



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.

Specimen – valid from session 2023-24 and until further notice.

This edition: August 2023 (version 1.0)

© Scottish Qualifications Authority 2023

Contents

Introduction	1
Instructions for teachers and lecturers	2
Marking instructions	7
Instructions for candidates	15

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 is a specimen assessment task for Higher Computing Science.

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 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 enables 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 or software that can create, edit and run SQL
- ◆ software that can create, edit and run HTML, 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 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 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 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 or screenshots, must have this information added to them.

Specific instructions for teachers and lecturers: specimen assignment

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 and design a solution to a software 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. Candidates must still have access to the problem description page during part B.

Give the following data file to candidates:

- ♦ athletes.csv

A CSV file is provided for candidates to use in this part of the task. You must not convert the CSV 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 and design a database 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 A and change their responses.

A Microsoft Access file (MyTreat.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.

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.

Specific instructions for database setup

The 'MyTreat' database includes table names, field names, primary keys and foreign keys.

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

MyTreat database			
Customer	CustomerOrder	Voucher	Supplier
<u>custID</u>	<u>orderID</u>	<u>voucherID</u>	<u>supplierCode</u>
firstName	custID*	voucherName	supplierName
surname	voucherID*	category	email
email	quantityPurchased	price£	
	dateOrdered	expiryDate	
		supplierCode*	
		quantityAvailable	

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 (Q2e_Query.txt) to add it to the database you provide to candidates.

Task 3 – part A requires candidates to design 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. Candidates must still have access to the end-user and functional requirements page during part B.

A folder named 'Web files' is provided. This contains the CSS, HTML and media 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.

Marking instructions

In line with SQA's normal practice, the following marking instructions for the Higher Computing Science assignment are addressed to the marker. They will also be helpful for those preparing candidates for course assessment.

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 detailed/specific marking instructions, and you are uncertain how to assess it, you must seek guidance from your team leader.
- c Award marks regardless of spelling, as long as the meaning is unambiguous and does not result in a syntax error in implemented code.
- d For design and implementation tasks, a sample response may be shown in the detailed marking instructions. This will not be the only valid response. You must use the detailed marking instructions and additional guidance to ensure that you consider alternative approaches and nuances of different programming languages. If in doubt you should refer to your team leader.
- e If a candidate puts a score through their entire response to a question and makes a further attempt, you should only mark the further attempt. If no further attempt is made and the original is legible, you should mark the original response.
- f In the marking instructions, if a word is underlined then it is essential; if a word is in brackets() then it is not essential. Words separated by / are alternatives.

Specific marking instructions

Task 1 – software design and development

Task	Expected response	Max mark	Additional guidance
1a	<p>Correct input (1 mark):</p> <ul style="list-style-type: none"> ♦ entryID, location, forename, surname, and jumps from CSV file <p>All three processes (1 mark):</p> <ul style="list-style-type: none"> ♦ generate bib value ♦ find the most jumps ♦ find the full name of athlete(s) who completed the highest number of jumping jacks 	2	<p>Award 0 marks for inputs if each input is not explicitly listed or it is not stated that inputs are read from a CSV file.</p> <p>All bullet points listed under processes must be included for mark to be awarded.</p> <p>Ignore any additional writing or/reading to/from file processes.</p>
1b	<p>Generate bib values:</p> <ul style="list-style-type: none"> ♦ IN: entryID (), forename (), surname(), location() <p>Find the highest number of jumps:</p> <ul style="list-style-type: none"> ♦ OUT: maxJumps <p>Display the full name of athletes:</p> <ul style="list-style-type: none"> ♦ IN: maxJumps, forename(), surname(), jumps () 	3	<p>Variable and arrays should be correctly identified.</p> <p>The same variable name selected for maximum jumps passed out in the 3rd procedure should match the variable passed in to the 4th procedure.</p>

Task	Expected response	Max mark	Additional guidance
1c(i)	<p>Read in athletes' data (3 marks):</p> <ul style="list-style-type: none"> ♦ read data from file ♦ assign athletes' data to parallel arrays ♦ module with data passed or returned to arrays <p>Generate and store bib value (4 marks):</p> <ul style="list-style-type: none"> ♦ module with correct parameters passed ♦ substrings extracted in loop and bib value concatenated ♦ ASCII value of first letter of location concatenated ♦ write entry ID and bib value to file <p>Find max number of jumps (3 marks):</p> <ul style="list-style-type: none"> ♦ module with correct parameters passed in and out ♦ loop from second athlete ♦ initialise and find max jumps <p>Display athletes with most jumps (1 mark):</p> <ul style="list-style-type: none"> ♦ module with correct parameters passed in and athletes' full names displayed <p>Implementation (2 marks):</p> <ul style="list-style-type: none"> ♦ matching top level design (use function to find max jumps) ♦ modular and maintainable (meaningful variable names and appropriate internal commentary) 	13	<p>Award 1 mark for each bullet point.</p> <p>A minimum of one comment describing the purpose of each module is required.</p>

Task	Expected response	Max mark	Additional guidance
1c(ii)	<p>Location of next year's host (2 marks):</p> <ul style="list-style-type: none"> ♦ module with correct parameters passed in ♦ correct values counted for each location 	2	
1d	<ul style="list-style-type: none"> ♦ maxjumps set to jumps(0) with corresponding value of 100 ♦ watchpoint triggered at index 2 with value 102 ♦ watchpoint triggered at index 3 with value 108 	3	Do not award marks for bullets two and three if treated as a trace table.
1e	<p>Evaluation of the following:</p> <ul style="list-style-type: none"> ♦ Fitness for purpose (1 mark): <ul style="list-style-type: none"> — meets requirements to generate bib values and store to file or — not fit for purpose as bib values may not be unique ♦ Maintainability (1 mark): <ul style="list-style-type: none"> — linking modularity to maintainability, for example, the use of local variables allow modules to be edited independently 	2	

Task 2 – database design and development

Task	Expected response	Max mark	Additional guidance
2a	<ul style="list-style-type: none"> ♦ (A query to) calculate total customer spend ♦ (A query to) calculate the number of vouchers sold by a single supplier ♦ (A query to) search for offers within a specific category ♦ (A query to) sort offers from best to worst ♦ (Queries to) update, insert or delete data 	2	<p>Award 1 mark for each bullet. Maximum 2 marks.</p> <p>Functional requirements must be extracted from end-user information.</p> <p>Award a maximum of 1 mark for update/insert/delete example.</p>
2b	<ul style="list-style-type: none"> ♦ entity names, in the correct order ♦ adding the correct number of instances ♦ adding the correct associations 	3	<p>Entities can be written in reverse order.</p>

Entity Names

Customer	CustomerOrder	Voucher	Supplier
Customer1	Order1	Voucher1	• Supplier1
Customer2	Order2	Voucher2	• Supplier2
Customer3	Order3	Voucher3	• Supplier3
	Order4	Voucher4	
	Order5	Voucher5	
	Order6		

Task	Expected response	Max mark	Additional guidance
2c	<ul style="list-style-type: none"> ♦ fields and alias <ul style="list-style-type: none"> — firstname, surname, voucherID — calculation as Amount of Money Spent on Voucher £ ♦ joins and category criteria ♦ wildcard 	3	Accept alternatives of escape room wildcard that produce the correct output. Sample query shown below.
	<pre>SELECT firstname, surname, Voucher.voucherID, (quantityPurchased * price£) AS [Amount of Money Spent on Voucher £] FROM Customer, CustomerOrder, Voucher WHERE Customer.custID = CustomerOrder.custID AND Voucher.voucherID = CustomerOrder.voucherID and category = "Adventure" AND voucherName LIKE "*escape room*";</pre>		
2d	<ul style="list-style-type: none"> ♦ sum to find number of vouchers already purchased ♦ calculation to subtract quantity purchased from quantity available ♦ correct fields, tables and aliases ♦ joins and criteria 	4	Combined query shown below.
	<pre>SELECT Voucher.voucherID, supplierName, voucherName, (quantityAvailable - SUM(quantityPurchased)) AS [Still Available] FROM CustomerOrder, Voucher, Supplier WHERE CustomerOrder.voucherID = Voucher.voucherID AND Voucher.supplierCode = Supplier.supplierCode AND Voucher.voucherID = "V543" GROUP BY Voucher.voucherID,supplierName, voucherName, quantityAvailable;</pre>		
2e	<ul style="list-style-type: none"> ♦ change aggregate function to Count(*) ♦ ORDER by aggregate function descending 	2	
2f	<ul style="list-style-type: none"> ♦ expiry date might be in the past ♦ invalid category entered ♦ invalid number of vouchers/price 	1	Award 1 mark for any bullet.

Task – web design and development

Task	Expected response	Max mark	Additional guidance
3a	<ul style="list-style-type: none"> ◆ Home with links to History, Animals, Events pages on level 1 ◆ Profile, Donate and Adopt on level 2 of Animals page ◆ Excursions and Parties level 2 of Events page 	3	<p>Candidates can use alternative names for level 2 pages.</p> <p>See below.</p>
	<pre> graph TD HP[Home Page] --- Bar subgraph Bar [] direction LR H[History] A[Animals] E[Events] end A --- P[Profile] A --- D[Donate] A --- Ad[Adopt] E --- Part[Parties] E --- Exc[Excursions] </pre>		
3b	<ul style="list-style-type: none"> ◆ edited HTML pages: <ul style="list-style-type: none"> — list of hyperlinks — associated page names ◆ edited colours (CSS): <ul style="list-style-type: none"> — hover text colour (Crimson/#DC143C) — hover background colour (White) — text colour (White) — background colour (Crimson/#DC143C) ◆ layout of navigation bar (CSS) <ul style="list-style-type: none"> — list style type: none — float left — centre-align text — 8px padding within the links 	3	

Task	Expected response	Max mark	Additional guidance
3c	<ul style="list-style-type: none"> ♦ two sections created with: <ul style="list-style-type: none"> — white backgrounds — width 432 pixels — height 1040 pixels ♦ 10px margin between the two sections 	2	Accept any solution that creates a 10px margin between the two sections.
3d	<ul style="list-style-type: none"> ♦ sections hidden when page first loads ♦ IDs added ♦ ID revealed when an image is clicked 	3	<p>Marks should be awarded from CSS and HTML code, not the printout of web page as viewed in a browser.</p> <p>Candidates must amend the HTML file for the first two marks.</p> <p>Candidates can use inline JavaScript or call a function to display the hidden sections.</p>
3e	<ul style="list-style-type: none"> ♦ the drop-down list is not set to required ♦ the drop-down list allows multiple options to be selected 	2	
3f	<ul style="list-style-type: none"> ♦ no video included in profile page ♦ comment on fitness for purpose in relation to JavaScript code 	2	This comment should be consistent with the candidate's own code for task 3d.

[END OF MARKING INSTRUCTIONS]

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. Ensure your name and candidate number is included on all your evidence.

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

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

Task 1 – software design and development

Evidence		Tick
1a	Completed task sheet identifying the functional requirements	
1b	Completed task sheet showing the data flow for the program	
1c(i)	Printout of your completed program code	
	Printout of your output, showing athlete(s) with the maximum number of jumps	
	Printout of your CSV file containing the entry ID and bib values	
1c(ii)	Printout of your edited program code	
	Printout of the display produced by the new sub-program	
1d	Completed task sheet showing lines of code and value of jumps	
1e	Completed task sheet evaluating fitness for purpose and maintainability	

Task 2 – database design and development

Evidence		Tick
2a	Completed task sheet identifying two functional requirements	
2b	Completed entity-occurrence diagram showing the entity names, sample instances and association between instances	
2c	Printout of the SQL statement(s) to find customers who have purchased vouchers for an escape room from the 'Adventure' category	
	Printout of the output produced	
2d	Printout of the SQL statement(s) to find how many vouchers are available for voucher ID V543	
	Printout of the output produced	
2e	Printout of the amended SQL statement to produce the expected output	
2f	Completed task sheet evaluating one potential problem	

Task 3 – web design and development

Evidence		Tick
3a	Completed task sheet showing a design of a multi-level navigation structure	
3b	Printout of edited 'styles.css' file and 'home.html' file	
3c	Printout of edited 'styles.css' file and 'animals.html' file	
	Printout of 'Animals' page as viewed in a browser	
3d	Printout of your edited code	
	Printouts of the 'Profile' page as viewed in a browser	
3e	Completed task sheet describing validation issues with the donation amount element	
3f	Completed task sheet evaluating the fitness for purpose of the 'Profile' page	

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

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

Task 1: software design and development

Problem description

Thirty athletes have qualified for the final of the Scottish Jumping Jacks competition. Qualifying events were held at four locations, where each athlete performed as many jumping jacks as they could in 1 minute.

The following details are stored in a CSV file, for each athlete who qualified for the final:

- ◆ Entry ID
- ◆ Qualifying location
- ◆ Forename
- ◆ Surname
- ◆ Number of jumping jacks completed



Purpose

A program is required to read each of the athlete's data from the existing CSV file. A bib value for each of the finalists will be generated and stored in a new CSV file along with the entry ID. The program will also display the full name of the athlete(s) who completed the highest number of jumping jacks.

An example of the bib value generated for the athlete Daniel Currie, who qualified at the Inverness event, is shown below. Note that 73 is the ASCII value of 'I', the first character of Inverness.



Assumptions

- ◆ The CSV file has data for thirty athletes, is formatted correctly and is error-free.
- ◆ Each line of the CSV file stores the entry ID, qualifying location, forename, surname and number of jumping jacks completed at qualification, as shown below:

```
f01,Motherwell,Ellie,McAninch,85
f02,Inverness,Ayat,Whyte,83
f03,Kirkcaldy,Simra,Zamora,42
f04,Motherwell,Dai,Nguyen,37
f05,Coatbridge,Max,Hughes,113
...
```

Task 1: software design and development (part A)

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

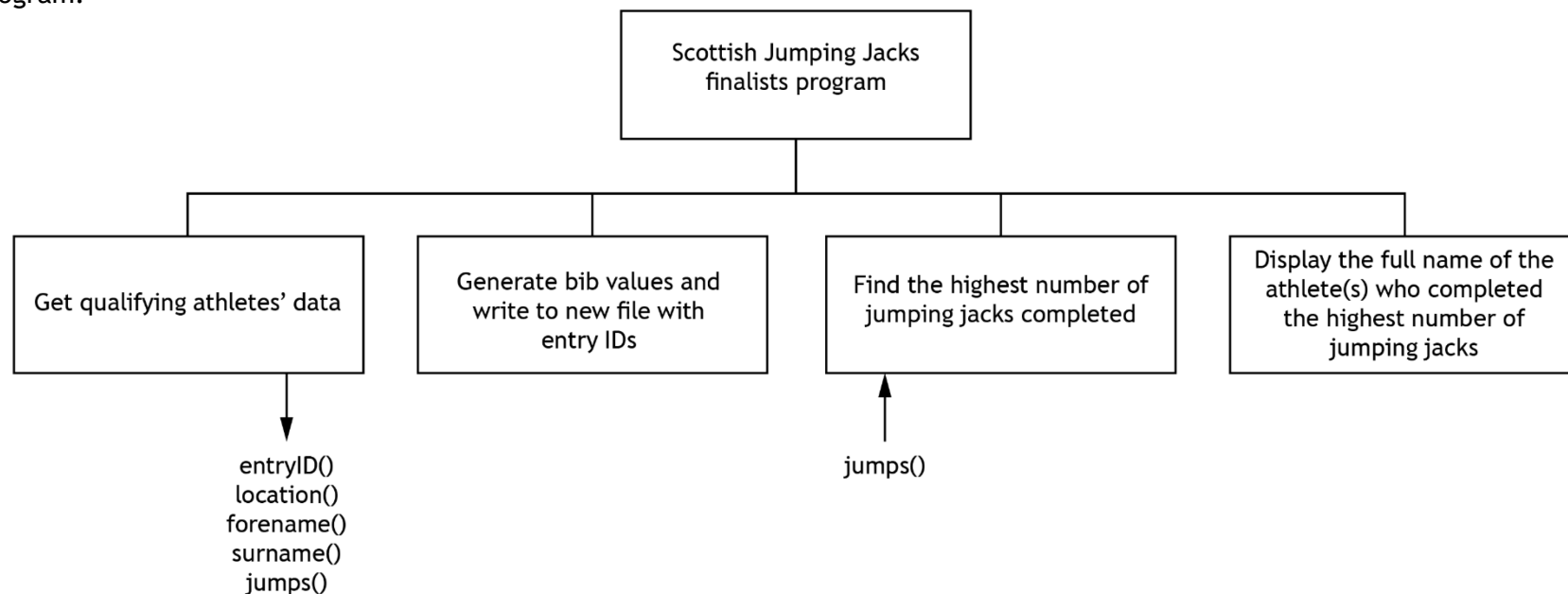
(2 marks)

Input(s)
Process(es)
Output(s) <ul style="list-style-type: none">◆ Store entry ID and bib values in file◆ Full name of athlete(s)

Candidate name_____ Candidate number_____

1b Each athlete at the national final will be identified by their bib value, as shown previously.

A top-level design of the main steps of the program is shown below. Data read from the CSV file is stored in parallel arrays in the program.



Complete the diagram to show the data flow for the program. Your completed diagram should include:

- ♦ the required data (arrays and variables)
- ♦ arrows to indicate the flow of the data

(3 marks)

- ♦ 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 name_____ Candidate number_____

Task 1: software design and development (part B)

The program design is shown below.

Program top-level design (pseudocode)

- | | |
|---|---|
| 1. Get qualifying athletes' data | OUT: entryID(), location(), forename(),
surname(), jumps() |
| 2. Generate bib values and write to new file with
entry IDs | IN: entryID(), location(), forename(), surname() |
| 3. Find the highest number of jumping jacks
completed | IN: jumps(),
OUT: maxJumps |
| 4. Display the full name of the athlete(s) who
completed the highest number of jumping jacks | IN: maxJumps, forename(), surname(), jumps() |

Refinements

- 1.1 Open athletes.csv file
- 1.2 Loop for thirty athletes
- 1.3 Store entryID, location, forename, surname, jumps for athlete in parallel arrays
- 1.4 End loop
- 1.5 Close athletes.csv file

- 2.1 Create bibValues.csv file
- 2.2 Loop for thirty athletes
- 2.3 Set bibValue to first letter of forename & full surname & ASCII value of first letter of location
- 2.4 Write entryID and bibValue to file
- 2.5 End loop
- 2.6 Close bibValues.csv file

- 3.1 Set maximum jumps to the value stored in the first index of the jumps array
- 3.2 Start loop from second index to end of array
- 3.3 If the current number of jumps is more than maximum jumps then
- 3.4 Set maximum jumps to current number of jumps
- 3.5 End if
- 3.6 End loop
- 3.7 Return maximum jumps

- 4.1 Loop for thirty athletes
- 4.2 If current number of jumps equals maximum jumps then
- 4.3 Display forename and surname
- 4.4 End if
- 4.5 End loop

1c(i) Using the problem description and design, implement the program in a language of your choice. Your program should:

- ♦ be maintainable and modular
- ♦ use a function to find and return the maximum number of jumps
- ♦ follow the design and the refinements provided

(13 marks)

Print evidence of:

- ♦ your completed program code
- ♦ your output, showing athlete(s) with the maximum number of jumps
- ♦ your CSV file containing the entry ID and bib values

1c(ii) The location with the fewest number of athletes qualifying will host the next final.

A new sub-program is to be implemented to find the total number of athletes from each location in the final. An example of the output is shown below.

```
Coatbridge has 6 finalists
Inverness has 8 finalists
Kirkcaldy has 7 finalists
Motherwell has 9 finalists
```

Implement the additional sub-program.

(2 marks)

Print evidence of:

- ♦ your edited program code
- ♦ the display produced by the new sub-program

- 1d The function to find the maximum number of jumps is tested using the following test data.

jumps = [100,87,102,108,95]

A watchpoint is placed on the variable storing the maximum number of jumps.

Complete the table below by entering:

- ♦ the lines of code from your program where the watchpoint is triggered
- ♦ the value of the maximum number of jumps variable when the watchpoint is triggered

(3 marks)

Line of code from your program	Value of the maximum number of jumps

- 1e With reference to your program code, evaluate:

(2 marks)

The fitness for purpose of the function to generate bib values
The maintainability of your program, referring to modularity

Candidate name_____ Candidate number_____

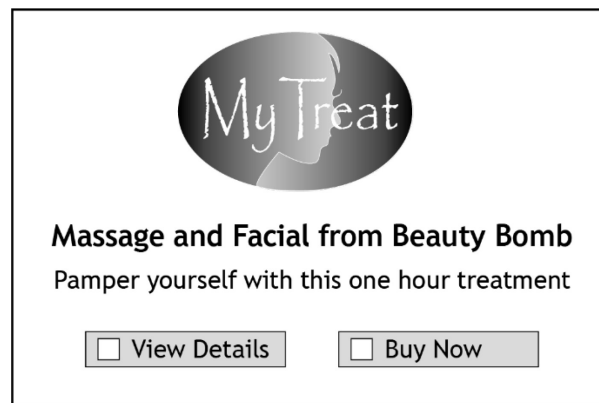
Task 2: database design and development (part A)

MyTreat is a company that is developing a database of voucher offers from a range of suppliers.

Suppliers will offer vouchers in the following categories: Food, Beauty, Adventure and Family. A supplier may have more than one active offer in the database at a time and can sell many vouchers for each offer.

Customers registered with MyTreat will be able to purchase multiple voucher offers. Vouchers will be sent to the customer's email address.

Vouchers will be promoted on the MyTreat website. An example is shown below.



The database development team at MyTreat asked the staff how they would like to use the database. Some of the comments are shown below.

We want to offer bonuses to customers who spend the most money. I need to know how much money customers are spending.

I need to give suppliers regular updates on how many of their voucher offers have been sold.

Customers often ask us to recommend offers. I would like to be able to look for specific categories of offer to send to the customer.

It's my job to analyse the voucher offers to see which are most popular. I want to be able to see our best and worst sellers.

I'm responsible for GDPR so I need to be able to keep all the customer and supplier data accurate and up to date.

- 2a Using the information gathered from the staff comments, create two functional requirements of the database.

(2 marks)

Functional requirement 1

Functional requirement 2

Candidate name_____ Candidate number_____

2b The following tables have sample data showing:

- ♦ vouchers included in each customer order
- ♦ customers who made orders
- ♦ supplier of each voucher

Voucher	CustomerOrder
Voucher2	Order1
Voucher4	Order2
Voucher5	Order3
Voucher4	Order4
Voucher1	Order5
Voucher3	Order6

Customer	CustomerOrder
Customer1	Order1
Customer3	Order2
Customer1	Order3
Customer2	Order4
Customer1	Order5
Customer2	Order6

Voucher	Supplier
Voucher1	Supplier3
Voucher2	Supplier1
Voucher3	Supplier3
Voucher4	Supplier2
Voucher5	Supplier2

Using the information from the sample data, complete the blank entity-occurrence diagram on the following page by:

- ♦ naming the entities
- ♦ completing the sample instances provided for each entity
- ♦ showing the association between those instances

(3 marks)

Entity-occurrence diagram

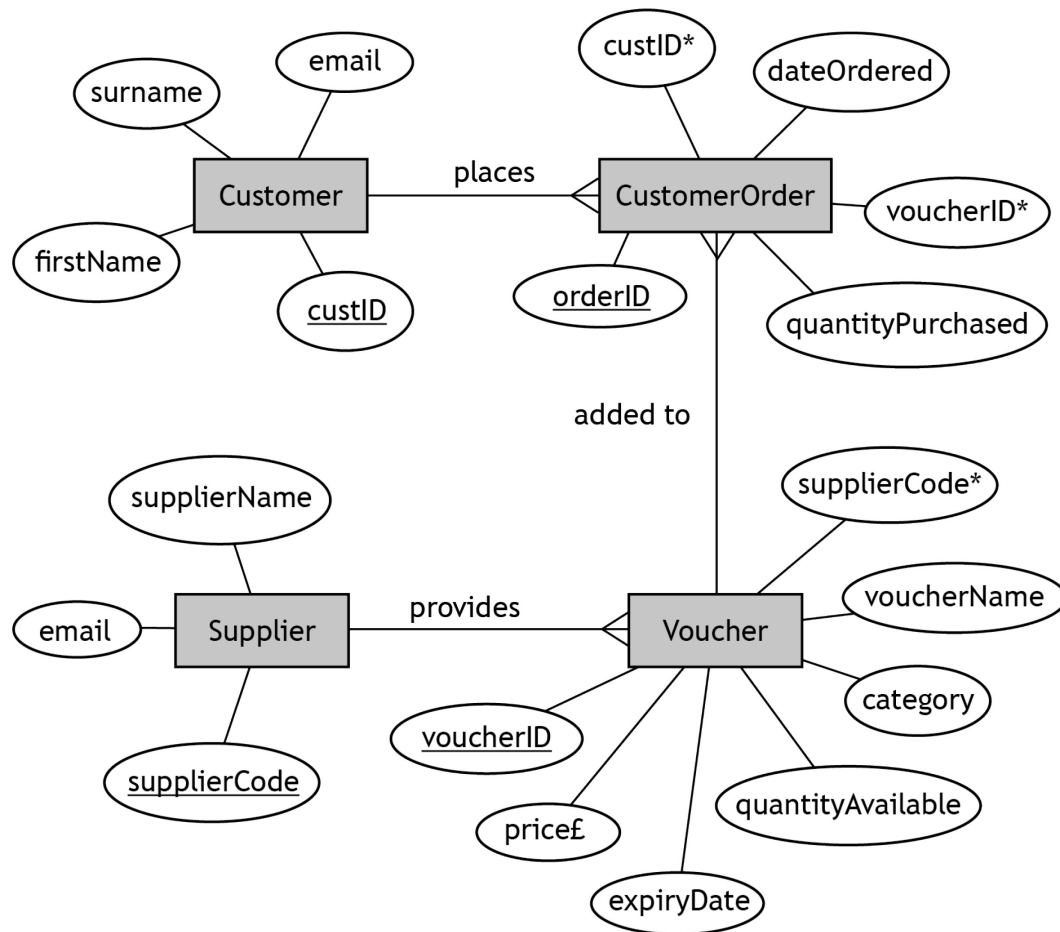
Entity Names			

- ♦ 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 2: database design and development (part B)

The entity-relationship diagram for the MyTreat database is shown below.



The design is then implemented.

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

- 2c MyTreat would like to know how much money is being spent on the different types of escape room vouchers.

A query is required to find customers who have purchased vouchers for an escape room from the 'Adventure' category. The output should include the amount of money spent by the customer on the voucher.

Implement the SQL statement to produce the following output.

(3 marks)

firstName	surname	voucherID	Amount of Money Spent on Voucher £
Neville	Wilson	V368	32
Bailey	Donald	V369	80
Aziah	Moqsud	V890	172
Chukka	Radebe	V890	86
Becky	Bennett	V890	344

- 2d MyTreat would like to know how many vouchers are still available for voucher ID V543.

Implement the SQL statement(s) to produce the following output.

(4 marks)

voucherID	supplierName	voucherName	Still Available
V543	SkatePark	Skate park and lunch	198

For 2c and 2d print evidence of:

- ♦ the implemented SQL statement(s)
- ♦ the output produced

- 2e A query is designed to find the number of customers who bought a voucher from the 'Family' category that costs less than £15.00.

The expected output from the query is shown below.

supplierName	voucherName	price£	Number of Customers
Cuddle World	Teady Bears' Picnic	10.00	4
WonderPlay	Trampoline	4.00	3
WonderPlay	Softplay and lunch for 2	12.00	2
Family Fun Club	Softplay and cake	6.00	1

The SQL statement shown below was implemented.

```
SELECT Supplier.supplierName, Voucher.voucherName,  
Voucher.price£, Sum(Voucher.price£) AS [Number of Customers]  
FROM CustomerOrder, Supplier, Voucher  
WHERE CustomerOrder.voucherID=Voucher.voucherID AND  
Supplier.supplierCode=Voucher.supplierCode AND  
Voucher.price£<15 AND Voucher.category="Family"  
GROUP BY Supplier.supplierName, Voucher.voucherName,  
Voucher.price£;
```


The query to test the above SQL statement is provided with the database.

Test the SQL statement by running the query.

Amend the query to produce the expected output as shown above.

(2 marks)

Print evidence of the amended SQL statement.

2f The `Voucher` table has no validation.

Evaluate one potential problem that may occur when adding new data to this table.

(1 mark)

--

Candidate name_____ Candidate number_____

Task 3: web design and development

Springfield Zoo wants to create a website to provide information about the zoo and to attract new visitors.

They commissioned a web developer to create the website.

After discussions with users, the following end-user requirements and functional requirements were identified.

End-user requirements

Users want to:

- ◆ see pictures of the animals in the zoo
- ◆ watch real-time footage of animals
- ◆ read a daily animal profile
- ◆ apply to adopt an animal
- ◆ donate money to the zoo
- ◆ find information on educational excursions and booking birthday parties
- ◆ read about the history of the zoo
- ◆ find contact information and location details

Functional requirements

The 'Home', 'History', 'Animals' and 'Events' pages should contain:

- text displaying relevant information to the user
- appropriate images
- contact information and location details

The 'Profile' page should:

- display an introductory paragraph and a fun fact about the daily animal, when an image button is selected
- have a video showing the animals in real-time

The 'Donate' page should:

- contain a form that allows users to submit a donation to the zoo

The 'Adopt' page should:

- provide users with information on how to adopt a tiger
- allow users to submit their email address to request an information pack

The 'Parties' page should:

- contain information about booking a birthday party at the zoo

The 'Excursions' page should:

- provide information on how to book educational excursions

Task 3: web design and development (part A)

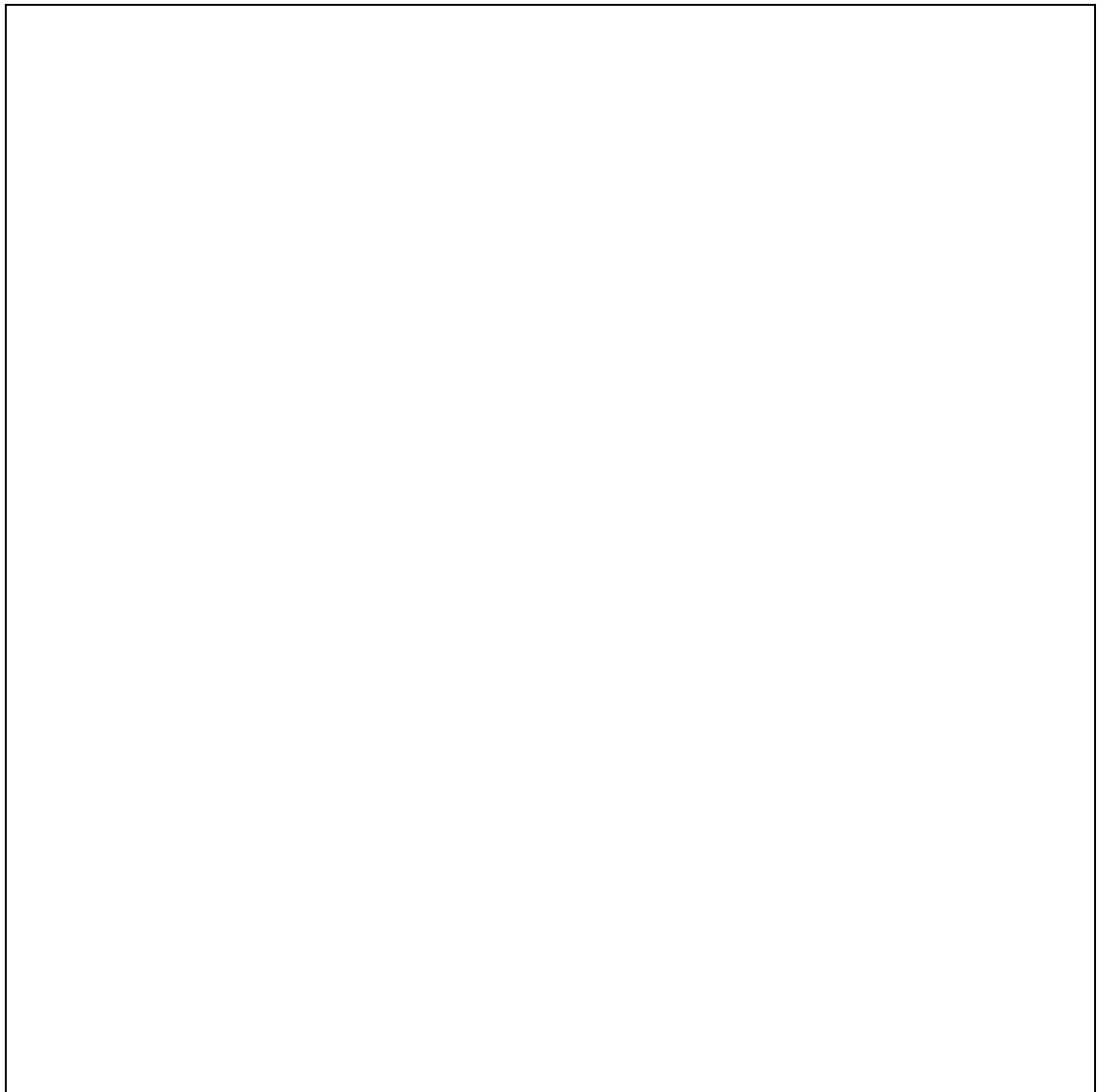
- 3a The website will have a multi-level navigation structure. This will have a 'Home' page with a horizontal navigation bar that links three main web pages: 'History', 'Animals' and 'Events'.

The 'Animals' page will have links to three sub-pages: Profile, Donate and Adopt.

The 'Events' page will have links to two sub-pages: Parties and Excursions.

Design a multi-level navigation structure for this website, clearly showing the navigation bar and associated pages.

(3 marks)

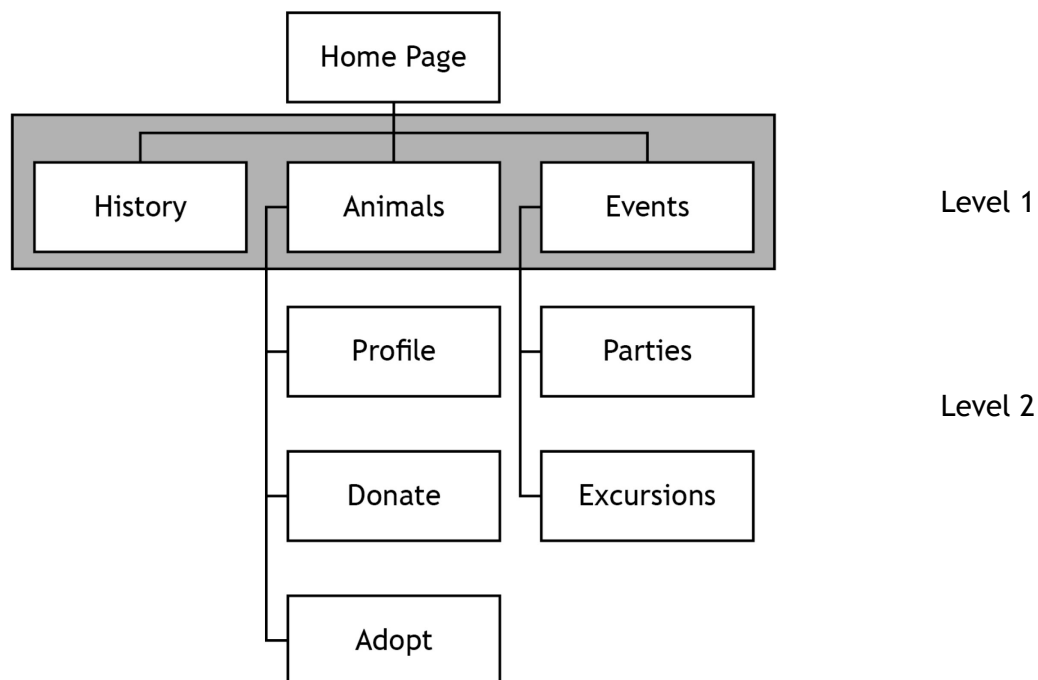


- ♦ Check your design carefully. You cannot return to part A after you hand it in.
- ♦ When you are ready, hand part A to your teacher or lecturer and collect part B.

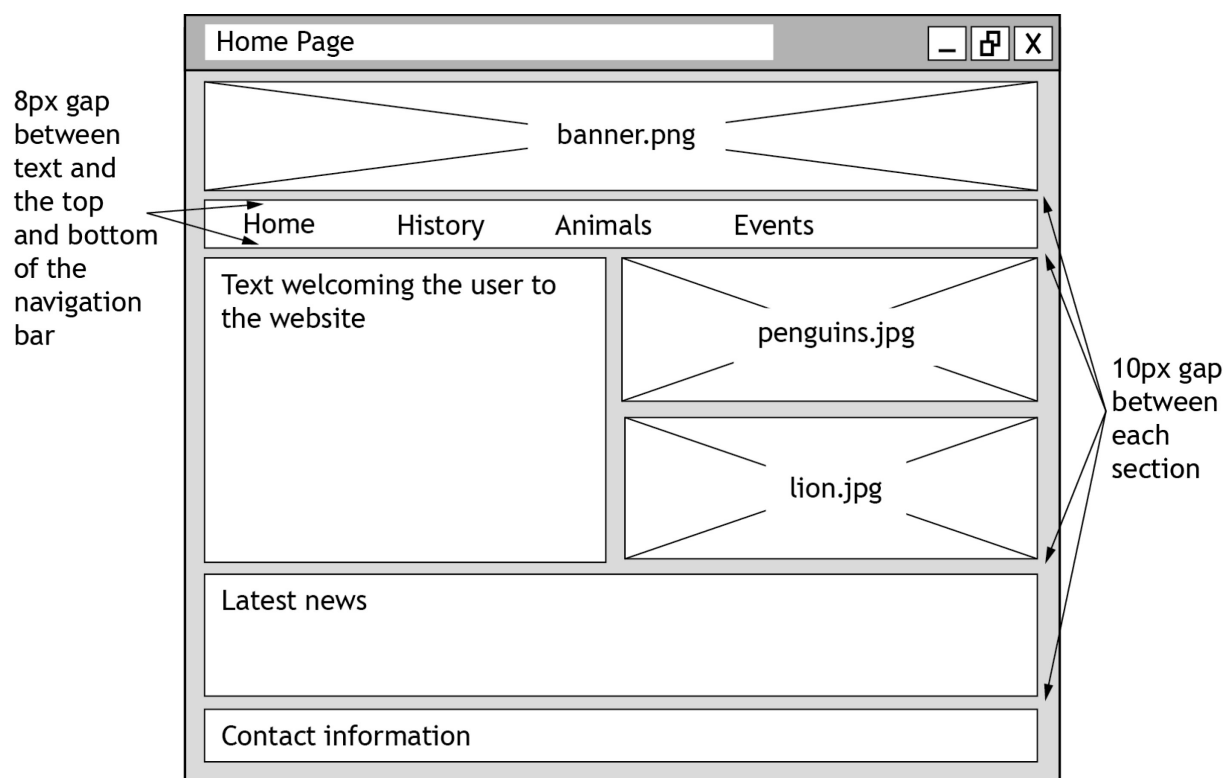
Candidate name_____ Candidate number_____

Task 3: web design and development (part B)

Below is the navigation structure for the Springfield Zoo website.



Below is a wireframe design for the home page of the website.



Your teacher or lecturer will provide you with the incomplete website for Springfield Zoo.

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

3b Locate the 'Navigation Bar' section in the 'styles.css' file.

Edit the CSS and HTML files to match the wireframe. Implement the navigation bar as shown below to appear on the 'Home', 'History', 'Animal' and 'Events' pages of the website.

Home	History	Animals	Events
----------------------	-------------------------	-------------------------	------------------------

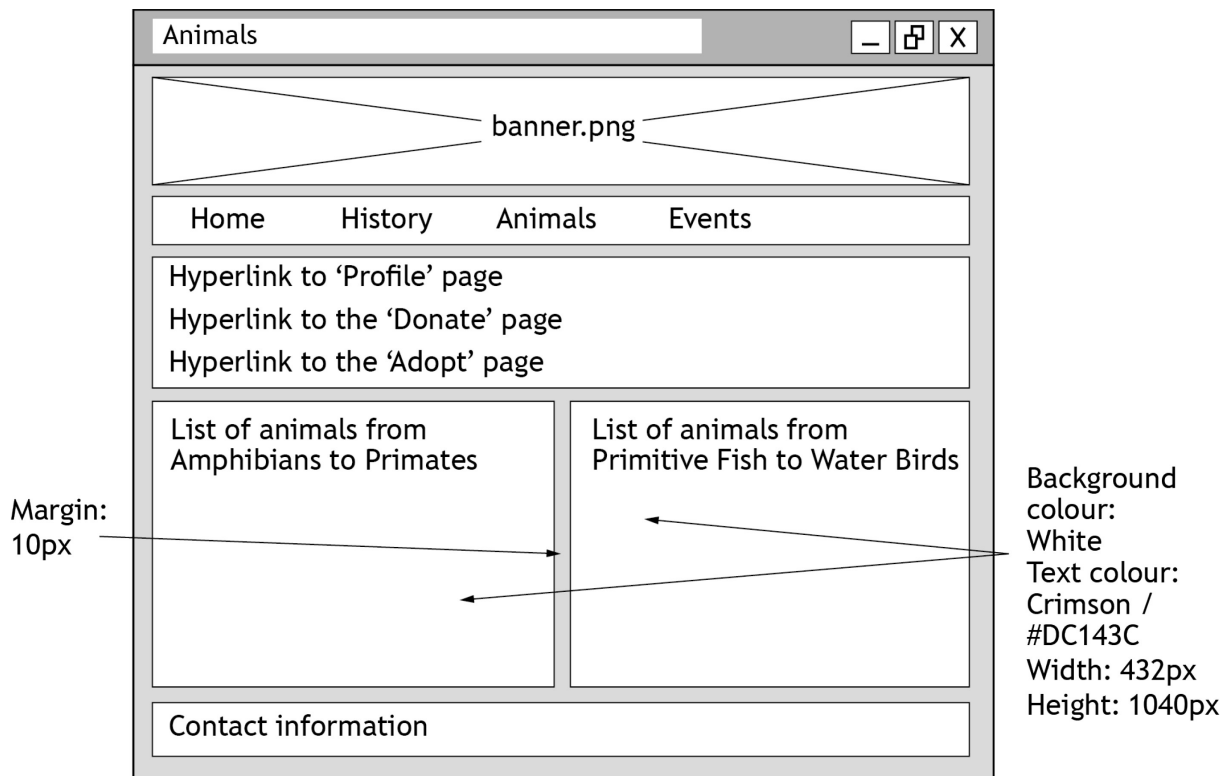
Background colour:	Crimson / #DC143C
Text colour:	White / #FFFFFF
Hover background colour:	White / #FFFFFF
Hover text colour:	Crimson / #DC143C

(3 marks)

Print evidence of:

- ♦ the edited 'styles.css' file showing the CSS for navigation bar
- ♦ the edited 'home.html' file

3c The wireframe design for the 'Animals' page is shown below.



Open the 'Animals' page in a browser.

The long list of animals has not been divided into two sections displayed side-by-side, as shown in the wireframe above.

Edit the 'animals.html' file and 'styles.css' file in a suitable editor to implement the wireframe.

(2 marks)

Print evidence of:

- ◆ the edited 'styles.css' file and 'animals.html' file
- ◆ the 'Animals' page as viewed in a browser

3d Open the 'profile.html' in a browser.

When the page first loads, it will display an image of an animal with introductory text. There will also be three 'Fun fact' buttons and three sections, each with the text for a fun fact.

Edit the code to:

- ◆ hide all of the sections of text below the buttons when the page first loads
- ◆ show the first fun fact section when the 'Fun fact 1' button is selected
- ◆ show the second fun fact section when the 'Fun fact 2' button is selected
- ◆ show the third fun fact section when the 'Fun fact 3' button is selected
- ◆ ensure that only one of the fun fact sections is visible at any time

(3 marks)

Print evidence of:

- ◆ your edited code
- ◆ the 'Profile' page as viewed in a browser when:
 - first loaded
 - the 'Fun fact 1' button is selected
 - the 'Fun fact 2' button is selected
 - the 'Fun fact 3' button is selected

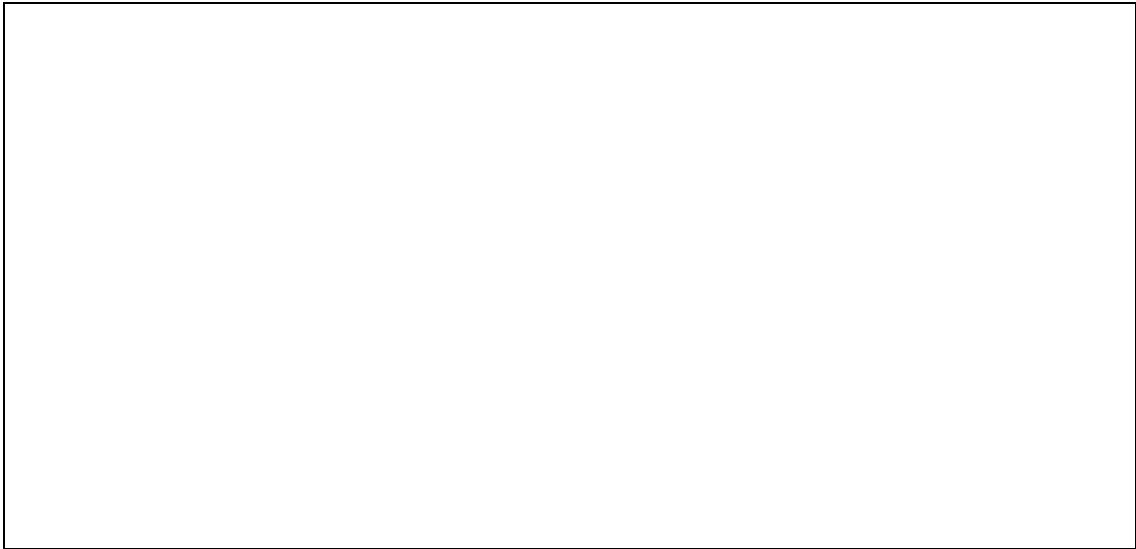
- 3e The 'Donate' page contains a form that users can fill in if they want to donate money to the zoo.

Open the 'donate.html' page in a suitable editor and examine the form code carefully.

During usability testing, validation issues were identified with the donation amount element.

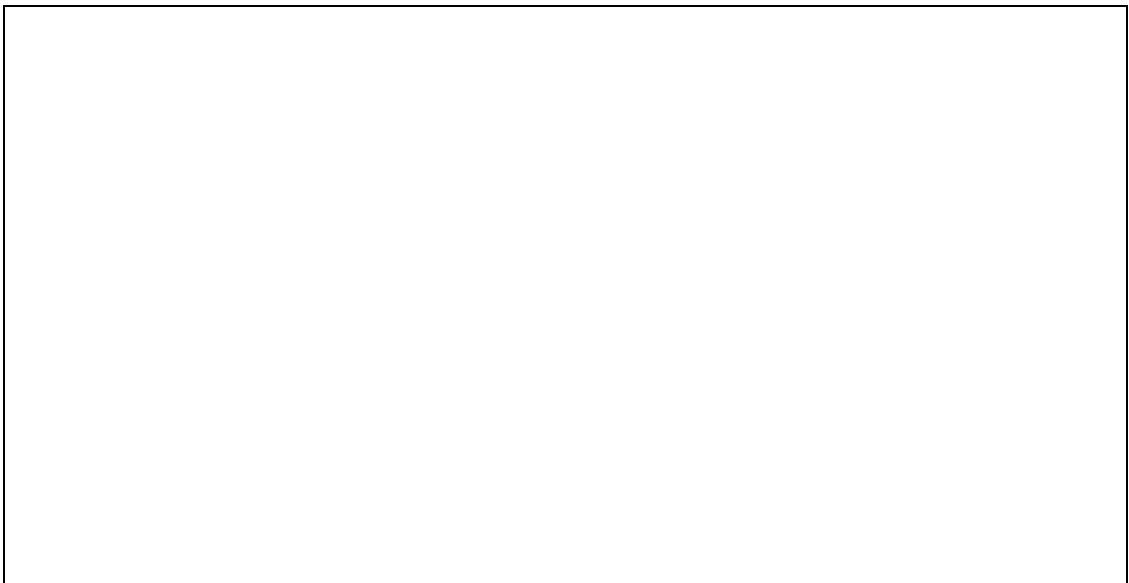
Describe the issues with the donation amount element.

(2 marks)



- 3f Evaluate the fitness for purpose of the 'Profile' page.

(2 marks)



Candidate name_____ Candidate number_____

Copyright acknowledgements

Electronic files:

- ◆ Yumeee/Shutterstock.com
- ◆ Nataalia85/Shutterstock.com
- ◆ Ruth Black/Shutterstock.com
- ◆ veronchick_84/Shutterstock.com
- ◆ Trong Nguyen/Shutterstock.com
- ◆ Erwin Niemand/Shutterstock.com
- ◆ Ricardo Reitmeyer/Shutterstock.com
- ◆ Pressmaster/Shutterstock.com
- ◆ Rosie Parsons/Shutterstock.com
- ◆ Kotomiti Okuma/Shutterstock.com
- ◆ Alan Jeffery/Shutterstock.com

Administrative information

Published: August 2023 (version 1.0)

History of changes

Version	Description of change	Date

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 2023