7

- ▪
- Definition of System Objectives
- ▪
- List of problems with existing system
- ▪
- Survey results
- ▪
- Feasibility study
- ▪
- Interview notes

Solution 7

- Max 2 marks
- 2
- 9618/23
- October/November
- 2024

Solution 7

(b) Worked steps

Decomposition means breaking the task into sub-modules that each do **one identifiable job**, and each mark is for a **name plus its use** — so give both halves. Sub-modules that fall naturally out of a loyalty-card transaction:

- `ScanCard()` or `GetID()` — read the barcode and obtain the customer's ID.
- `GetPoints()` — look up how many points that customer currently has.
- `CalculatePoints()` — work out the points earned by this purchase.
- `ExchangePoints()` or `UpdateCard()` — reduce the stored points when they are spent.
- `CalculateDiscount()` or `GetDiscount()` — work out the discount those points buy.

Solution 7

Two habits get the marks. **Name each module as a verb phrase** — a module does something, and a name like `Card()` says nothing about what. And make each one **a single job**: a `DoEverything()` that scans, calculates and updates is not a decomposition, and its use cannot be stated in one line.
A quick check on a proposed set: could each module be written and tested on its own, by somebody who knows only its name and its parameters? If not, it has not been decomposed far enough. **Mark scheme**

- One mark for name and use
- Mark as follows:
- Examples include:

Solution 7

- Sub-module: ScanCard()/ GetID()
- Use: Read the barcode from the loyalty card / Get the customer ID from the
- barcode
- Sub-module: GetPoints()
- Use: Get the number of points the customer has
- Sub-module: ExchangePoints() / UpdateCard()
- Use: Reduce the number of loyalty points
- Sub-module: GetDiscount() / CalculateDiscount()
- Use: Calculate the discount
- Sub-module: CalculatePoints()

Solution 7

- Use: Calculate points (following a purchase)
- Sub-module: UpdatePoints()
- Use: Update total points a customer has (following a purchase)
- Sub-module: ShowDiscount()
- Use: Display the discount
- Max 4 marks
- 4
- 9618/23
- October/November
- 2024