Higher
Coursework
Assessment Task

# Higher Computing Science
# Assignment
# Assessment task

This document provides information for teachers and lecturers about the coursework component of this course in terms of the skills, knowledge and understanding that are assessed. It must be read in conjunction with the course specification.

**Valid for session 2024-25 only.**

**This assessment is given to centres in strictest confidence. You must keep it in a secure place until it is used.**

This edition: January 2025 (version 1.1)

© Scottish Qualifications Authority 2025

# Contents

# Introduction

This document contains instructions for teachers and lecturers, and instructions for candidates for the Higher Computing Science assignment. You must read it in conjunction with the course specification.

This assignment has 40 marks out of a total of 120 marks available for the course assessment.

This is one of two course assessment components. The other component is a question paper.

# Instructions for teachers and lecturers

This assessment applies to the assignment for Higher Computing Science for the academic session 2024–25.

The task is valid for 2024-25 only. Once complete, you must send the assignment responses to SQA to be marked.

You must conduct the assignment under a high degree of supervision and control. This means:

- candidates must be supervised throughout the session(s)
- candidates must not have access to email or mobile phones
- candidates must complete their work independently — no group work is permitted
- candidates must not interact with each other
- with no interruption for targeted learning and teaching
- in a classroom environment

You can use any integrated development environments (IDE) that enable candidates to generate evidence — this includes online IDEs. However, the IDE must have a facility that prevents candidates accessing their files and tasks outside the supervised classroom environment.

## Time

Candidates have 6 hours to carry out the assignment, starting at an appropriate point in the course, after all content has been delivered. It is not anticipated that this is a continuous 6-hour session, although it can be, but conducted over several shorter sessions. This is at your discretion.

You have a responsibility to manage candidates' work, distributing it at the beginning and collecting it in at the end of each session, and storing it securely in between. This activity does not count towards the total time permitted for candidates to complete the assignment.

Candidates are prompted to print their work at appropriate stages of the tasks. They can print on an ongoing basis or save their work and print it later. Whatever approach they take, time for printing is not part of the 6 hours permitted for the assignment.

## Resources

Each candidate must have access to a computer system with a high-level (textual) programming language and **either**:

- database application and software that can create, edit and run SQL
- software that can create, edit and run HMTL, CSS and JavaScript

This is an open-book assessment. Candidates can access resources such as programming manuals, class notes, textbooks and programs they have written throughout the course. These may be online resources.

You must not create learning and teaching tasks that make use of constructs required in the assessment task, **with the specific purpose of developing a solution that candidates can access during the assignment.**

You can provide candidates with templates, however these templates must only contain general starter code used in learning and teaching (for example, a web page that contains the HTML, title and body elements) — templates must not be tailored to this year's task.

There may be instances where restriction of network use is prohibited (for example, a local authority-managed network with specific limitations). However, it remains your professional responsibility to make every effort to meet the assessment conditions.

## Reasonable assistance

The assignment consists of three independent tasks. They are designed in a way that does not require you to provide support to candidates, other than to ensure that they have access to the necessary resources. Candidates can complete the tasks in any order.

Once the assignment is complete, you must not return it to the candidate for further work to improve their mark. You must not provide feedback to candidates or offer an opinion on the perceived quality or completeness of the assignment response, at any stage.

You can provide reasonable assistance to support candidates with the following aspects of their assignments:

- printing, collating and labelling their evidence to ensure it is in the format specified by SQA
- ensuring candidates have all the materials and equipment required to complete the assignment — this includes any files provided by SQA
- ensuring candidates understand the conditions of assessment and any administrative arrangements around the submission and storage of evidence, and the provision of files
- technical support

## Evidence

All candidate evidence (whether handwritten or created electronically) must be submitted to SQA in a paper-based format. The evidence checklist details all evidence to be gathered. You can use it to ensure you submit all evidence to SQA.

You should advise candidates that evidence, especially code, must be clear and legible. This is particularly important when pasting screenshots into a document.

There is no need for evidence to be printed single-sided or in colour.

If evidence is handwritten, candidates must use a blue or black pen.

When packaging, ensure that:

- each assignment is accompanied by an SQA A4 flyleaf
- all completed task sheets and additional evidence of screenshots or printouts from development environments are included and ordered in line with the task
- where possible, sheets that do not contain candidate evidence are removed
- candidates' SCNs are on every page
- the flyleaf and candidate evidence are stapled together in the top left corner

## Alteration or adaptation

The tasks are in PDF and Word formats. Each task is available as a separate file from the secure site. Word files allow candidates to word process their responses to parts of the task.

You must not adapt the assignment in any way that changes the instructions to the candidate and/or the nature and content of the tasks. However, you can make changes to font size, type and colour and to the size of diagrams for candidates with different assessment needs, for example, visual impairment.

If you are concerned that any particular adaptation changes the nature and/or the content of the task, please contact our Assessment Arrangements Team for advice as soon as possible at aarequests@sqa.org.uk.

## Submission

Each page for submission has the number of the assignment task that it refers to, for example 1a, and contains space for candidates to complete their candidate number. Any other pages submitted, for example, prints of program listings or screenshots, must have this information added to them.

## Specific instructions for teachers and lecturers: 2024–25

All candidates must complete task 1 (software design and development) and **either** task 2 (database design and development) **or** task 3 (web design and development).

It is at your discretion how you approach this optionality in assessment. The task your candidates complete might be pre-determined by your progress through the course, or you may be able to let candidates choose which task to complete.

You must follow these specific instructions and ensure that candidates are aware of what you will give them at each stage in the assessment.

Print each task on single-sided paper, where applicable:

- this allows candidates to refer to information on other pages
- this helps you manage tasks that are split into more than one part

**Task 1 — part A** requires candidates to analyse a problem. They must submit their evidence to you before you issue part B.

**Task 1 — part B** is a separate section. This ensures that candidates do not access part A and change their responses. Part B requires candidates to complete the data flow design. They must submit their evidence to you before you issue part C.

**Task 1 — part C** is a separate section. This ensures that candidates do not access part B and change their responses.

Candidates must still have access to the problem description during parts B and C.

A text file (orders.txt) is provided for candidates to use in part C. You must not convert the text file into a different format. Candidates are assessed on their ability to implement a solution to the given task, using the specific file type provided.

**Task 2 — part A** requires candidates to analyse a problem. They must submit their evidence to you before you issue part B.

**Task 2 — part B** is a separate section. This ensures that candidates do not access part B and change their responses.

A Microsoft Access file (norasComics.accdb) is provided for candidates to use in part B. If your centre uses a different database management system, you can create the relational database using the CSV files or the text files provided.

If using the CSV files, you should set up all tables and fields as shown below. The CSV files contain the data for each table.

The text files contain SQL create and insert statements for each table.

In both cases, you will have to add primary keys and foreign keys as specified below. The field ‘mainCharacter’ in the ComicCharacter table is a Boolean. If required, you should set this up as an integer with ‘0’ or ‘1’.

| **Publisher** | **Comic** | **Series** | **ComicCharacter** | **CharacterInfo** |
|---|---|---|---|---|
| publisherID | comicID | seriesID | comicCharacterID | characterID |
| publisherName | comicTitle | seriesName | comicID* | characterName |
| foundedYear | valuation | startYear | characterID* | alias |
| headquarters | genre | endYear | mainCharacter | creator |
| | issue | | | |
| | publicationDate | | | |
| | publisherID* | | | |
| | seriesID* | | | |

**Task 2e —** requires candidates to test an SQL statement. You must provide this to candidates as part of the database. The SQL statement is already included in the MS Access file. If you use a different database management system, you should use the supplied text file ‘Query for 2e.txt’ to add it to the database you provide to candidates.

**Task 3 — part A** requires candidates to analyse a website. They must submit their evidence to you before you issue part B.

**Task 3 — part B** is a separate section. This ensures that candidates do not access part A and change their responses.

A folder named ‘Web files’ is provided. This contains the CSS, HTML and image files candidates need to complete this task. These files must not be renamed, and they must remain in the folders provided. However, the case of suffixes may be changed if the environment you work in requires them to be lower or upper case.

Candidates do not need to print completed web pages in colour.

# Instructions for candidates

This assessment applies to the assignment for Higher Computing Science.

This assignment has 40 marks out of a total of 120 marks available for the course assessment.

It assesses the following skills, knowledge and understanding:

- applying aspects of computational thinking across a range of contexts
- analysing problems within computing science across a range of contemporary contexts
- designing, implementing, testing and evaluating digital solutions (including computer programs) to problems across a range of contemporary contexts
- demonstrating skills in computer programming
- applying computing science concepts and techniques to create solutions across a range of contexts

Your teacher or lecturer will let you know if there are any specific conditions for doing this assessment.

In this assessment, you have to complete two short practical tasks.

You must complete task 1 (software design and development) and **either** task 2 (database design and development) **or** task 3 (web design and development).

You may complete the tasks in any order.

## Advice on how to plan your time

You have 6 hours to complete the assignment. Marks are allocated as follows:

| | | |
|---|---|---|
| ♦ Task 1 — software design and development | 25 marks | (63% of total) |
| **AND EITHER** | | |
| ♦ Task 2 — database design and development | 15 marks | (37% of total) |
| **OR** | | |
| ♦ Task 3 — web design and development | 15 marks | (37% of total) |

You can use this split as a guide when planning your time for each of the two tasks.

## Advice on gathering evidence

As you complete each task, you must gather evidence as instructed.

Your evidence, especially code, must be clear and legible. This is particularly important when you paste screenshots into a document. You can print code from the software environment or copy and paste this into other packages such as notepad or Word.

Use the evidence checklist provided to make sure you submit everything necessary at the end of the assignment. Make sure you include your candidate number on all your evidence.

Evidence may take the form of printouts of code, screenshots, typed answers, handwritten answers or drawings of diagrams and designs.

You must use a blue or black pen for any handwritten answers.

## Advice on assistance

This is an open-book assessment. This means that you can use:

- any classroom resource as a form of reference (for example, programming manuals, class notes and textbooks) — these may be online resources
- any files you have previously created throughout the course

The tasks are designed so you can complete them independently, without any support from your teacher or lecturer. This means that you:

- cannot ask how to complete any of the tasks
- cannot access any assignment files outside the classroom

# Computing Science assessment task: evidence checklist

You should complete the checklist for task 1 and **either** task 2 **or** task 3.

## Task 1 — software design and development

| Task | Evidence | Tick |
|---|---|---|
| 1a | Completed task sheet identifying the missing functional requirements | |
| 1b | Completed task sheet showing completed data flow design | |
| 1c | Printout of your completed program code, program output and ‘winningCustomer.txt’ file | |
| 1d | Completed task sheet showing completed trace table | |
| 1e | Completed task sheet evaluating efficiency and maintainability | |

## Task 2 — database design and development

| Task | Evidence | Tick |
|---|---|---|
| 2a | Completed task sheet showing the missing functional requirements | |
| 2b | Completed task sheet explaining why the entity was included | |
| 2c | Printout of the implemented SQL statement(s) | |
| | Printout of the output produced | |
| 2d | Printout of the implemented SQL statement | |
| | Printout of the output produced | |
| 2e | Printout of the amended SQL statement | |
| | Printout of the output produced | |
| 2f(i) | Completed task sheet identifying the functional requirement that could not be met | |
| 2f(ii) | Completed task sheet explaining what additional data would be required | |

## Task 3 — web design and development

| Task | Evidence | Tick |
|---|---|---|
| 3a | Completed task sheet showing the missing functional requirements | |
| 3b | Printout of the edited ‘prices.html’ and ‘style.css’ files | |
| 3c | Printout of the edited ‘scorecard.html’ file | |
| 3d | Completed task sheet describing how to comprehensively test form elements | |
| 3e | Completed task sheet stating why website is not fit for purpose | |

Please follow the steps below before handing your evidence to your teacher or lecturer.

- Check you have completed all parts of task 1 and either task 2 or task 3.
- Label any printouts and screenshots with the task number (for example 1c, 2a).
- Clearly display your candidate number on each printout.

# Task 1: software design and development

## Problem description

A fast-food outlet uses an app to allow users to order food online. The data collected by the app is stored in a text file. The owner of the fast-food outlet wants a program to analyse and use this data.

## Purpose

The program should:

- display the total number of orders delivered and the total number of orders collected
- find the first customer who has submitted a 5-star rating for a given month

The text file stores:

- order number
- date of order
- customer email address
- delivery or collection option
- order cost
- star rating from 1 (lowest) to 5 (highest)

The data from the text file will be read into an array of records.

The program will display the total number of orders delivered and the total number of orders collected to date.

A month will be entered into the program to find the first customer to give a 5-star rating for that month. This customer will win a free meal. Details of the winning customer will be written to a text file called 'winningCustomer.txt'. If there is no winning customer, a suitable message will be written to the file.

## Assumptions

- A new text file of orders will be created for each year.
- Data will be exported from the app to the text file on a regular basis.
- The data in the text file is in chronological order.
- The data is formatted correctly and is error-free.

A text file 'orders.txt' is used when the program is being created. This file contains data for the 505 orders that were made in October, November and December 2024. A sample of this data is shown below.

```
Ord1,01-Oct-24,jacobjoseph@lukewarmmail.com,Delivery,14.45,4
Ord2,01-Oct-24,arthurian@geemail.com,Collection,34.24,1
Ord3,01-Oct-24,rebeccashearer@wahoo.co.uk,Collection,17.13,3
...
Ord505,31-Dec-24,buddy@lol.com,Collection,34.67,1
```

When the month October is entered, the expected output is shown below.

```
Enter the first three letters of the month to search: Oct
Total number of orders delivered to date: 290
Total number of orders collected to date: 215
```

The expected content of the file, ‘winningCustomer.txt’ is shown below.

```
Ord9,maxpower11@inlook.com,19.34
```

## Task 1: software design and development (part A)

1a Using the problem description, identify the missing functional requirements of the program.

**(3 marks)**

| |
|---|
| **Input(s)**<br><br>♦ Read the order number, date of order, customer email, delivery or collection option, order cost and customer star rating from file to an array of records. |
| **Process(es)** |
| **Output(s)**<br><br>♦ Display the total number of orders delivered and the total number of orders collected.<br>♦ Write details of the winning customer, or 'no winner' message, to a text file. |

♦ Check your answers carefully, as you cannot return to part A after you hand it in.

♦ When you are ready, hand part A to your teacher or lecturer and collect part B.

Candidate number______________________________

# Task 1: software design and development (part B)

1b The top-level design for the program, with missing data flow, is shown below.

Complete the missing data flow in the table below. **(2 marks)**

| | | | |
|---|---|---|---|
| 1 | Read from file into array of records. | IN | |
| | | OUT | orders(orderNum,date,email,option,cost,rating) |
| 2 | Find the position of the customer who gave the first 5-star rating in a given month. | IN | |
| | | OUT | position |
| 3 | Write details of the winning customer, or 'no winner' message, to a text file. | IN | orders(orderNum,date,email,option,cost,rating) |
| | | OUT | |
| 4 | Display the total number of orders delivered and the total number of orders collected. | IN | orders(orderNum,date,email,option,cost,rating) |
| | | OUT | |

- Check your answers carefully, as you cannot return to part B after you hand it in.
- When you are ready, hand part A to your teacher or lecturer and collect part C.

Candidate number______________________________

# Task 1: software design and development (part C)

Your teacher or lecturer will provide you with a file called ‘orders.txt’.

The design of the program is shown below.

**Program top-level design (pseudocode)**

| | | | |
|---|---|---|---|
| 1 | Read from file into array of records. | OUT | orders(orderNum,date,email,option,cost,rating) |
| 2 | Find the position of the customer who gave the first 5-star rating in a given month. | IN<br><br>OUT | orders(orderNum,date,email,option,cost,rating)<br><br>position |
| 3 | Write details of the winning customer, or ‘no winner’ message, to a text file. | IN | orders(orderNum,date,email,option,cost,rating), position |
| 4 | Display the total number of orders delivered and the total number of orders collected. | IN | orders(orderNum,date,email,option,cost,rating) |

**Refinements**

```
2.1     Set position to -1
2.2     Set index to 0
2.3     Ask user to enter month to search for
2.4     While position is -1 and index is less than the length of the array
2.5         If current month is equal to searched month and current rating is 5 then
2.6             Set position to index
2.7         End if
2.8         Add 1 to index
2.9     End while
2.10    Return position

3.1     Open new file ‘winningCustomer.txt’
3.2         If position is 0 or above then
3.3             Write winning order number, email and cost to ‘winningCustomer.txt’
3.4         Else
3.5             Write ‘No winner’ to ‘winningCustomer.txt’
3.6         End if
3.7     Close ‘winningCustomer.txt’

4.1     Call countOption function to return the number of orders delivered
4.2     Call countOption function to return the number of orders collected
4.3     Output the total number of orders delivered
4.4     Output the total number of orders collected
```

**1c** Using the problem description and design, implement the program in a language of your choice.

Your program should:

- read data from the 'orders.txt' file and store into an array of records
- use procedures to:
  - write details of winner or 'no winner' message to file
  - display the totals
- use a single function to find and return the total number of orders delivered and the total number of orders collected
- use a function to find and return the position of the winning customer
- be maintainable and modular
- be tested using the month of 'Oct' as the input. The expected output is shown in the problem description.

**(15 marks)**

Print evidence of your:
- program code
- program output from your test run
- 'winningCustomer.txt' file

**1d** The function ‘Find the position of the customer who gave the first 5-star rating in a given month’ is tested using the sample data below.

```
Ord165,31-Oct-24,scr83@lol.com,Collection,19.99,2
Ord166,31-Oct-24,itsjustjen@inlook.com,Delivery,24.00,4
Ord167,01-Nov-24,arthurian@geemail.com,Delivery,38.18,4
Ord168,01-Nov-24,thetribalchief@lol.com,Collection,12.49,5
Ord169,01-Nov-24,scr83@lol.com,Delivery,55.55,3
Ord170,01-Nov-24,duckguy@male.com,Collection,44.89,5
```

Complete the trace table below to show the values up to the end of the iteration of the conditional loop.

**(2 marks)**

| | |
|---|---|
| **Month searched** | **Nov** |

| **orderNum** | **position** | **If current month is equal to searched month and current rating is 5** |
|---|---|---|
| Ord165 | -1 | False |
| Ord166 | | |
| Ord167 | | |
| Ord168 | | |

Candidate number______________________________

1e With reference to your own program code, evaluate the following.

| |
|---|
| The efficiency of your program with reference to the use of the ‘countOption’ function. **(1 mark)** |
| The maintainability of the first subprogram ‘Read from file into array of records’ if the file now contained data for a whole year from January 2025 to December 2025. Your answer should refer to data structures and loops. **(2 marks)** |

Candidate number______________________________

## Task 2: database design and development (part A)

Nora has recently been given a large number of comic books. She wants to create a database to manage the collection.

She wants to be able to use the database to find out information about the comics.

Some of the things she would like to find out are:

- the total value of comics that feature specific main characters
- what comics are above the average value of the collection
- what comics are of below average value of the collection

2a Assuming all the entities and attributes have been identified, use the end-user requirements above to create two additional functional requirements.

**(2 marks)**

Functional requirements:

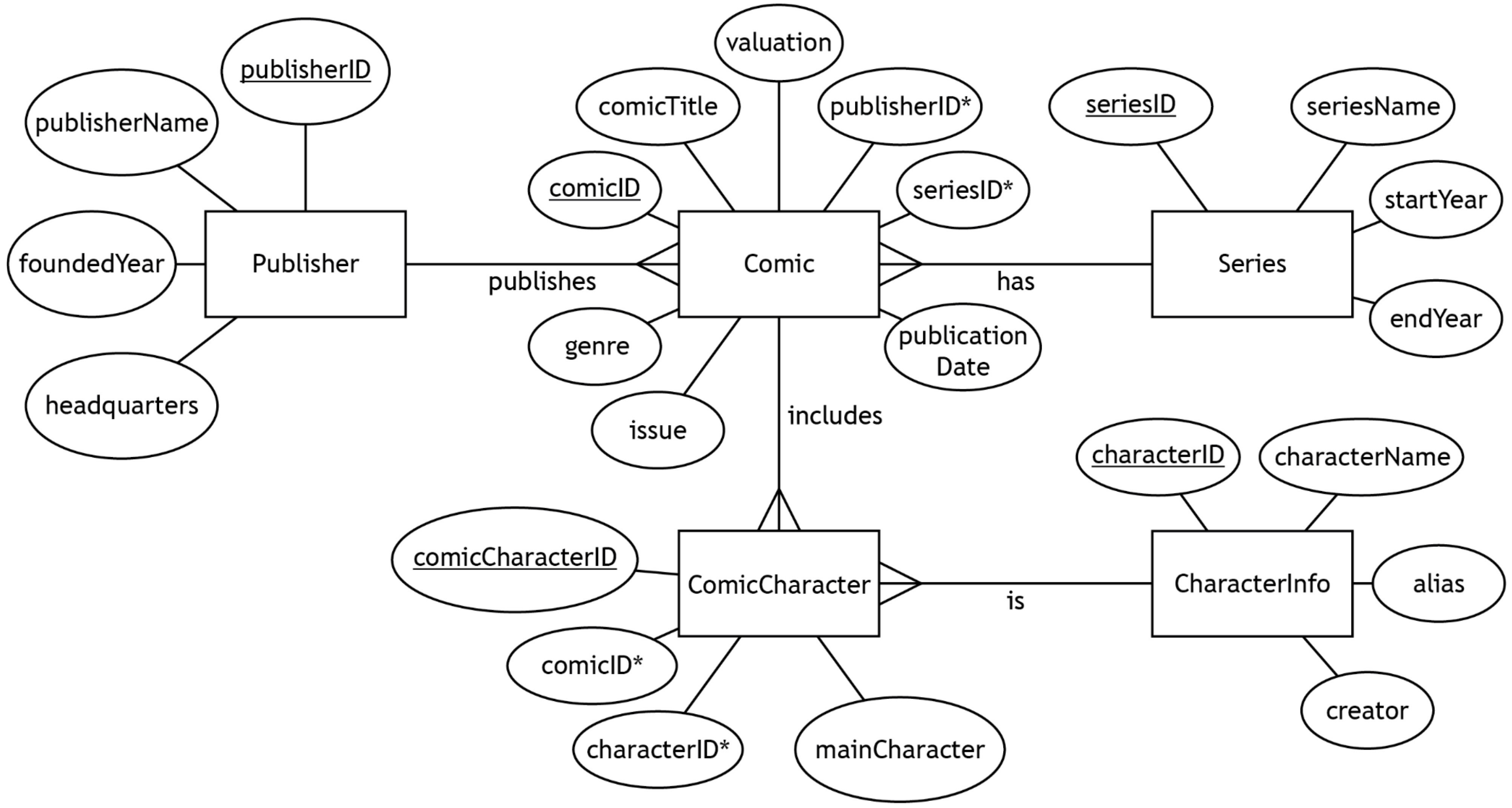
- A query to search for comics valued above the average value
- A query to search for comics valued below the average value

- Check your answers carefully, as you cannot return to part A after you hand it in.
- When you are ready, hand part A to your teacher or lecturer and collect part B.

Candidate number________________________________

## Task 2: database design and development (part B)

An entity-relationship diagram for the database was created and is shown below.



2b Explain why Nora had to include the ‘ComicCharacter’ entity in the final design.

**(1 mark)**

Candidate number________________________________

Your teacher or lecturer will provide you with a completed and populated database.

2c Nora would like to work out the average value of all the comics in the collection and those that are above this average.

Implement the SQL statement(s) to display a list of the comic title, issue, publisher name, and valuation for comics that are valued at least £300 above this average.

The expected output is shown below.

**(3 marks)**

| comicTitle | issue | publisherName | valuation |
|---|---|---|---|
| Duckguy Chronicles | 4 | Panel Pals | 558 |
| Falconer Fusion | 4 | Kapow Publications | 589 |
| Green Lamplight Avenged | 18 | Epic Doddles | 580 |
| Nightcrawlerly | 6 | Kapow Publications | 601 |
| Silver Surface | 5 | Epic Doddles | 590 |
| Superdude | 11 | Kapow Publications | 840 |
| The Incredible Sulk | 6 | Epic Doddles | 579 |
| The Incredible Sulk | 20 | Epic Doddles | 528 |

Print evidence of the implemented SQL statement(s) and the output produced.

2d Nora wants to see the total value of comics where the main character's name contains 'Duck'.

Implement the SQL statement to display the total values by character name with the highest value first.

The expected output is shown below.

**(5 marks)**

| characterName | Total Valuation |
|---|---|
| Dare-Duck | 1497 |
| Duckguy | 1153 |
| Duck Gal | 431 |
| Doctor Duck | 370 |

Print evidence of the implemented SQL statement and the output produced.

2e A comic book collector contacts Nora saying he would like to purchase any comics from the series ‘The OK Seven’ that feature the character named ‘Starlordly’. He is willing to pay double what they are worth to ensure he gets them.

Nora creates the following SQL statement to find any comics that match this description, showing the title of the comic, the issue, the publisher name, and the comic’s valuation after being doubled.

```
SELECT comicTitle, issue, publisherName, valuation AS [Double
Price]
FROM Comic, Publisher, Series, CharacterInfo, ComicCharacter
WHERE Series.seriesName = "The OK Seven"
AND characterName = "Starlordly"
AND Comic.publisherID = Publisher.publisherID
AND Comic.seriesID = Series.seriesID
AND CharacterInfo.characterID = ComicCharacter.characterID;
```

The query to test the above SQL is provided with the database. When run, the output appears to be incorrect.

Amend the query by making the required changes to produce the correct output.

Print evidence of the amended SQL query and the output produced.

**(2 marks)**

The expected output is shown below.

| comicTitle | issue | publisherName | Double Price |
|---|---|---|---|
| Silver Surface | 5 | Epic Doddles | 1180 |

2f A full list of functional requirements for the database is shown below.

- A query to calculate the total value of comics that include specific characters
- A query to calculate how much a comic has increased in value since it was purchased
- A query to calculate the average value of the comics in the collection
- A query to find and display a list of all the comics valued below the average value
- A query to find and display a list of all the comics valued at £300 or more above the average value
- A query to find all the comics from a series that started in the 1980s

(i) Identify the functional requirement that cannot be met using the current database structure.

**(1 mark)**

(ii) Explain, with reference to the database structure, what additional data would be required to allow this requirement to be met.

**(1 mark)**

Candidate number________________________________

## Task 3: web design and development (part A)

Paradise Mini Golf wants a website to promote the business and attract more customers. They commissioned a web developer to produce the website.

The developer identifies that the website should allow users to:

- reveal the relevant record score when they select ‘adult’, ‘junior’ or ‘senior’
- submit an enquiry about an event
- choose a menu (kids, a la carte, lunch, pre-theatre) on the ‘Food’ page

3a Use the end-user requirements above to create two additional functional requirements for the website.

**(2 marks)**

- A drop-down list allowing users to select a menu (Lunch, Pre-Theatre, A La Carte and Kids)

- Check your answers carefully, as you cannot return to part A after you hand it in.
- When you are ready, hand part A to your teacher or lecturer and collect part B.

Candidate number______________________________

## Task 3: web design and development (part B)

On completing the analysis stage, the developer identifies the following end-user and functional requirements.

**End-user requirements**
Customers want:

- information about Paradise Mini Golf
- to see pictures of the course and facilities at Paradise Mini Golf
- to find out about the course at Paradise Mini Golf
- to find out about the food available at Paradise Mini Golf
- to reveal the relevant record score when they select 'adult', 'junior' or 'senior'
- to be able to submit an enquiry via the website

**Functional requirements**
- the 'Home' page should contain:
  - information about Paradise Mini Golf
- the 'Food' page should contain:
  - information about the food on offer
  - a drop-down list allowing users to select a menu (Lunch, Pre-Theatre, A La Carte and Kids)
  - a button to view a selected menu
- the 'Course' page should contain:
  - information about the mini golf course
  - a link to the 'Scorecard' page
- the 'Prices' page should contain:
  - the prices for adults, senior citizens, juniors and infants
  - the prices for groups of differing sizes
  - the prices for parties
- the 'Contact' page should contain a form to allow customers to:
  - submit an enquiry
  - choose the category of enquiry
  - choose to join the mailing list or choose to opt out
- the 'Scorecard' page should contain:
  - a blank scorecard
  - a scorecard with the record score for an adult
  - a scorecard with the record score for a senior
  - a scorecard with the record score for a junior

Your teacher or lecturer will provide you with a copy of Paradise Mini Golf's incomplete website.

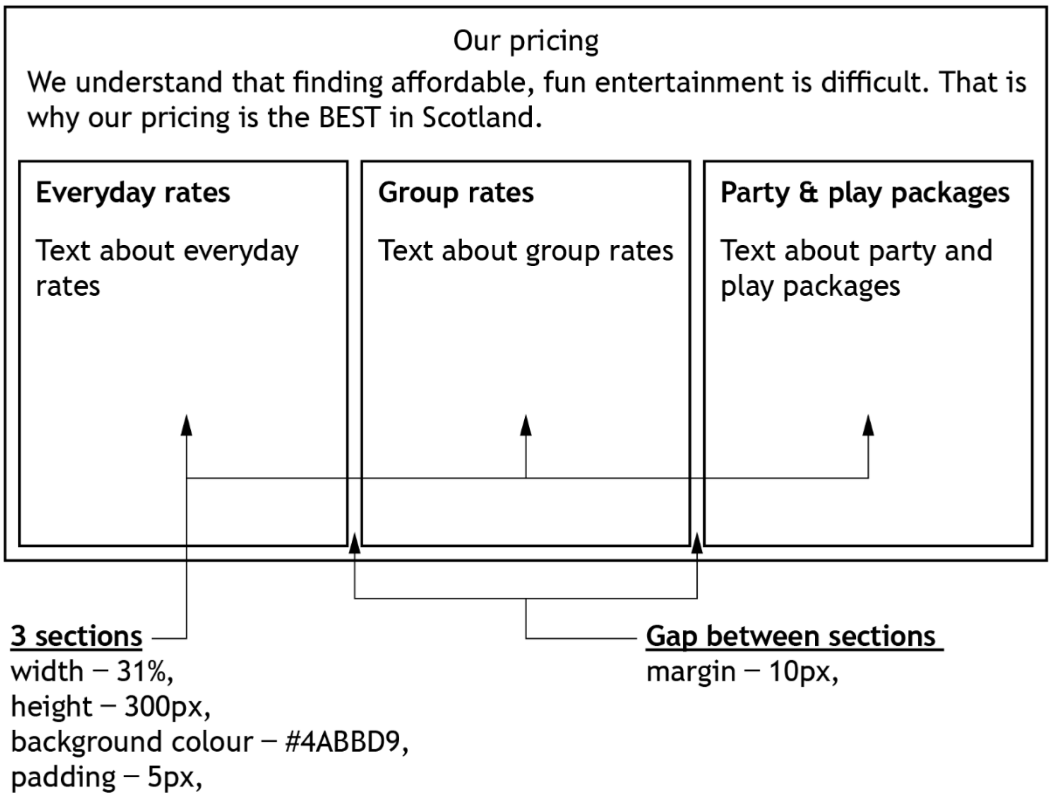
Examine the home page and each of the other pages on the website.

3b After seeing the initial prototype, the client asks to change the look of the 'Prices' page. The developer decides to change the layout of the page so that the prices are displayed in three side-by-side sections.

Open the 'prices.html' file in a suitable editor.

Edit the 'prices.html' file so that it matches the following wireframe:

**(4 marks)**

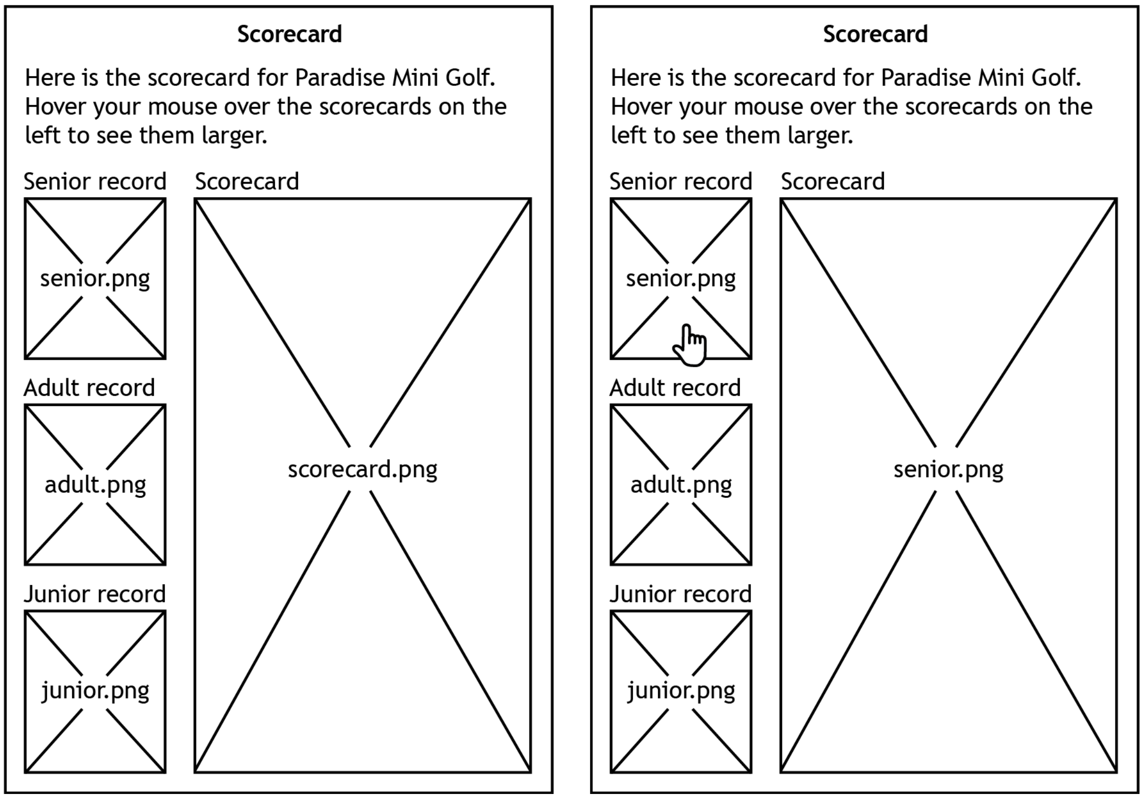

Our pricing
We understand that finding affordable, fun entertainment is difficult. That is why our pricing is the BEST in Scotland.
Everyday rates
Text about everyday rates
Group rates
Text about group rates
Party & play packages
Text about party and play packages
3 sections
width – 31%,
height – 300px,
background colour – #4ABBD9,
padding – 5px,
Gap between sections
margin – 10px,

Print evidence of the edited:

- 'prices.html' file
- 'style.css' file.

3c The developer decides to make the 'Scorecard' page interactive. Currently, it shows a blank scorecard with the three record scores on the left-hand side.

The wireframe design for the interactive 'Scorecard' page is shown below.


Scorecard
Here is the scorecard for Paradise Mini Golf. Hover your mouse over the scorecards on the left to see them larger.
Senior record
Scorecard
senior.png
Adult record
adult.png
Junior record
junior.png
scorecard.png
Scorecard
Here is the scorecard for Paradise Mini Golf. Hover your mouse over the scorecards on the left to see them larger.
Senior record
Scorecard
senior.png
Adult record
adult.png
Junior record
junior.png
senior.png

Edit the 'scorecard.html' by adding events to show:

- the record score for junior, adult and senior golfers when the mouse is moved over the corresponding image
- the blank scorecard when the mouse is moved away from the corresponding image

**(3 marks)**

Print evidence of the edited 'scorecard.html' file.

3d The ‘Contact’ page contains a form that users can fill in if they wish to contact Paradise Mini Golf.

Open the ‘contact.html’ page in a suitable editor and examine the form code carefully.

Describe how you would fully test the ‘contact number’ and ‘choose your enquiry’ form elements.

**(3 marks)**

Candidate number______________________________

3e State three reasons why the Paradise Mini Golf website is not fit for purpose.

**(3 marks)**

Candidate number_______________________________

## Copyright Acknowledgements

**Electronic Files:**

Image of Pizza – stockcreations/Shutterstock.com

## Administrative information

**Published:** January 2025 (version 1.1)

### History of changes

| Version | Description of change | Date |
|---|---|---|
| 1.1 | Error in date format in 1d corrected from ‘31-Oct,24’ to ’31-Oct-24’<br><br>‘Candidate name__________________’ removed from task sheets in separate documents as candidates are only required to note their SCN on each page of their evidence. | 03/02/25 |
| | | |
| | | |
| | | |

## Security and confidentiality

This document can be used by SQA approved centres for the assessment of National Courses and not for any other purpose.

This document may only be downloaded from SQA’s designated secure website by authorised personnel.

© Scottish Qualifications Authority 2025