H

Higher
Coursework
Assessment Task

SQA

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.

Specimen — valid from session 2018-19 and until further notice.

This edition: August 2018 (version 1.0)

Contents

Introduction

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

This assignment has 50 marks out of a total of 160 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 is a specimen assessment task.

SQA publishes a new assessment task on the secure website each academic year. The task is valid for that year 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 e-mail 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

Time

Candidates have 8 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 8-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 8 hours permitted for the assignment.

Resources

Each candidate must have access to a computer system with a high-level (textual) programming language, database application and software that can create, edit and run SQL, HTML and CSS.

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**.

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 created manually or 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 Word document.

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

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 name and candidate number. Any other pages submitted, for example, prints of program listings, screenshots, must have this information added to them.

Specific instructions for teachers and lecturers: specimen assignment

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 and design a database. 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. A Microsoft Access file (Doctors Surgery) is provided for candidates to use in part B. If your centre uses a different database management system, you can create the relational database for part B using the CSV files provided:

- appointment.csv
- doctor.csv
- drug.csv
- patient.csv
- prescribedDrug.csv

Specific instructions for database setup

The ‘Doctors surgery database’ table includes table names, field names, primary keys and foreign keys.

A compound key is required in the following tables:

- Appointment (compound key = appDate, appTime and doctorID)
- PrescribedDrug (compound key = drugID, patientID and datePrescribed)

You do not need to add validation to any of the fields in the database tables.

Doctors surgery database				
Doctor	**Patient**	**Drug**	**Appointment**	**PrescribedDrug**
doctorID	patientID	drugID	appDate	drugID*
forename	forename	drugName	appTime	patientID*
surname	surname	manufacturer	doctorID*	datePrescribed
street	dateOfBirth	dosage	patientID*	
town	address	cost		
postcode	contactNo			
telNo				
dateQualified				

Task 2

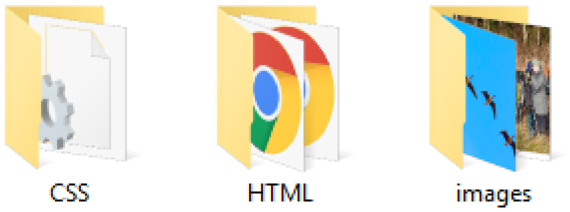
Give the following data file to candidates:

- beachData.csv

Task 3

The following file has been provided (BirdWatchingWebsiteIncomplete.zip).

When unpacked this contains the images, HTML and CSS files candidates need to complete this task. These files must remain in the folders shown below and must not be renamed.



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 50 marks out of a total of 160 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 three short practical tasks. You may complete the tasks in any order.

Advice on how to plan your time

You have 8 hours to complete the assignment. Marks are allocated as follows:

♦ Task 1 — database design and development	12 marks	(24% of total)
♦ Task 2 — software design and development	25 marks	(50% of total)
♦ Task 3 — web design and development	13 marks	(26% of total)

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

Advice on gathering evidence

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

Your evidence, especially code, must be clear and legible. This is particularly important when you paste screenshots into a Word document.

Use the evidence checklist provided to make sure you submit everything necessary at the end of the assignment. Ensure your name and candidate number are included on all your evidence.

Evidence may take the form of printouts of code/screenshots/typed answers, hand-written answers or drawings of diagrams/designs.

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

Task 1	Evidence	
Part A		
1a	Completed task sheet showing the relationships and cardinality on the entity-relationship diagram	□
1b	Completed task sheet showing the additional relationships and cardinality between the entities	□
Part B		
1c (i)	SQL statement implemented to list drugIDs showing how many times each drugID is prescribed (sorted from the most prescribed to least)	□
	Printout of output	□
1c (ii)	SQL statements implemented to identify a list of patients who have been prescribed the drug with the highest dosage	□
	Printout of outputs	□
1d	Completed task sheet showing responses on testing	□
Task 2	**Evidence**	
2a	Completed task sheet showing two boundaries	□
2b (i)	Printout of your program code	□
2b (ii)	Printout of your altered program code	□
2c	Completed task sheet showing description of comprehensive test plan	□
2d	Breakpoint clearly marked on printout of code	□
2e	Completed task sheet showing evaluation	□

Task 3	Evidence	
3a	Completed task sheet showing end-user and functional requirements	□
3b	Completed task sheet (or printout from a design application) of form design including validation	□
3c	Printout of HTML (applications.html) showing form code	□
	Printout of ‘Applications’ page as viewed in browser	□
3d (i) and (ii)	Printout of HTML and JavaScript (profile.html) showing code	□
	Printout of ‘In Profile’ page as viewed in browser	□

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

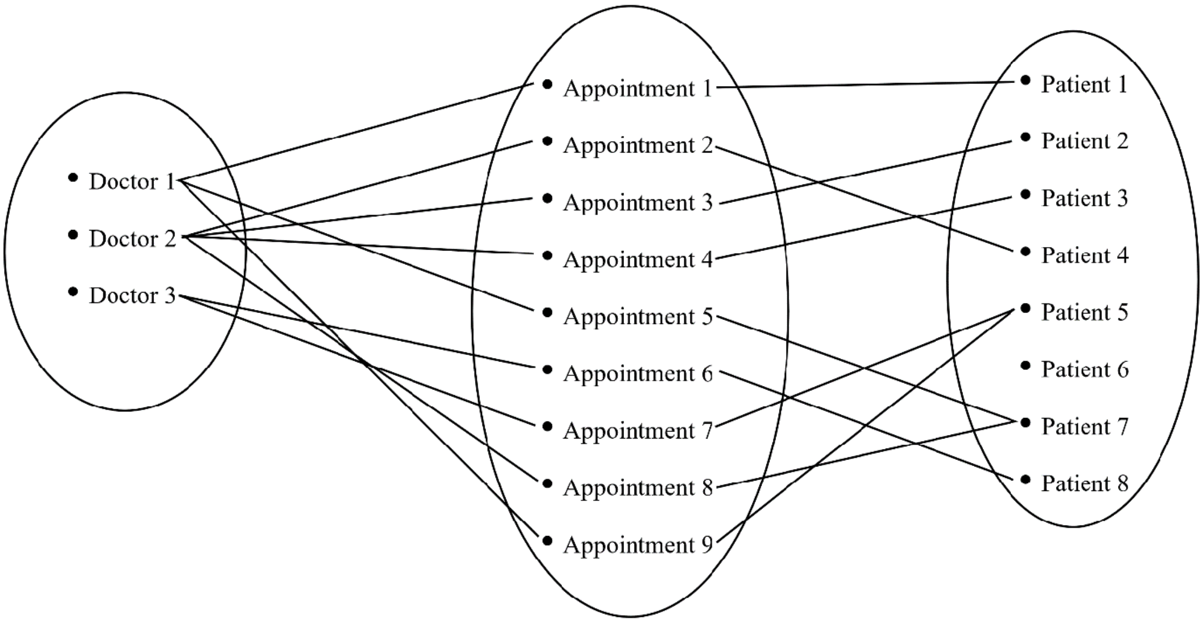
- Check you have completed all parts of tasks 1, 2 and 3.
- Label any printouts/screenshots with the task number (for example 1c (ii), 2a).
- Clearly display your name and candidate number on each printout.

Task 1: database design and development (part A)

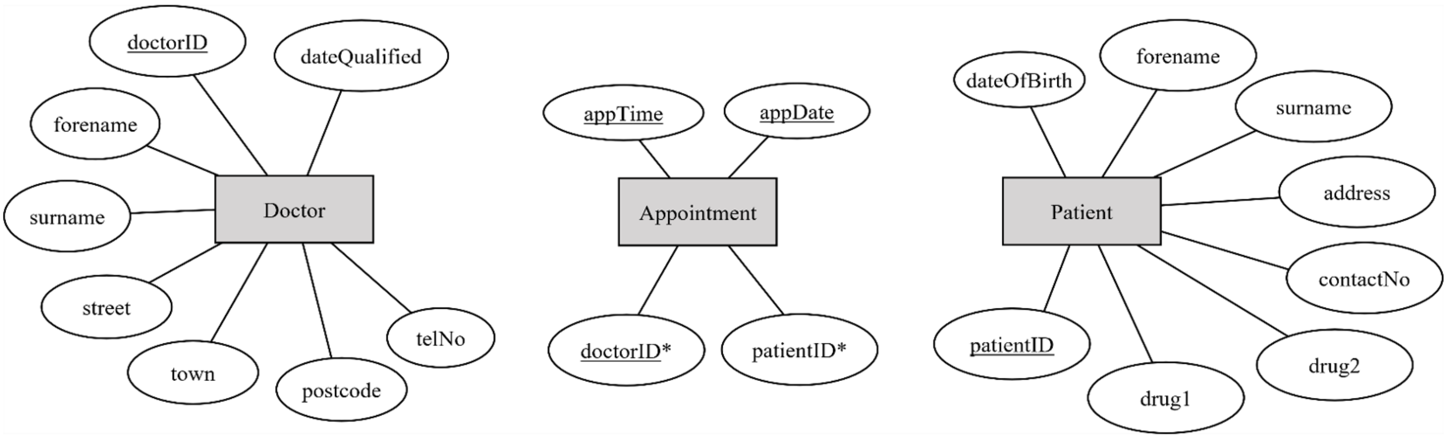
A doctors surgery wants a database to store information about their doctors, patients and appointments.

The surgery has appointed a database designer, who creates the two diagrams shown below.

Entity-occurrence diagram



Entity-relationship diagram



1a Using the information provided by the entity-occurrence diagram, add the relationships and cardinality to the above entity-relationship diagram.

(1 mark)

Candidate name________________________________ Candidate number________________

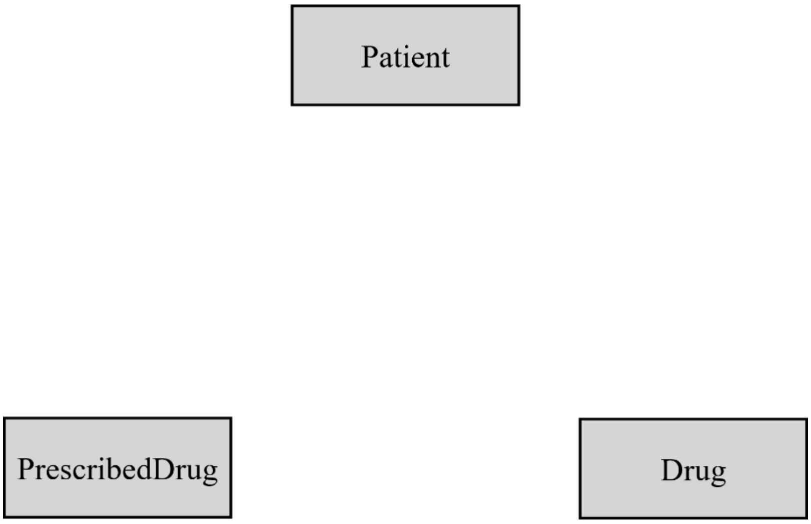
1b The current design shows that a maximum of two prescribed drugs can be added to a patient's record. Feedback from the doctors is that this is an error, as it is common for more than two drugs to be prescribed to one patient.

This error could be corrected by creating two separate entities to store the drugs prescribed to each patient and the details of all drugs.

- PrescribedDrug — a drug given to a patient (a drug can be prescribed on different dates)
- Drug — details about each prescribed drug

Show how these two new entities could be used to correct the design, by adding additional relationships and cardinalities between the entities.

(2 marks)



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

Candidate name________________________________ Candidate number________________

Task 1: database design and development (part B)

Your teacher or lecturer will provide you with a completed database file. This file contains a relational database with the following tables.

Doctors surgery database				
Doctor	**Patient**	**Drug**	**Appointment**	**PrescribedDrug**
doctorID	patientID	drugID	appDate	drugID*
forename	forename	drugName	appTime	patientID*
surname	surname	manufacturer	doctorID*	datePrescribed
street	dateOfBirth	dosage	patientID*	
town	address	cost		
postcode	contactNo			
telNo				
dateQualified				

1c (i) The surgery wants to create a list of drugIDs showing how many times each drugID is prescribed. The list should be sorted from the most prescribed drug to the least.

Implement the SQL statement that will produce this output.

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

(3 marks)

1c (ii) The surgery wants to identify a list of patients who have been prescribed the drug with the highest dosage.

The output should include the patientID and the datePrescribed fields, as shown below.

patientID	datePrescribed
75	14/05/2018
88	14/05/2018
92	10/05/2018
…	…

Implement two SQL statements that will produce this output.

Print evidence of both the implemented SQL statements and the outputs they produced.

(4 marks)

1d The five patients listed below live in the postcode area EH12:

- Susan Lamb
- Gifty Adisa
- Lewis Robinson
- Morgane Dubios
- Toby Malone

The following SQL statement is designed to find all the patients who live within this postcode area.

```
SELECT forename, surname, patientID
FROM Patient
WHERE address LIKE '%EH12%';
```

Test the above SQL statement.

Comment on any difference between the actual output it produces and the expected output.

(1 mark)

Describe how the Patient table could be changed to ensure the expected output matches the actual output.

(1 mark)

Candidate name________________________________ Candidate number________________

Task 2: software design and development

The Marine Conservation Society collects data on the amount of pollution on UK beaches and gives each beach a rating.

		Rating
Excellent	(excellent water quality and local sewage treatment)	1
Good	(good water quality)	2
Poor	(water quality meets minimum standards)	3
Fail	(water quality below minimum standards)	4
Not tested		5

Program analysis

A program is required to read the beach pollution data from a CSV file. This file contains the names of 973 UK beaches, along with a rating for each beach.

When run, the program should:

- display the average rating for all beaches tested
- prompt the user to select a rating and display the names of all beaches with that rating

Assumptions

- the data file is formatted correctly and is error-free
- each line of the data file stores the name and rating (1 to 5) for a single beach, as shown below:

```
Aber Mawr Bay,1
Aberaeron - Harbour (Fourth Groyne North),5
Aberafan,3
Aberarth,3
Abercastle,5
Aberdaron Beach,1
…
```

Functional requirements

Inputs

- data from the file (name of beach and rating)
- user enters rating for the beaches to be displayed

Processes

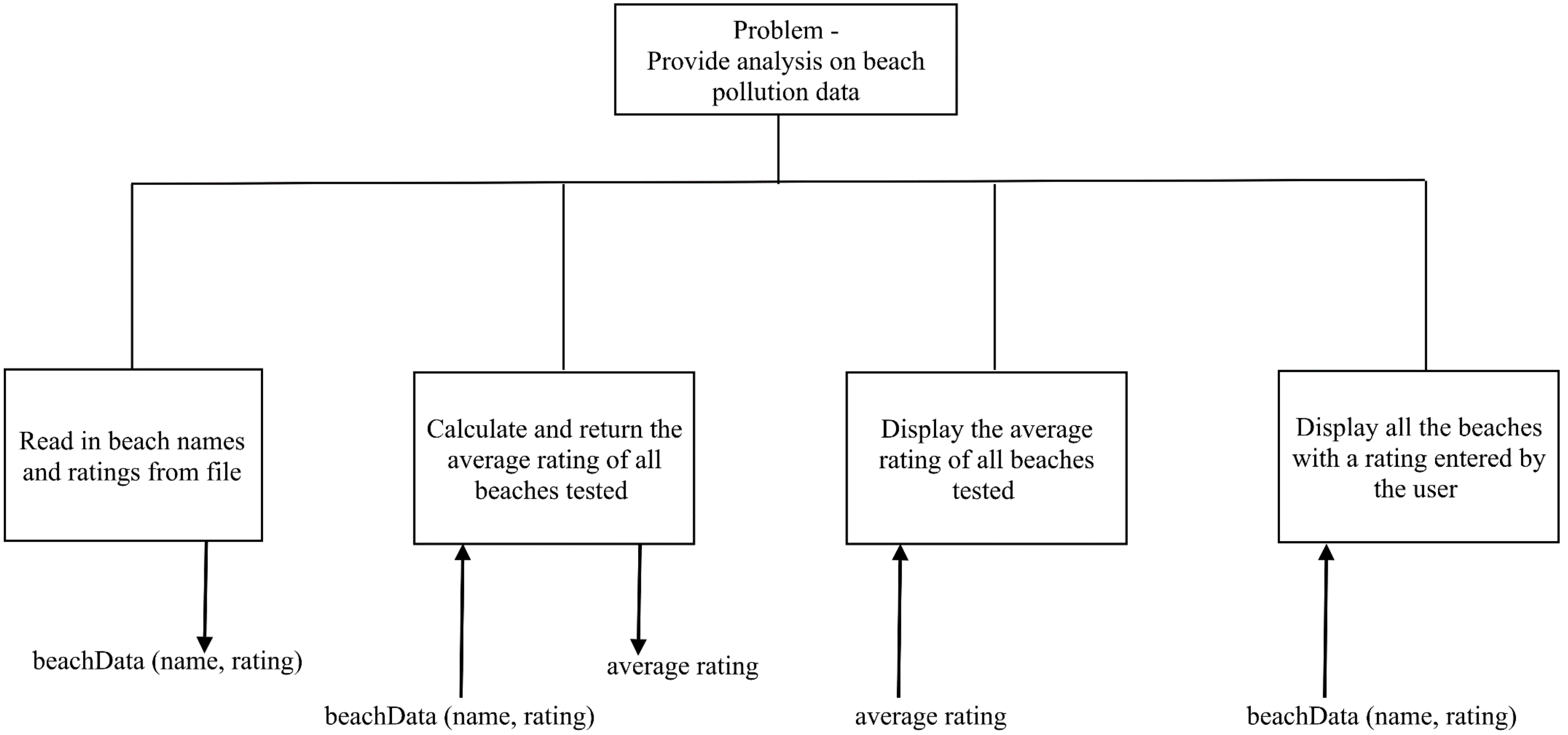
- calculate the average rating for all beaches tested (those with a rating between 1 and 4)
- only display names of beaches with a rating entered by the user

Outputs

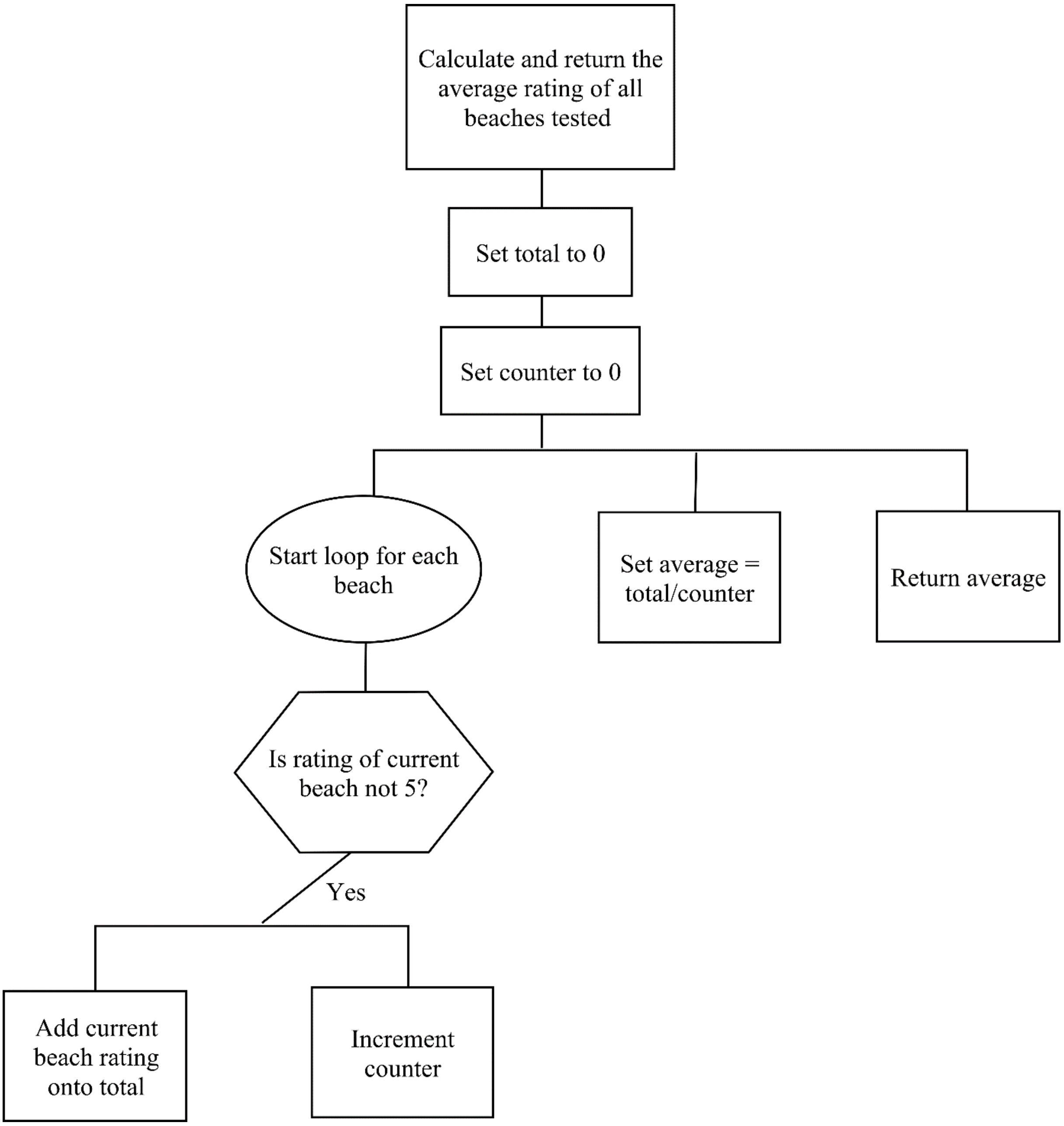
- the average beach rating for all beaches tested
- the name of all beaches with a rating entered by the user

Program design (structure diagram)

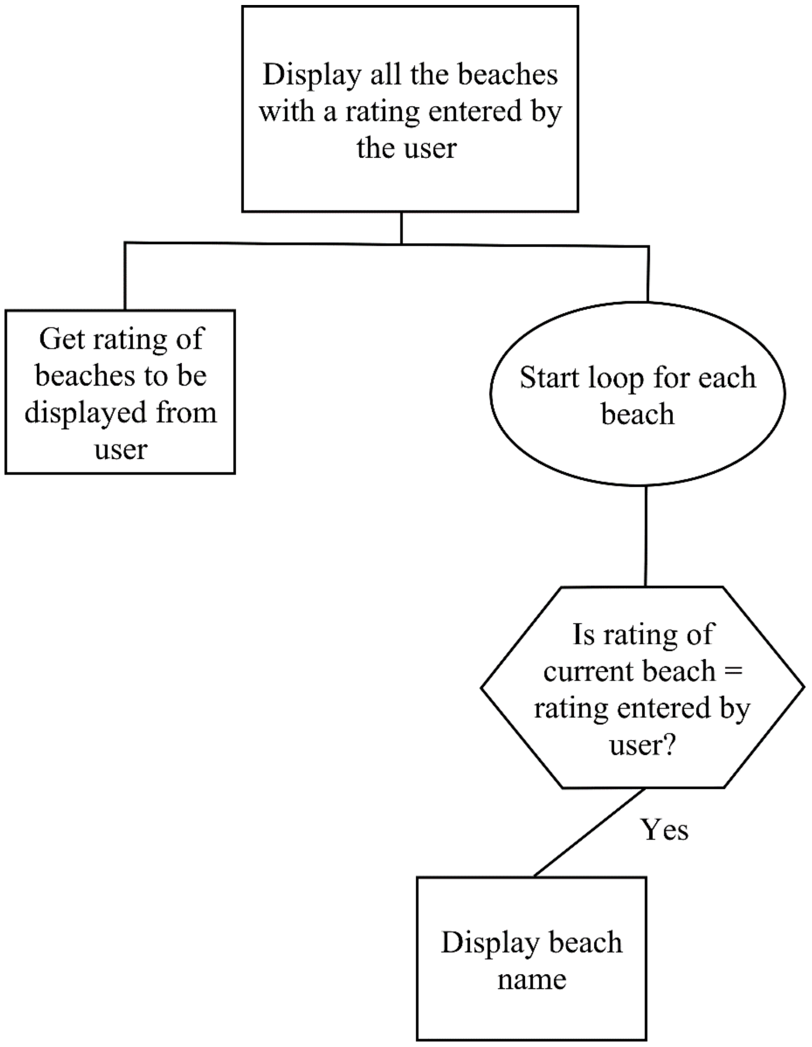
Top level design with data flow


Problem -
Provide analysis on beach pollution data
Read in beach names and ratings from file
Calculate and return the average rating of all beaches tested
Display the average rating of all beaches tested
Display all the beaches with a rating entered by the user
beachData (name, rating)
beachData (name, rating)
average rating
average rating
beachData (name, rating)

Refinement of ‘Calculate and return the average rating for all beaches tested’


Calculate and return the average rating of all beaches tested
Set total to 0
Set counter to 0
Start loop for each beach
Set average = total/counter
Return average
Is rating of current beach not 5?
Yes
Add current beach rating onto total
Increment counter

Refinement of ‘Display all the beaches with a rating entered by the user’


Display all the beaches with a rating entered by the user
Get rating of beaches to be displayed from user
Start loop for each beach
Is rating of current beach = rating entered by user?
Yes
Display beach name

Task 2: software design and development

2a Using the program analysis and design, identify two boundaries in this problem.

(2 marks)

Boundary 1:
Boundary 2:

Candidate name________________________________ Candidate number________________

2b (i) Using the program analysis and design, implement the program in a language of your choice. Your program should:

- be maintainable and modular
- use a function to 'calculate and return the average rating for all beaches tested'
- use a procedure to 'display all the beaches with a rating entered by the user'
- match the top level design and the refinements provided

Print evidence of your program code.

(12 marks)

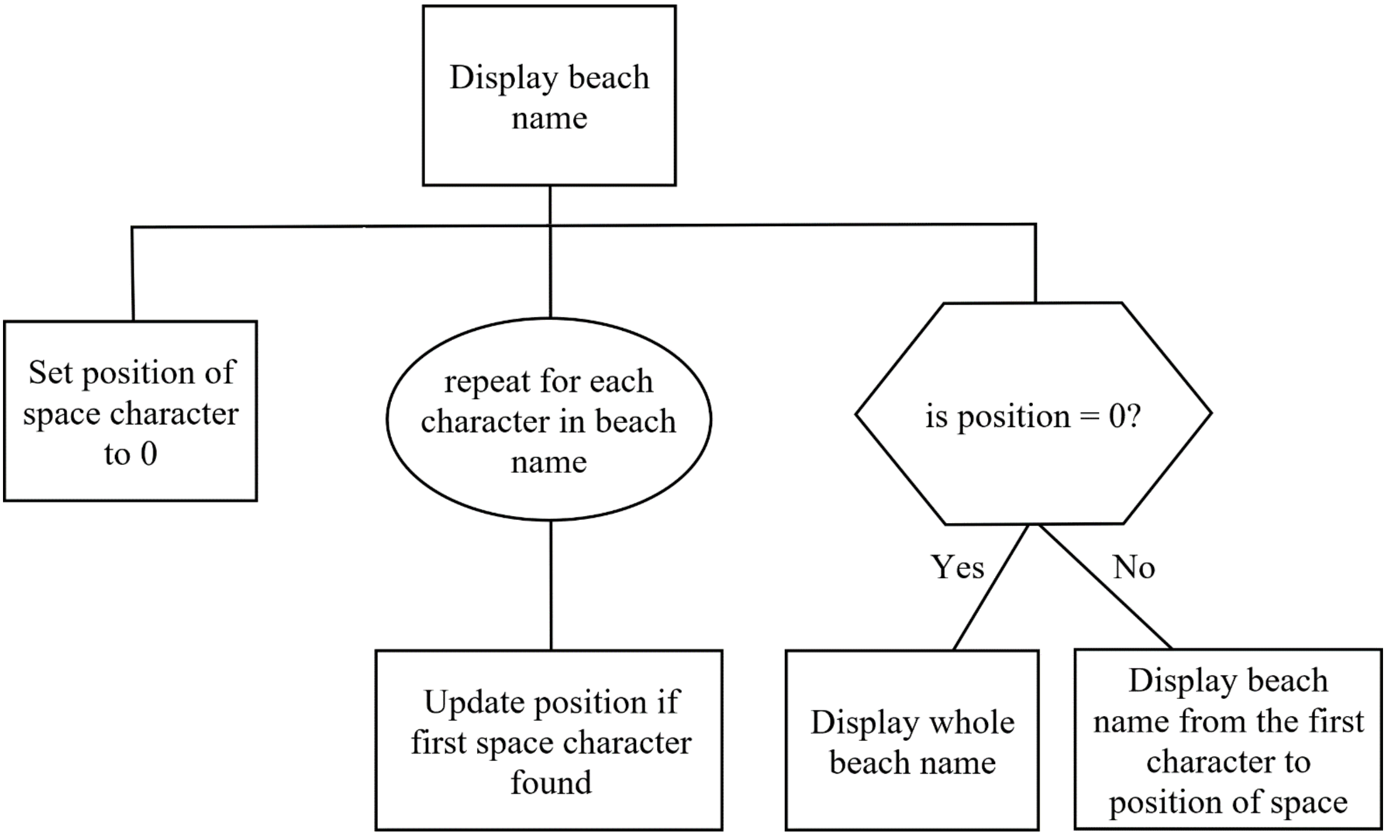
2b (ii) The process 'display all the beaches with a rating entered by the user' currently displays the complete name of each beach.

The program is redesigned so that if the beach name is longer than one word, only the first word is displayed.

For example: `Aberdaron Beach` becomes `Aberdaron`

A refinement of 'Display beach name' is shown on the following page. This finds the position of the first space character in the string and displays only the characters up to this position.

Refinement of ‘Display beach name’


Display beach name
Set position of space character to 0
repeat for each character in beach name
Update position if first space character found
is position = 0?
Yes
No
Display whole beach name
Display beach name from the first character to position of space

Using the above design, alter your program code so that only the first word in the beach name is displayed.

Print evidence of your altered program code.

(3 marks)

2c Describe a comprehensive test plan that could be used to test the program inputs.

(2 marks)

2d Identify where a breakpoint could be used to test that only the beach names with a rating entered by the user are displayed.

Mark this clearly on one of the printouts of your code.

(1 mark)

Candidate name_________________________________ Candidate number______________

2e With reference to your own program code, evaluate:

♦ the robustness of your program if additional beach data was added to the CSV file **(1 mark)**
♦ the usability of your program **(2 marks)**

Candidate name__________________________________ Candidate number_________________

- the maintainability of your program

(2 marks)

Candidate name__________________________________ Candidate number_________________

Task 3: web design and development

Cool Twitching is a birdwatching club.

The club uses a noticeboard in its meeting room to display photographs and information about interesting birds each month, and to organise excursions to famous birdwatching areas. It also displays photographs and articles of previous excursions.

In the meeting room, the club provides a regular newsletter for members, and application forms to join.

The club is working on a small website. When complete, the website will include all the information currently available in the meeting room.

3a State **two** end-user requirements and **one** functional requirement for the website.

End-user requirement 1 **(1 mark)**
End-user requirement 2 **(1 mark)**
Functional requirement **(1 mark)**

Candidate name_____________________________ Candidate number______________________

3b To join the club, applicants must complete all parts of an application form that requires the following information:

- forename
- surname
- country (Scotland, England, Northern Ireland or Wales)
- age range (under 18, 18–65 or over 65)
- a short paragraph about their own area of interest (max 300 characters)

Design a form suitable for entering the required information on the new applications page on the website (you need to implement this form in the next task, so ensure your design is achievable).

You can design the form in the space below or use a suitable design application.

(2 marks)

Candidate name____________________________ Candidate number______________________

3c Your teacher or lecturer will provide you with a copy of Cool Twitching's incomplete website.

Open the home page in a browser. Examine the home page and each of the other pages in the website.

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

Write HTML code to implement your form. Your coded form should:

- match your design
- include any input validation required
- include presence checks on each form element

Print evidence of the following:

- the code for the form in the applications.html file
- the 'Applications' page as viewed in a browser

(4 marks)

3d The ‘In Profile’ page on the website contains four section elements, each containing text and an image relating to one species of bird.

Open the file called ‘profile.html’ in a suitable editor.

(i) Using code, hide three of the section elements, leaving only the section containing information on the kingfisher visible.

(1 mark)

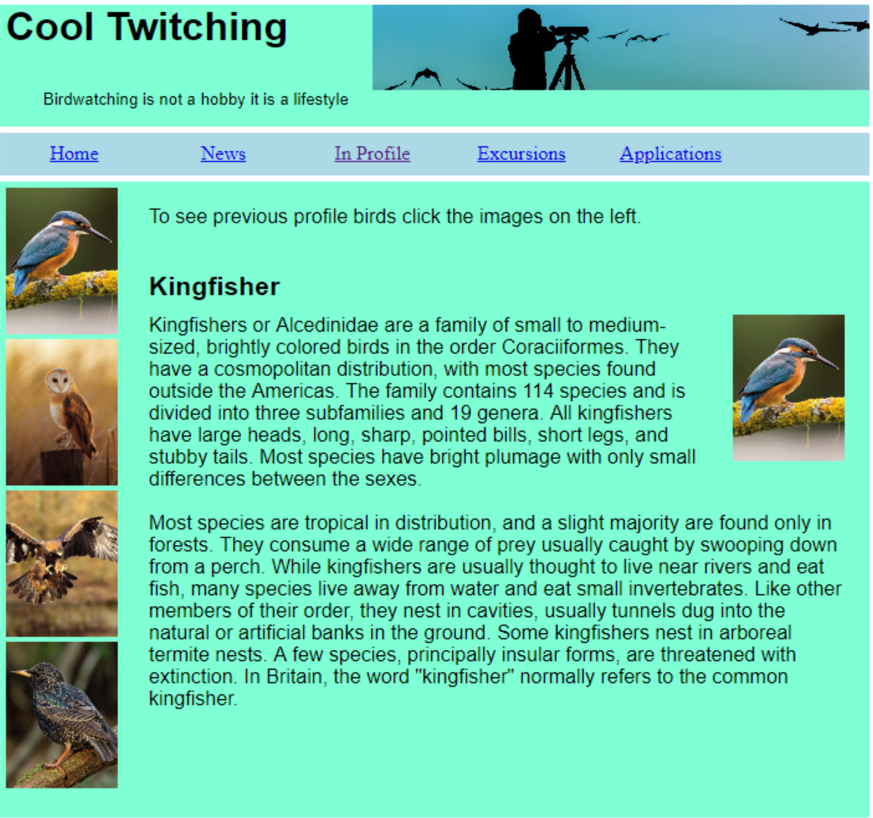
3d (ii) Add JavaScript to create the following interactive content:

When any of the bird pictures on the left of the page are clicked, a matching section element is displayed. For example, clicking on the owl image will display only the section containing the owl heading, text and image.

Only one of the four sections should be visible at any time.

(3 marks)

The implemented page should look like this:



Cool Twitching

Birdwatching is not a hobby it is a lifestyle

To see previous profile birds click the images on the left.

Kingfisher

Kingfishers or Alcedinidae are a family of small to medium-sized, brightly colored birds in the order Coraciiformes. They have a cosmopolitan distribution, with most species found outside the Americas. The family contains 114 species and is divided into three subfamilies and 19 genera. All kingfishers have large heads, long, sharp, pointed bills, short legs, and stubby tails. Most species have bright plumage with only small differences between the sexes.

Most species are tropical in distribution, and a slight majority are found only in forests. They consume a wide range of prey usually caught by swooping down from a perch. While kingfishers are usually thought to live near rivers and eat fish, many species live away from water and eat small invertebrates. Like other members of their order, they nest in cavities, usually tunnels dug into the natural or artificial banks in the ground. Some kingfishers nest in arboreal termite nests. A few species, principally insular forms, are threatened with extinction. In Britain, the word "kingfisher" normally refers to the common kingfisher.

Print evidence of the following:

- the code added to implement 3d (i) and 3d (ii)
- the ‘In Profile’ page as viewed in a browser

Marking instructions

The following marking instructions are for the Higher Computing Science specimen assessment task. In line with SQA's normal practice, they are addressed to the marker. They will also be helpful if you are preparing candidates for course assessment.

Marking instructions are not provided for annual assessment tasks. Candidates' evidence is submitted to SQA for external marking.

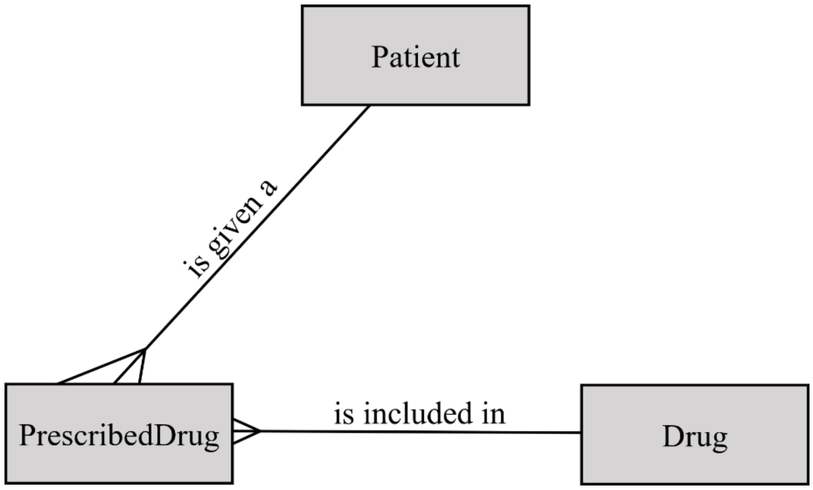
General marking principles

Always apply these general principles. Use them in conjunction with the specific marking instructions, which identify the key features required in candidates' responses.

a Always use positive marking. This means candidates accumulate marks for the demonstration of relevant skills, knowledge and understanding; marks are not deducted for errors or omissions.

b If a candidate response does not seem to be covered by either the principles or specific marking instructions, and you are uncertain how to assess it, you must seek guidance from your team leader.

Specific marking instructions

Task	Expected response	Additional guidance	Marks available	
1	**Database design and development — part A**			
1a	Award **1 mark** for completing: ♦ names of relationships ♦ 1 – M – 1 cardinality		1	**Design (3)**
1b	Award **1 mark each** for: ♦ Patient-PrescribedDrug: relationship and cardinality ♦ Drug-PrescribedDrug: relationship and cardinality	Description of relationship may vary. Award 1 mark for both cardinalities with no named relationships.	2	
	Solution for 1b  Patient is given a PrescribedDrug is included in Drug			

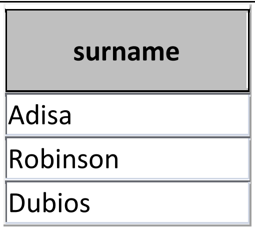
Task	Expected response	Additional guidance	Marks available	
1	**Database design and development — part B**			
1c (i)	Award **1 mark each** for: ♦ SELECT — drugID — drugID with COUNT function FROM PrescribedDrug ♦ GROUP BY drugID ♦ ORDER BY COUNT(drugID) DESC		3	**Implementation (7)**

	`SELECT drugID, COUNT(drugID)`
	`FROM PrescribedDrug`
	`GROUP BY drugID`
	`ORDER BY COUNT(drugID) DESC;`			

1c (ii)	Award **1 mark** for query to find the drug with the highest dosage: ♦ Query1 SELECT Max(dosage) FROM Drug; Award **1 mark each** for: ♦ SELECT patientID, datePrescribed FROM PrescribedDrug, Drug ♦ WHERE PrescribedDrug.drugID = Drug.drugID ♦ AND dosage = [Query1 result]	Do not award a mark for the final bullet point if the candidate answers “AND dosage = 2000”. This implies they have manually searched for the drug with the highest dosage. If the candidate implements the search using a sub-clause: `AND Drug.dosage = (SELECT MAX(Drug.dosage) FROM Drug;)` still award marks for: ♦ Query 1 ♦ AND dosage =	4	

Task	Expected response	Additional guidance	Marks available	
1	**Database design and development — part B**			

	Query 1 saved as MaxDosage:
	`SELECT Max(dosage) AS MaxDose`
	`FROM Drug;`
	Query 2:
	`SELECT PrescribedDrug.patientID, PrescribedDrug.datePrescribed`
	`FROM PrescribedDrug, Drug, MaxDosage`
	`WHERE PrescribedDrug.drugID = Drug.drugID`
	`AND Drug.dosage = MaxDose;`			

1d	Award **1 mark each** for: ♦ the SQL only finds three of the five patients ♦ the address field could be split into components and one of these could be a required postcode	 surname Adisa Robinson Dubios	2	**Testing (2)**

Task	Expected response		Additional guidance	Marks available	
2	**Software design and development**				
2a	Award **1 mark each** for any two of the following: ♦ there are 973 beaches in file ♦ the file data (names and rating) is assumed to be accurate ♦ no input validation required for file data ♦ rating entered by user will only be an integer from 1 to 5 ♦ the user can only enter one beach to display (as the program only runs once)			2	**Analysis (2)**
2b (i)	Input from file	Procedure with use of parameter passing (beach data record structure) created to read file data		1	**Implementation (12)**
		Read data from file		1	
		Store beach names and ratings in a record structure		1	
	Calculate average rating	Function with use of parameter passing (beach data record structure) created to return average rating		1	
		Count occurrences algorithm. Award **2 marks** for: ♦ fixed loop for each record ♦ selection (not 5) ♦ running total of ratings ♦ incrementing counter	Award 1 mark for three bullets Award 0 marks for fewer than three bullets	2	
		Average rating calculated correctly		1	

Task	Expected response		Additional guidance	Marks available
2	**Software design and development**			
	Display selected	Procedure with use of parameter passing (beach data record structure) created to display selected beaches		1
		Linear search algorithm used to display user-selected records Award **2 marks** for: ♦ user input validated as an integer between 1 and 5 ♦ selection — record data matching user input ♦ display output — beach name	Award 1 mark for two bullets Award 0 marks for fewer than two bullets	2
	Program code is maintainable Award **1 mark** for any two bullets: ♦ internal commentary ♦ white space ♦ meaningful identifiers			1
	Matches supplied refinements			1

Task	Expected response		Additional guidance	Marks available	
2	**Software design and development**				
2b (ii)	Single word output	Position of space identified		1	**Implementation (3)**
		Sub-string used to display first word in string		1	
		Solution accounts for beach names with originally one word		1	
2c	Award **1 mark each** for: ♦ description of test used to check reading of data from file ♦ description of tests used for user input (1 to 5)			2	**Testing (3)**
2d	Breakpoint drawn on code to pause code when condition 'rating = 4' is met			1	
2e	Award **1 mark** for evaluating the robustness of the program in relation to coping with a longer file. Award **1 mark each** for evaluating the usability of the program, for any two of the following: ♦ the general user interface ♦ the user prompts ♦ the screen layout ♦ any help screens Award **1 mark each** for evaluating the maintainability of the program, in relation to two of the following: ♦ internal commentary ♦ modularity ♦ white space ♦ meaning identifiers			5	**Evaluation (5)**

Task	Expected response	Additional guidance	Marks available	
3	**Web design and development**			
3a	Award **1 mark each** for stating two end-user requirements from the following: ♦ users can access the regular newsletters ♦ users can apply to join the club ♦ users can learn about different birds ♦ users can read about upcoming excursions ♦ users can read about previous excursions ♦ users can see photographs of previous excursions Award **1 mark** for stating one functional requirement from the following: ♦ users can complete an application form ♦ photographs can be displayed ♦ information about interesting birds can be displayed ♦ excursion information can be displayed		3	**Analysis (3)**

Task	Expected response	Additional guidance	Marks available	
3	**Web design and development**			
3b	Award **1 mark** for each complete bullet: ♦ all required form elements for: — forename — surname — country — age — area of interest ♦ correct input types identifiable from design: — input for forename, surname — drop down or radio button input for country and age — input area for paragraph	Mark using the form design	2	**Design (2)**
3c	Award **1 mark each** for: ♦ forename and surname with ‘required’ ♦ country and age with correct options ♦ text area with max 300 characters and ‘required’ ♦ submit button	Mark using the form code and browser screenshot	4	**Implementation (8)**
3d (i)	Award **1 mark** for: ♦ three sections set to display:none	Display may be implemented as inline HTML or external CSS	1	

Task	Expected response	Additional guidance	Marks available	
3	**Web design and development**			
3d (ii)	Award **1 mark each** for: ♦ IDs assigned to sections ♦ onclick events added to images ♦ procedure shows one element while hiding the others		3	

Copyright acknowledgements

All images Shutterstock:

Page 27: Bird watchers silhouette — 112279877
Page 27: Kingfisher — 696120013
Page 27: Owl — 270202490
Page 27: Eagle — 383270053
Page 27: Starling — 593417801

Images used in ‘BirdWatchingWebsiteIncomplete.zip’ for task 3:

Bird watchers silhouette — 112279877
Pink footed geese — 520667317
Bird watchers — 1040887033
Kingfisher — 696120013
Owl — 270202490
Eagle — 383270053
Starling — 593417801

Administrative information

Published: August 2018 (version 1.0)

History of changes

Version	Description of change	Date

Note: you are advised to check SQA's website to ensure you are using the most up-to-date version of this document.

Security and confidentiality

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