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 2018-19 only.

The information in this publication may be reproduced in support of SQA qualifications only on a non-commercial basis. If it is reproduced, SQA must be clearly acknowledged as the source. If it is to be reproduced for any other purpose, written permission must be obtained from permissions@sqa.org.uk.

Where this publication includes material for which SQA does not own the copyright, this material must only be reproduced on a non-commercial basis for the purposes of instruction in an educational establishment. If it is to be reproduced for any other purpose, it is the user's responsibility to obtain the necessary copyright clearance from the copyright owner. The acknowledgements page lists the owners of copyright items that are not owned by SQA.

This edition: January 2019 (version 1.0)

© Scottish Qualifications Authority 2019

Contents

Introduction 1

Instructions for teachers and lecturers 2

Instructions for candidates 7

Introduction

This document contains instructions for teachers and lecturers, and instructions for candidates for the Higher Computing Science assignment. It must be read 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 assessment applies to the assignment for Higher Computing Science for the academic session 2018-19.

The task is valid for session 2018-19 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, 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.**

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: 2018-19

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 (Flight Booking) 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:

- customer.csv
- booking.csv
- flight.csv
- route.csv

Specific instructions for database setup
The Flight Booking 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.

Flight Booking database			
Customer	**Booking**	**Flight**	**Route**
customerID	bookingNo	flightID	routeID
forename	adultTicket	departureDate	departFrom
surname	childTicket	departureTime	arriveAt
street	concessionTicket	arrivalDate	midStopOne
town	customerID*	arrivalTime	midStopTwo
postcode	flightID*	capacity	
		routeID*	

Task 2 – 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 2 — part B is a separate section. This ensures that candidates do not access part A and change their responses. Candidates should still have access to the problem description page during part B.

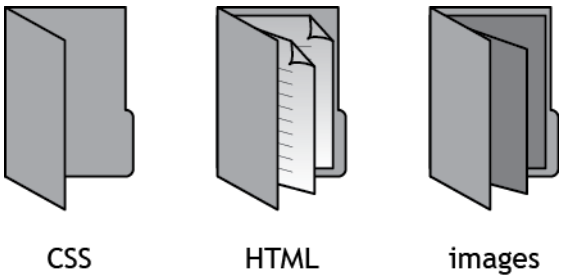
Give the following data file to candidates:

- members.txt

Task 3

The following compressed file has been provided (playingCards.zip).

When unpacked this contains the CSS, HTML and images candidates need to complete this task. These files should remain in the folders shown below and should 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 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 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/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 1 sheet showing the entity names, instances and associated relationships on the entity-occurence diagram	☐
Part B		
1b (i)	SQL statement implemented to calculate tax for booking	☐
	Printout of output	☐
1b (ii)	SQL statements implemented to identify the customer(s) who made a booking with the greatest number of children	☐
	Printout of output	☐
1c	Completed task 1 sheet showing evaluation of potential problems	☐
Task 2	**Evidence**	
Part A		
2a	Completed task 2 sheet identifying functional requirements	☐
2b	Completed task 2 sheet showing completed data flow design	☐
Part B		
2c (i)	Printout of your program code and the results.txt file	☐
2c (ii)	Printout of your edited program code and evidence of new data being added to results.txt file	☐
2d	Completed task 2 sheet showing trace table	☐
2e	Completed task 2 sheet showing evaluation	☐

Task 3	Evidence	
3a	Completed task 3 sheet showing functional requirements	☐
3b (i)	Printout of HTML (history.html) and CSS (styles.css) showing form code	☐
	Printout of 'History' page as viewed in browser	☐
3b (ii)	Printout of HTML (patience.html) showing form code	☐
	Printout of 'Patience' pages as viewed in browser	☐
3c	Completed task 3 sheet showing description of comprehensive test plan	☐

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, 2a)
- Clearly display your name and candidate number on each printout

Task 1: database design and development (part A)

A small passenger airline wishes to create a database as part of a new online booking system. The database will be required to store the following information:

- customer details
- bookings made by customers
- flight details
- scheduled routes

1a The following tables include sample data showing:

- bookings on some flights
- customers that made those bookings
- routes taken by each flight

Flight	Booking
Flight1	Booking1
Flight4	Booking2
Flight3	Booking3
Flight2	Booking4
Flight5	Booking5
Flight1	Booking6

Customer	Booking
Cust1	Booking1
Cust2	Booking2
Cust1	Booking3
Cust3	Booking4
Cust1	Booking5
Cust3	Booking6

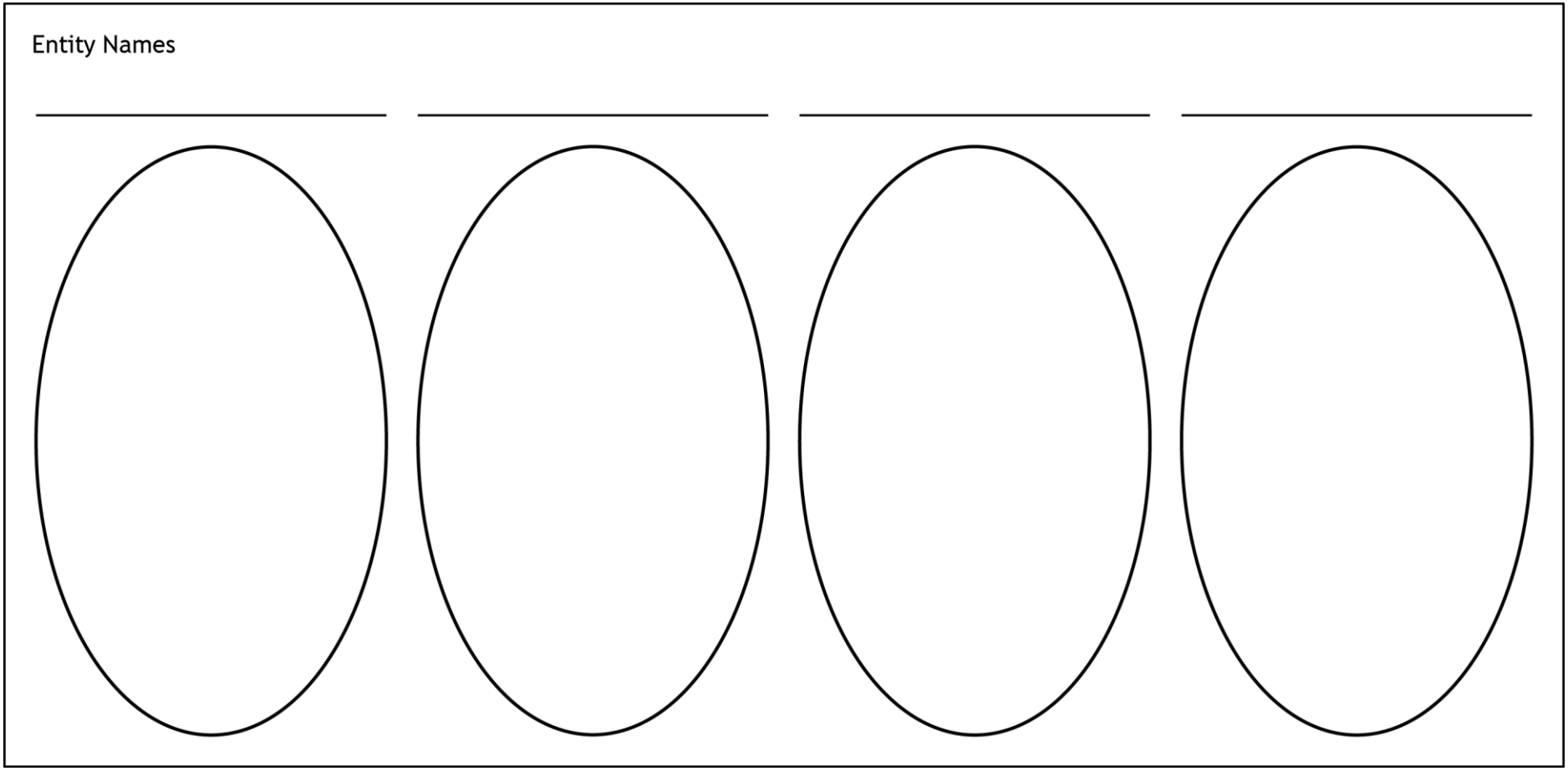
Flight	Route
Flight1	Route1
Flight2	Route1
Flight3	Route3
Flight4	Route3
Flight5	Route2

Using the information provided by the sample data, complete the blank entity-occurrence diagram provided by:

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

(3 marks)

Entity-occurrence diagram

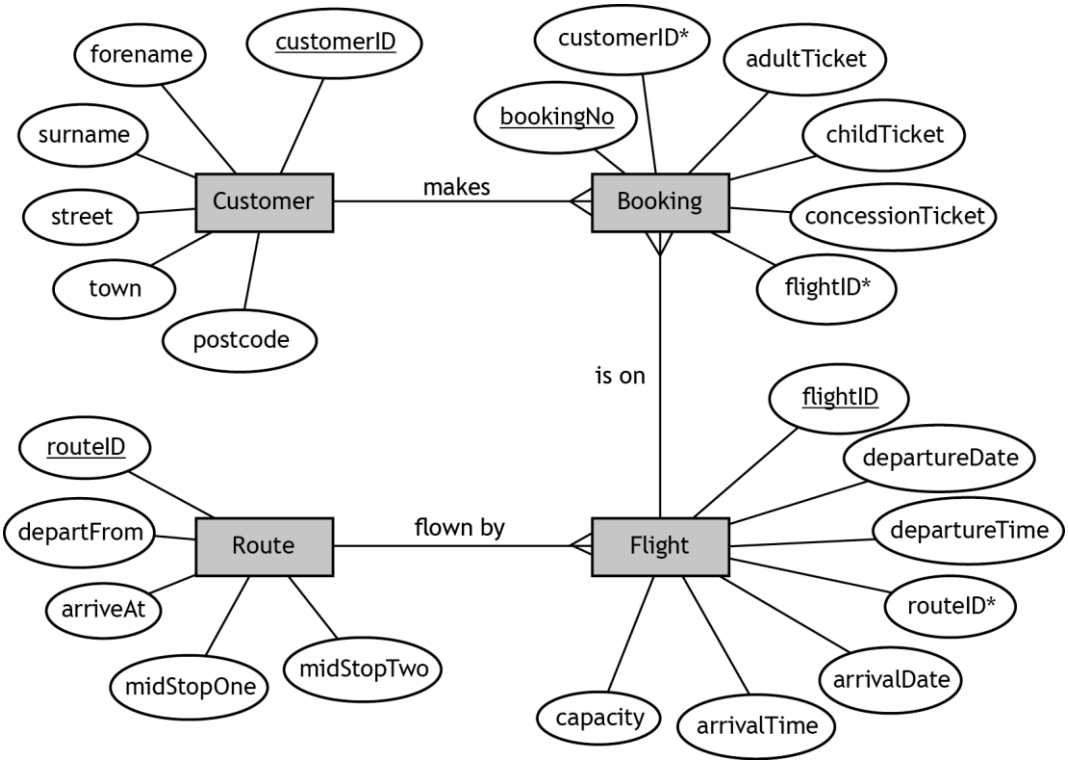


- Check your answers carefully, as you cannot return to part A after you hand it in.
- When you are ready, hand part A to your teacher or lecturer and collect part B.

Candidate name_____ Candidate number_____

Task 1: database design and development (part B)

Following further analysis the entity-relationship diagram below is created.



This design is then implemented.

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

Flight Booking Database			
Customer	**Booking**	**Flight**	**Route**
customerID	bookingNo	flightID	routeID
forename	adultTicket	departureDate	departFrom
surname	childTicket	departureTime	arriveAt
street	concessionTicket	arrivalDate	midStopOne
town	customerID*	arrivalTime	midStopTwo
postcode	flightID*	capacity	
		routeID*	

1b(i) John Smith, Customer ID - GR01932, has asked for a copy of the tax he has paid on flight QH182. The tax for a booking is calculated as follows:

- adults pay £5.50
- children pay £2.00
- concessions pay £1.50

Implement the SQL statement that will produce an output with the headings.

forename	surname	Tax (£)

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

(3 marks)

1b(ii) The airline wishes to identify the customer(s) who made a booking with the greatest number of children.

Implement two SQL statements that will find the forename and surname of the customer(s) who made a booking with the greatest number of children.

forename	surname

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

(4 marks)

1c The database has primary key fields but has no other validation. Evaluate two potential problems that may occur when adding new data to the Flight table.

Problem 1 **(1 mark)**
Problem 2 **(1 mark)**

Candidate name_____ Candidate number_____

Task 2: software design and development (part A)

Problem description

Once a year a walking club asks all its members to submit the total number of miles they have walked. The club collates this information in a text file. A section of the .txt file, which includes the names of members and the total miles they walked, is shown below.

```
…
Nikolai,Bryant,145.6
Susan,Brown,34.2
Teressa,Jones,398.5
…
```

The information in the file is then used to select prize winners. Prizes will be awarded for:

- the furthest distance walked
- any members who have walked more than 70% of the furthest distance

A program is required to read the data for each member from the text file. The program should use this data to find then display the furthest distance walked. The names of every member who has walked more than 70% of the furthest distance should be written to an empty text file so that the file can be printed out later.

2a Using the problem description, identify the functional requirements of the program.

(3 marks)

Input(s)
Process(es)
Output(s)

Candidate name_____ Candidate number_____

2b The top-level design for the program is shown below.
Complete the design to show the missing data flow in and out of each module.

(2 marks)

Top-level design main program modules		
Read members' data from file into array of records	IN	
	OUT	members(forename,surname,distance)
Find the furthest distance walked	IN	
	OUT	furthest
Display the furthest distance walked	IN	furthest
	OUT	
Write club prize winners to file	IN	members(forename,surname,distance)
	OUT	

- 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 2: software design and development (part B)

The design for the walking club program is shown below.

Program top-level design (pseudocode)

1.	Read members' data from file into array of records	(OUT: members(forename,surname,distance))
2.	Find the furthest distance walked	(IN: members(forename,surname,distance) OUT: furthest)
3.	Display the furthest distance walked	(IN: furthest)
4.	Write club prize winners to file	(IN: members(forename,surname,distance), furthest)

Refinements

```
1.1 Open members.txt file
1.2 Start loop for each member
1.3       Get member forename
1.4       Get member surname
1.5       Get member distance
1.6       Store member forename, surname and distance in members() array
1.7  End loop
1.8 Close members.txt file

2.1 Set furthest to distance stored for first member in members() array
2.2           Start fixed loop from second member to end of array
2.3                 If distance the current member walked is greater than furthest Then
2.4                       Set furthest to current distance
2.5                 End If
2.6           End fixed loop

4.1 Open results.txt file
4.2 Write "The prize winning members are:" to the results.txt file
4.3 Start loop for each record in members() array
4.4       If the distance the member walked is greater than 0.7*furthest
4.5           write the forename and surname to the results.txt file
4.6       End if
4.7 End loop
```

2c(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 furthest distance walked by a member
- use a procedure to display the furthest distance walked
- follow the design and the refinements provided

Print evidence of your program code and the results.txt file.

(13 marks)

2c(ii) The club wants to display the number of whole marathons each member has walked. A marathon is 26.22 miles long.

An example of the calculation required is:

If Nikolai Bryant walks 145.6 miles this equates to:
145.6/26.22 miles = 5.59115179 marathons
Nikolai has therefore walked 5 whole marathons.

In the above example 'Nikolai,Bryant,5' would be stored.

To accomplish this, further refinements of step 4 are added to the end of the current design.

```
4.8   Write "The number of whole marathons walked by each member is" to the results.txt file
4.9   Start loop for each record in members() array
4.10        Calculate the number of whole marathons walked
4.11        Write the forename, surname and the number of whole marathons to the results.txt file
4.12  End loop
4.13  Close the results.txt file
```

Using the above design, edit your original program code so that, for each member, the forename, surname and the number of whole marathons walked are stored in the results.txt file.

Run your program and print evidence of your edited program code and evidence that your program correctly stores the new data in the results.txt file.

(2 marks)

2d The function in step 2 is to be tested with the data shown below.

```
John,Davie,189.4
Susie,Small,14.6
Johnny,Atom,490.2
Wendy,Khan,512.5
Emir,Jones,170.3
```

Using the variable names and data structure names from your own code, create a trace table to find the furthest distance walked by the members in the test data.

(2 marks)

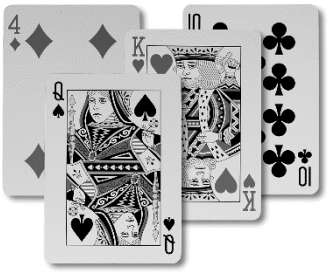
Candidate name_____ Candidate number_____

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

♦ the fitness for purpose of your program **(1 mark)**
♦ the maintainability of your program with reference to readability and modularity **(2 marks)**

Candidate name_____ Candidate number_____

Task 3: web design and development



The Card Foundation wishes to create a website to promote the use of playing cards.

They hire a web developer who asks them to fill in a client form. The completed client form is shown below.

Client Form Please answer the questions below as accurately as you can as we will use the information you provide to identify the requirements of your website.	
What is the main purpose of the website you would like us to build for you?	I would like you to create a website to highlight the work of the foundation. The foundation was set up to teach people about playing cards and encourage people to play more card games.
Are there any specific topics you would like the web pages to focus on?	I'd like to include a history of playing cards from their invention to how they came to Europe, and a bit about modern packs. A few pictures of packs of cards would be good. I'd like to list a few single player and multiplayer games along with the rules of each game.
Is there any media you would like to include?	I don't see the need for any video or sound. A few pictures scattered throughout the pages will do.
How do you feel about the site being interactive?	It would be good if there were a few things that visitors could click on to make it more interesting. Maybe the game rules could appear if you click on the name of the game.
Do you wish to gather any information from your users?	Yes. We offer a free pack of playing cards to anyone who contacts the foundation. It would be good if visitors to the site could request a pack of cards by submitting a few details about themselves.

3a State **two** functional requirements for the website.

Functional requirement 1 **(1 mark)**
Functional requirement 2 **(1 mark)**

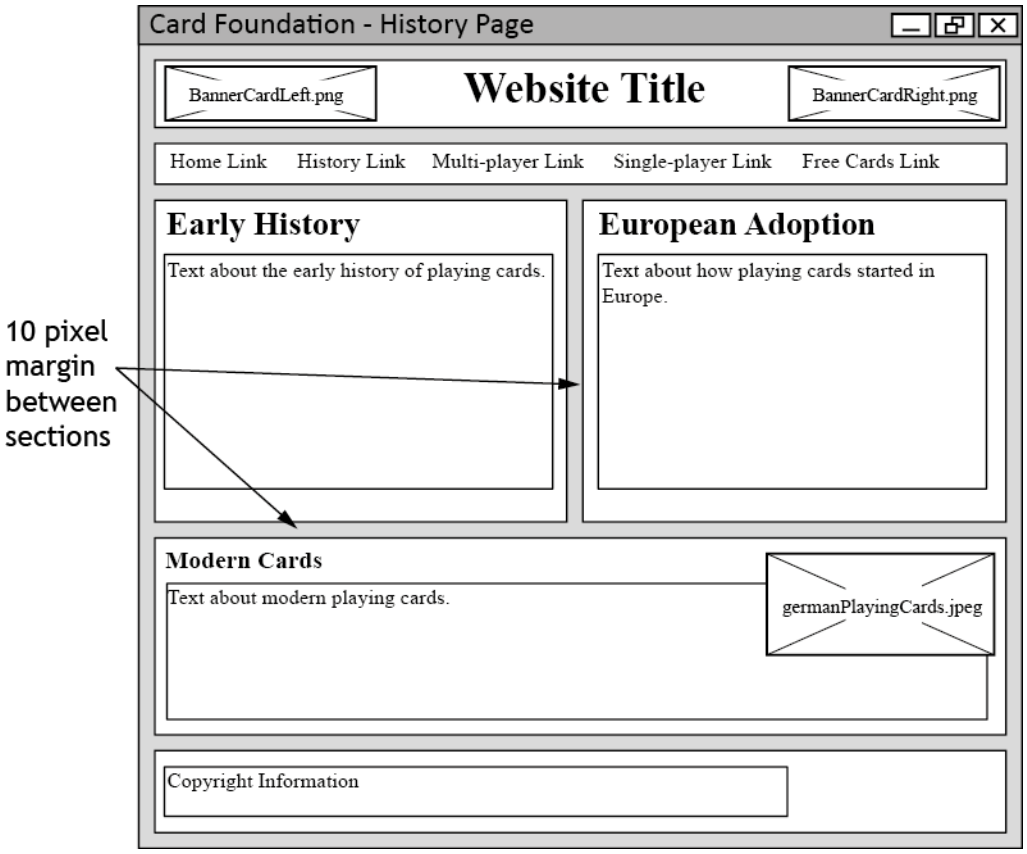
Candidate name_____ Candidate number_____

Your teacher or lecturer will provide you with a copy of the Card Foundation's incomplete website.

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

3b(i) A wireframe design for the 'History' page is shown below with the following requirements:

- 'Early History' and 'European Adoption' sections should be displayed side-by-side
- 'Modern Cards' section should fill the width of the page below 'Early History' and 'European Adoption'
- germanPlayingCards image should sit level with the 'Modern Cards' heading
- text about modern playing cards should wrap round this image

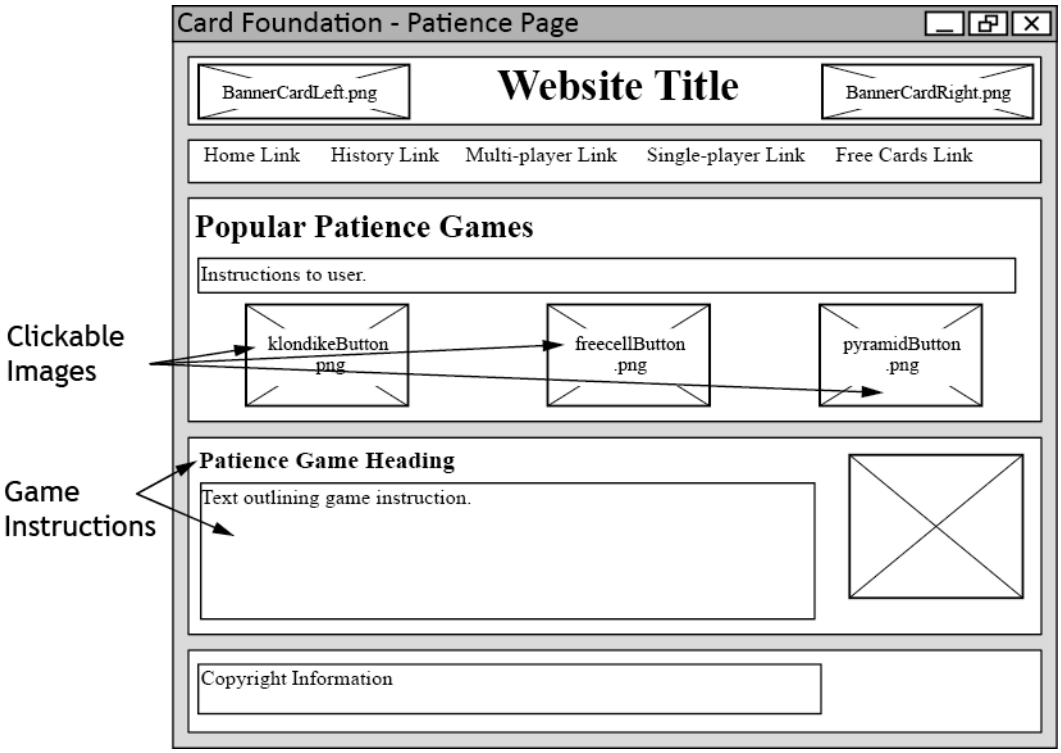


Open the 'history.html' and 'style.css' files in a suitable editor.
Edit the two files to create the layout shown above.

(4 marks)

3b(ii) The website contains a page which explains the rules of three single-player card games. When the page first loads none of the game instructions should be visible.

The wireframe design for this page is shown below.



When each of the images is clicked, the section including the Patience Game Heading, game instructions and photograph for that game should appear below.

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

Edit the file by adding events to:

- hide all the game instructions when the page first loads
- show only one section when the matching image is clicked

(4 marks)

Print evidence of:

- the edited history.html file
- the edited patience.html file
- the edited style.css file

Print evidence of:

- the 'History' page as viewed in a browser
- the 'Patience' page as viewed in a browser when first loaded
- the 'Patience' page when each of the three games is selected

3c The ‘Free Cards’ page contains a form that users can fill in if they wish to register for a free pack of playing cards.

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

Describe how all the form’s inputs could be comprehensively tested.

(3 marks)

Candidate name_____ Candidate number_____

Acknowledgement of copyright
Electronic files -
Task 3 - aquatarkus/Shutterstock.com
Task 3 - BEPictured/Shutterstock.com
Task 3 - J and S Photography/Shutterstock.com
Task 3 - Bardocz Peter/Shutterstock.com
Task 3 - stockfour/Shutterstock.com
Task 3 - SKatzenberger/Shutterstock.com
Task 3 - In Green/Shutterstock.com

Administrative information

Published: January 2019 (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.