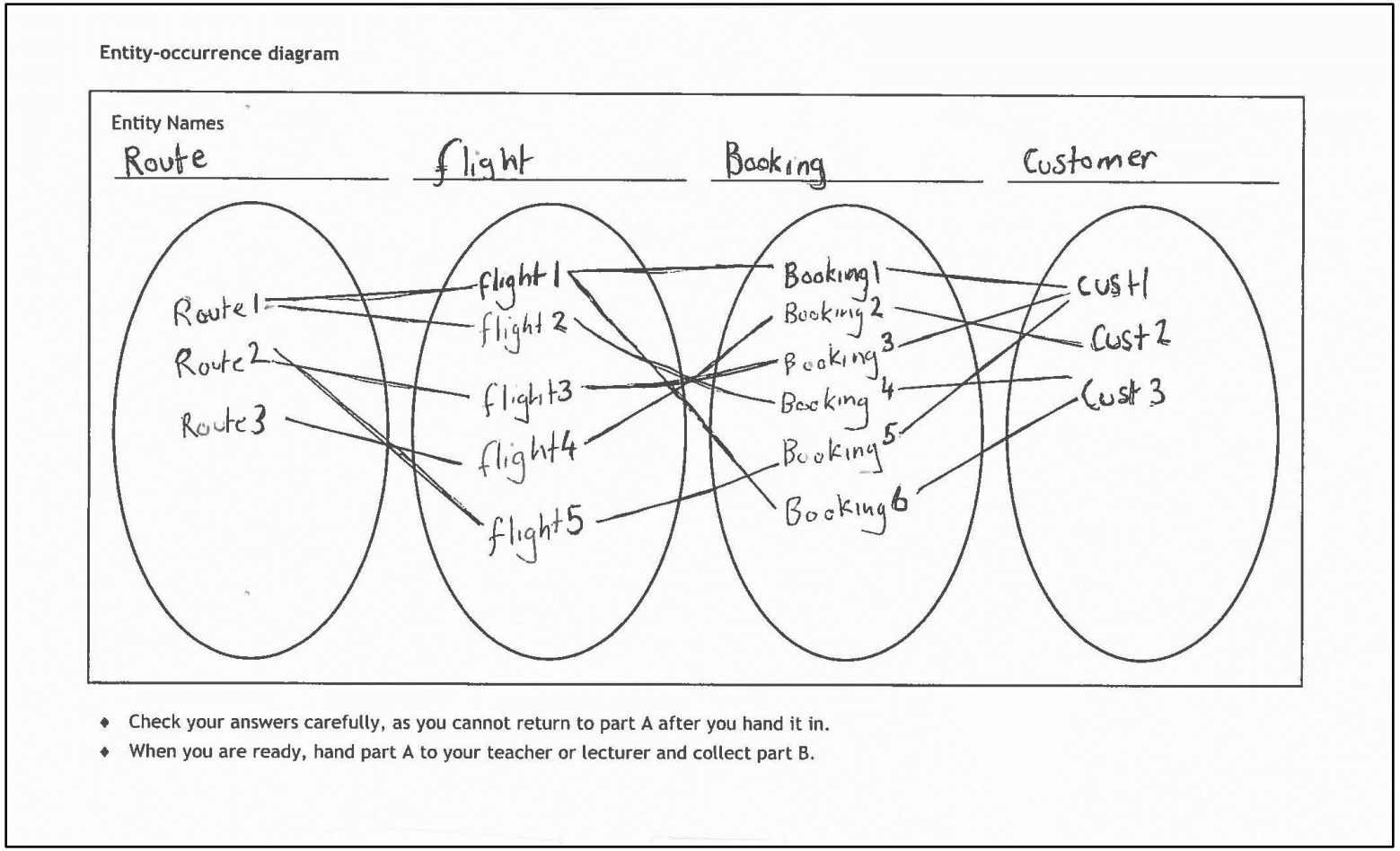
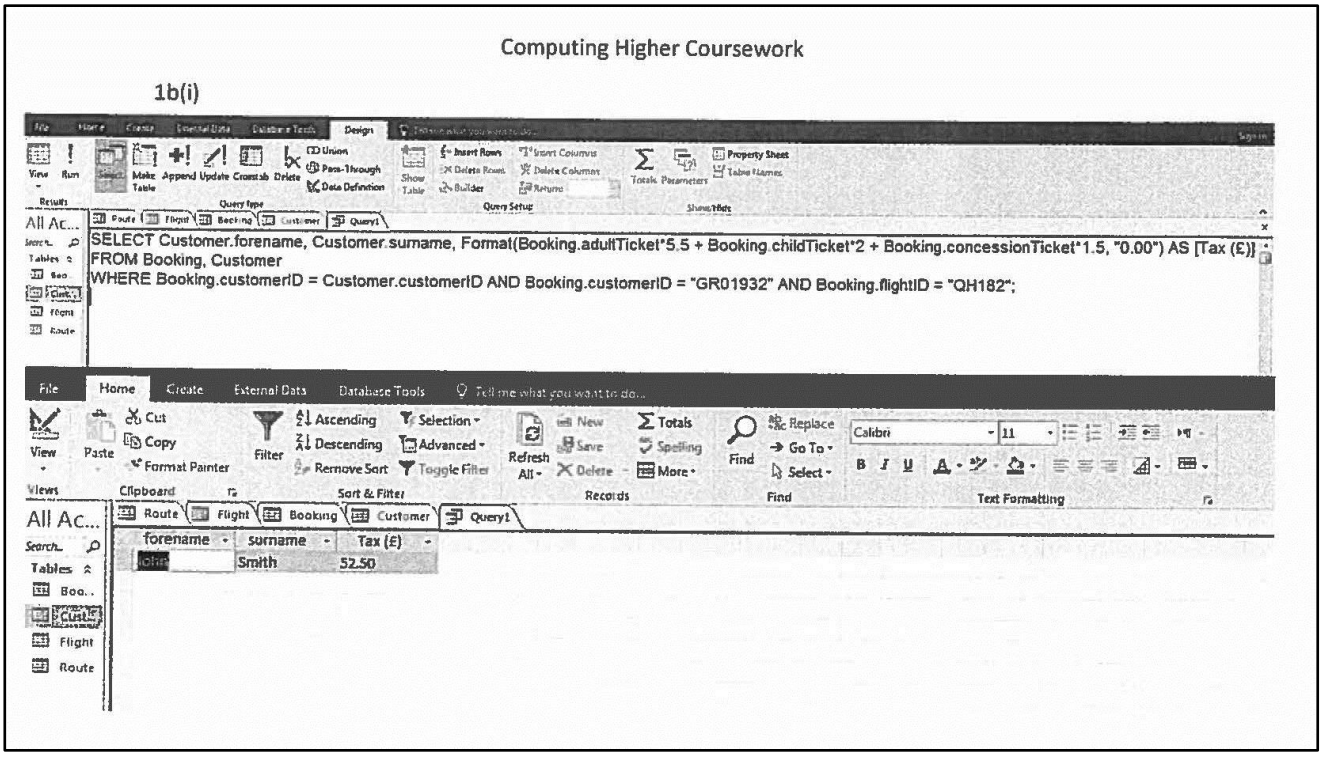
Candidate 1

Task 1 – Database design and development

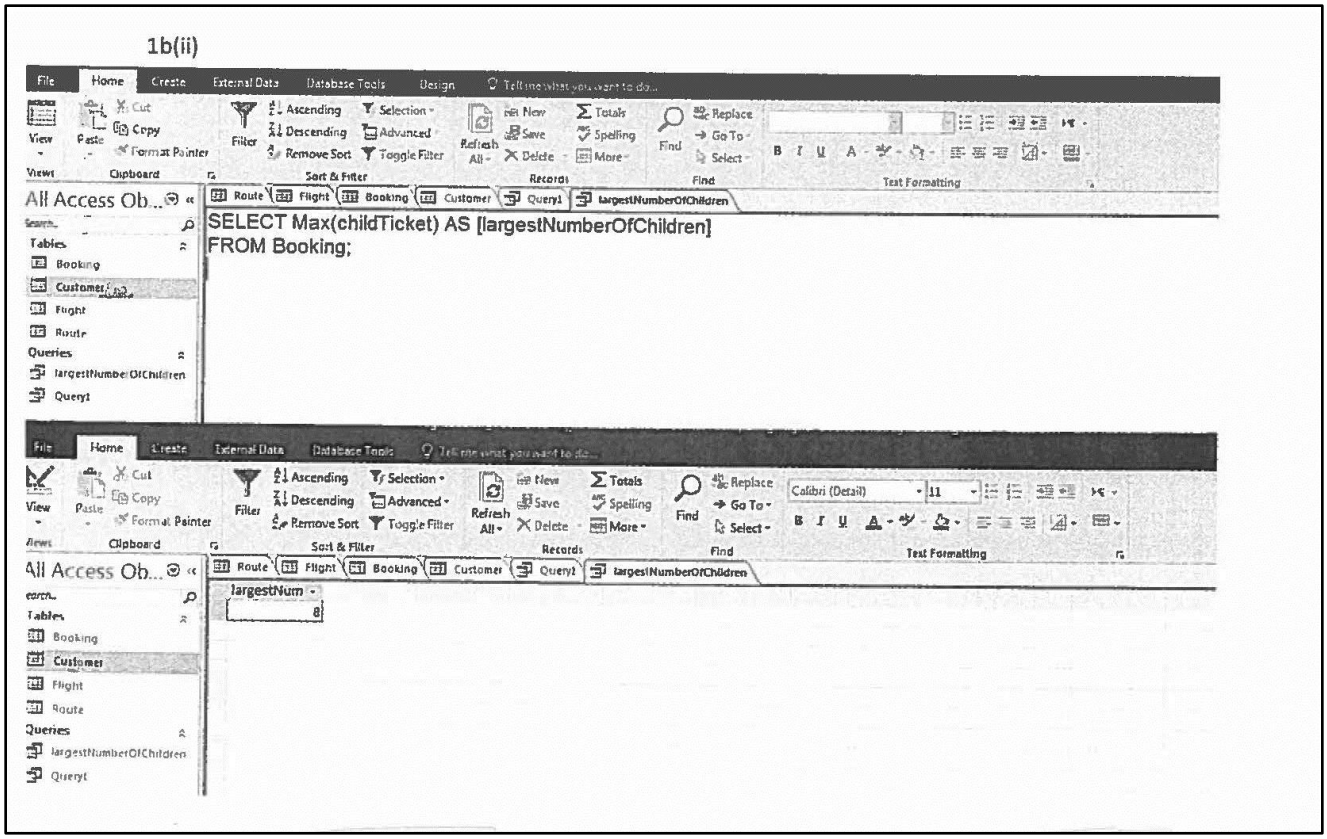
#Question 1(a)



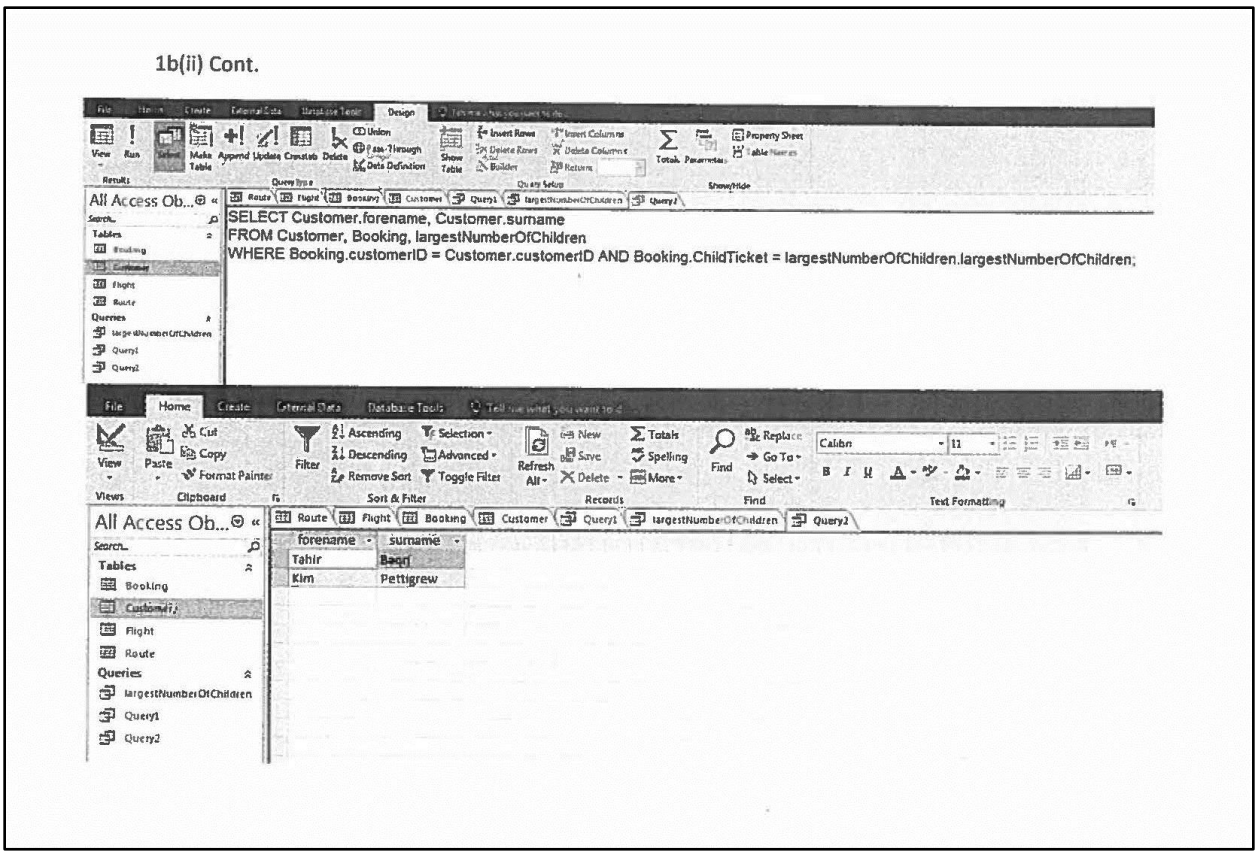
Question 1(b)(i)



Question 1(b)(ii)



1b(ii) Cont.


SELECT Customer.forename, Customer.surname
FROM Customer, Booking, largestNumberOfChildren
WHERE Booking.customerID = Customer.customerID AND Booking.ChildTicket = largestNumberOfChildren.largestNumberOfChildren;
All Access Ob...
Tables
Booking
Customer
Flight
Route
Queries
largestNumberOfChildren
Query1
Query2
forename
surname
Tahir
Baqri
Kim
Pettigrew

Question 1(c)

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)

One potential problem that may occur is queries may turn out inaccurate. If a query was used to add new data to the flight table and it was dependant on a condition of a field and that field has incorrect data or no data, the query would fail to properly add the new data

Problem 2

(1 mark)

Another problem that may occur is that inputed data would be inaccurate. This could be inaccurate in the sense of no data or useless data. If a client was to ask for their data and it was unable to be found, there would be a problem in the sense that they cannot get their flight in this case to the company being sued for false handeliing of data.

Candidate nam

Task 2 – Software design and development
Question 2(a)

2a Using the problem description, identify the functional requirements of the program.
(3 marks)

Input(s) Forename of walkers Surename of walkers Amount of miles walked
Process(es) Find the highest amount of miles walked Work out 70% of the highest amount of miles walked
Output(s) Display the names of the walker that are within 70% of the highest amount of miles walked

Candidate name_____ Candidate number_____

Question 2(b)

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	members (forename, surname, distance)
	OUT	members(forename,surname,distance)
Find the furthest distance walked	IN	furthest
	OUT	furthest
Display the furthest distance walked	IN	furthest
	OUT	
Write club prize winners to file	IN	members(forename,surname,distance) furthest
	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.

Question 2(c)(i)

N:\Higher Computing\Coursework\Python\Membersprogram.py 01 March 2019 11:39

```
class memberList():
    def __init__(self, forename, surname, distance):
                                                #initilises the data for the array
        self.forename = forename


        self.surname = surname
        self.distance = float(distance)


def readFile():


    members = []

    #creates an array
    membersFile = open("members.txt", "r")
                                                        #opens the memebrs
    file

    for line in membersFile:

        array = line.split(",")
                                                                    #splits each
        line
        hiker = memberList(array[0], array[1], array[2])
                                                #sets hiker to the current line
        members.append(hiker)


    membersFile.close()
                                                                        #closes
    the members file
    return members

    #returns members

def furthestDis(members):
    furthest = members[0].distance
                                                            #sets furthest to
    the first hiker in the array
    for index in range(1, len(members)):
                                                         #creates a loop
        if members[index].distance >furthest:
                                                        #checks which value is the
        highest
            furthest = members[index].distance
                                                          #sets furthest to the
            highest value
    return furthest


def printDistance(furthest):
    print("The furthest distance walked was", furthest)
                                            #pritns the furthest distance walked
```

-1-

N:\Higher Computing\Coursework\Python\Membersprogram.py 01 March 2019 11:39

```
def write(members, furthest):
    highest = open("Highest.txt", "w")

    highest.write("The Winning Members are:\n")                    #creates the results file

    for counter in range (len(members)):                           #writes the text to the file

        if members[counter].distance > 0.7*furthest:               #starts a loop

        the furthest distance walked                               #checks if each value is 70% of
            highest.write(members[counter].forename + "," + members[counter].surname +
            "\n")       #writes the winners to the
            file
    highest.write("\nThe number of whole marathons walked by each member is:\n")
    for counter in range(len(members)):

    loop for the whole members file                                #Starts a
        marathons = int(members[counter].distance/26.22)

        store the number of marathons walked                       #Creates a variable to
        highest.write(members[counter].forename + "," + members[counter].surname + "," +
        str(marathons) + "\n") #Writes to file

    highest.close()

    #closes the file



#--MAIN--#
members = readFile()
furthest = furthestDis(members)
printDistance(furthest)
write(members, furthest)
```

-2-

Question 2(c)(ii)

N:\Higher Computing\Coursework\Python\Membersprogram.py 01 March 2019 11:39

```
def write(members, furthest):
    highest = open("Highest.txt", "w")

    highest.write("The Winning Members are:\n")                    #creates the results file

    for counter in range (len(members)):                           #writes the text to the file

        if members[counter].distance > 0.7*furthest:               #starts a loop

        the furthest distance walked                               #checks if each value is 70% of
            highest.write(members[counter].forename + "," + members[counter].surname +
            "\n")       #writes the winners to the
            file
    highest.write("\nThe number of whole marathons walked by each member is:\n")
    for counter in range(len(members)):

    loop for the whole members file                                #Starts a
        marathons = int(members[counter].distance/26.22)

        store the number of marathons walked                       #Creates a variable to
        highest.write(members[counter].forename + "," + members[counter].surname + "," +
        str(marathons) + "\n") #Writes to file

    highest.close()

    #closes the file



#--MAIN--#
members = readFile()
furthest = furthestDis(members)
printDistance(furthest)
write(members, furthest)
```

-2-

Question 2(d)

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)

Line	distance	furthest
20	145.6	145.6
20	145.6	145.6
20	145.6	145.6
20	398.5	398.5
20	398.5	398.5
20	409.0	409.0
20	409.0	409.0
20	409.0	409.0
20	409.0	409.0
20	409.0	409.0
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2
20	505.2	505.2

Candidate name_____ Candidate number_____

Question 2(e)

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

- the fitness for purpose of your program

(1 mark)

My program has been accurratly made for the purpose required,it finds the furthest distance walked with data from a text file then displays the names of people who walked 70% or more of the furhtast distance

- the maintainability of your program with reference to readability and modularity

(2 marks)

The program is readable as commentary has been added to describe what each line does.

I have used indentation which helps the readablility of the program

I have clearly separated each class aiding the readability of the program

Candidate name_____ Candidate number_____

Task 3 – Web design and development
Question 3(a)

3a State two functional requirements for the website.

Functional requirement 1 :

The website must allow for an application to the foundation to be sent by filling out and submitting a form on the website in order to recieve a set of cards.

(1 mark)

Functional requirement 2 :

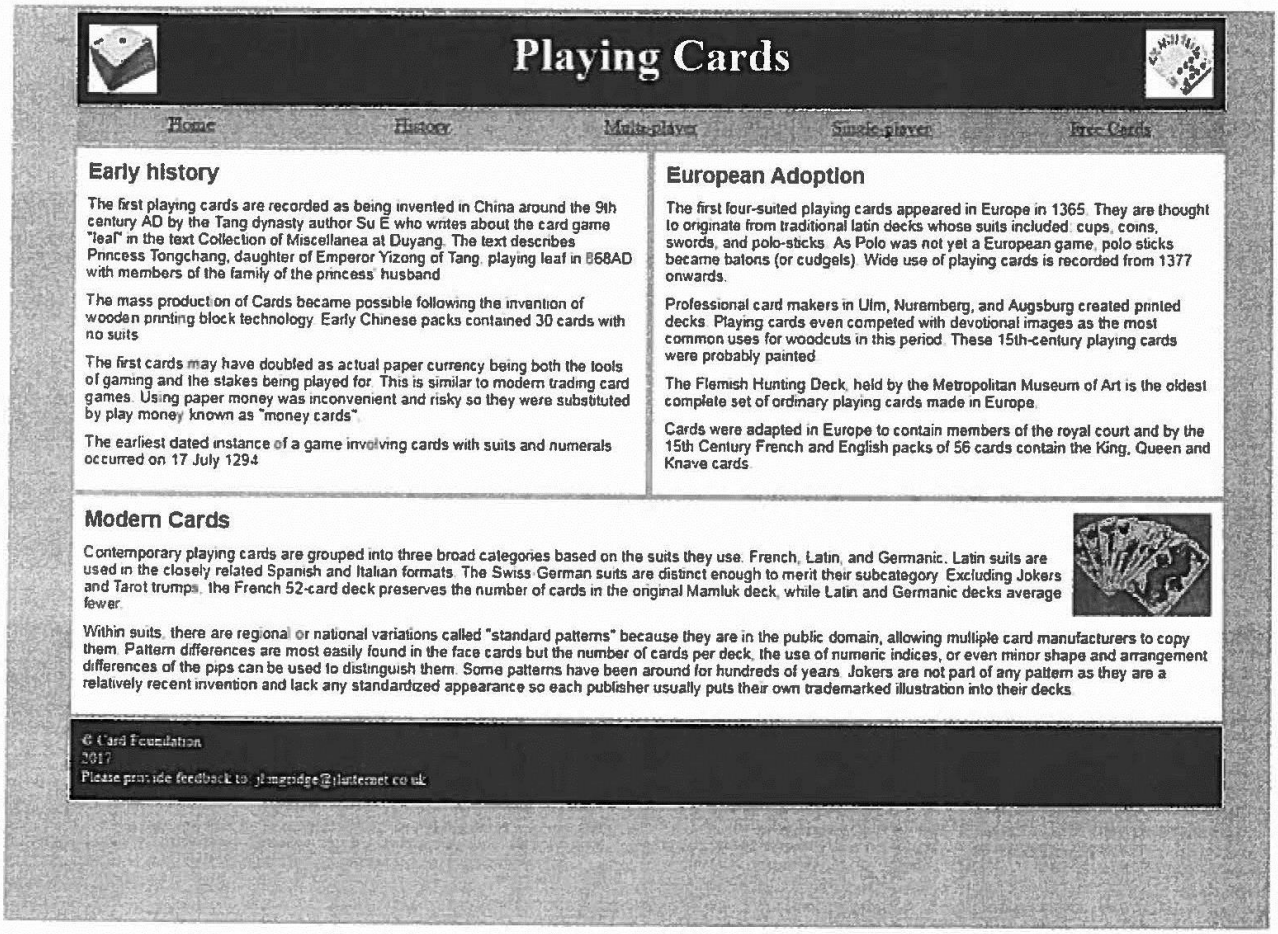
The website must contain details of singleplayer and multiplayer cards games and instructions and tips on how to play these games.

(1 mark)

Candidate name: Candidate number:

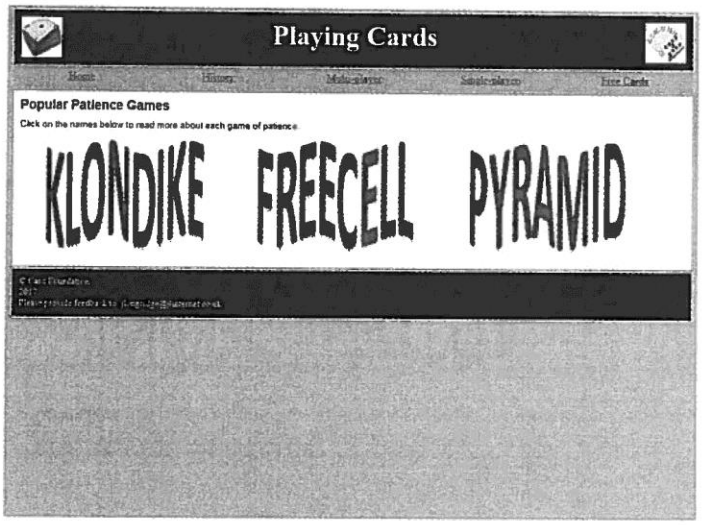
Question 3(b)(i)

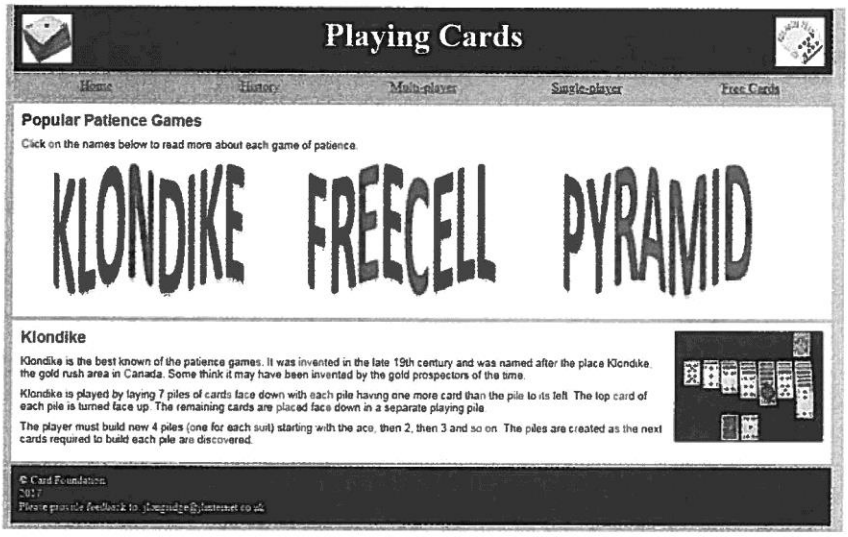
3b(i)



Question 3(b)(ii)

3b(ii)





```
<!DOCTYPE html>
<html>

<head>
<title>History</title>
<link rel="stylesheet" type="text/css" href="../CSS/styles.css">
</head>

<body>

<!-- Page Header -->
<header>
<img class="imageBannerLeft" src="../images/bannerCardsLeft.png">
<img class="imageBannerRight" src="../images/bannerCardsRight.png">
<h1>Playing Cards</h1>

</header>

<!-- Navigation Bar -->
<nav>
<ul>
<li><a href="home.html">Home</a></li>
<li><a href="history.html">History</a></li>
<li><a href="multi.html">Multi-player</a></li>
<li><a href="single.html">Single-player</a></li>
<li><a href="register.html">Free Cards</a></li>
</ul>
</nav>

<!-- The main content of the page -->
<main>
<section ID="historyLeft">
<h2>Early history</h2>
<p>The first playing cards are recorded as being invented in China around the
9th century AD by the Tang dynasty author Su E who writes about the card game
"leaf" in the text Collection of Miscellanea at Duyang.  The text describes
Princess Tongchang, daughter of Emperor Yizong of Tang, playing leaf in 868AD
with members of the family of the princess' husband.</p>
<p>The mass production of Cards became possible following the invention of
wooden printing block technology.  Early Chinese packs contained 30 cards with
no suits.</p>
<p>The first cards may have doubled as actual paper currency being both the
tools of gaming and the stakes being played for.  This is similar to modern
trading card games. Using paper money was inconvenient and risky so they were
substituted by play money known as "money cards".</p>
<p>The earliest dated instance of a game involving cards with suits and numerals
occurred on 17 July 1294.</p>
</section>

<section ID="historyRight">
<h2>European Adoption</h2>
<p>The first four-suited playing cards appeared in Europe in 1365.  They are
thought to originate from traditional latin decks whose suits included: cups,
coins, swords, and polo-sticks.  As Polo was not yet a European game, polo
sticks became batons (or cudgels).  Wide use of playing cards is recorded from
1377 onwards.</p>
<p>Professional card makers in Ulm, Nuremberg, and Augsburg created printed
decks. Playing cards even competed with devotional images as the most common
uses for woodcuts in this period. These 15th-century playing cards were probably
painted.</p>
<p>The Flemish Hunting Deck, held by the Metropolitan Museum of Art is the
oldest complete set of ordinary playing cards made in Europe.</p>
<p>Cards were adapted in Europe to contain members of the royal court and by the
15th Century French and English packs of 56 cards contain the King, Queen and
Knave cards.</p>
</section>

<section ID="historyBot">
<img class="imageCardsRight" src="../images/germanPlayingCards.jpg">
<h2>Modern Cards</h2>
```

```
51  <p>Contemporary playing cards are grouped into three broad categories based on
    the suits they use: French, Latin, and Germanic. Latin suits are used in the
    closely related Spanish and Italian formats. The Swiss-German suits are distinct
    enough to merit their subcategory. Excluding Jokers and Tarot trumps, the French
    52-card deck preserves the number of cards in the original Mamluk deck, while
    Latin and Germanic decks average fewer.</p>
52  <p>Within suits, there are regional or national variations called "standard
    patterns" because they are in the public domain, allowing multiple card
    manufacturers to copy them. Pattern differences are most easily found in the
    face cards but the number of cards per deck, the use of numeric indices, or even
    minor shape and arrangement differences of the pips can be used to distinguish
    them. Some patterns have been around for hundreds of years. Jokers are not part
    of any pattern as they are a relatively recent invention and lack any
    standardized appearance so each publisher usually puts their own trademarked
    illustration into their decks. </p>
53  </section>
54
55
56
57  </main>
58
59  <!-- Page Footer -->
60  <footer>
61  <p> &copy; Card Foundation <br> 2017 <br> Please provide feedback to:
    jlongridge@jlinternet.co.uk </p>
62  </footer>
63
64  </body>
65  </html>
```

```
/* The Universal Selector has been used here to cancel the browser default
margins and */
/* padding for every page element.  This ensures that only assigned margin and
padding values are seen when viewing pages. */
* {margin:0;padding:0}

/* Margins - main page areas */
header, main, footer  {margin-top:5px}

/* Padding */
header, footer, section {padding:10px}

/* Positioning (Float, Display and Clear) */
header, nav, main, footer {display:block;clear:both}
.hidden{display:none}

/* Background Colours */
body, div, main {background-color:LightBlue}
nav {background-color:LightSteelBlue}
header, footer {background-color:DarkBlue}
section {background-color:White}



/* CSS specific to page areas */

/* Body */
body{margin:auto;width:1000px}



/* Header */
header {height:60px}
h1{font-family:Verdanda;color:White;font-size:30pt;text-align:center;display:block
}
.imageBannerRight {width:60px;height:60px;float:right}
.imageBannerLeft {width:60px;height:60px;float:left}



/* Nav */
nav  {margin-top:0px}
nav {height:34px}

nav ul {list-style-type:none}
nav ul li {float:left;width:200px;text-align:center}
nav ul li a {display:block;padding:8px}
nav ul li a:hover {background-color:DarkBlue;color:White}



/* Main */
main {display:block}
main section {margin-bottom:5px}
p ul {margin-top:15px;margin-bottom:15px;font-size:12pt;text-align:left}
h2, p{font-family:Helvetica;color:Black}
h2 {margin-bottom:10px;font-size:14pt;text-align:left}
p{margin-bottom:10px;font-size:10pt;text-align:left}
input, select {margin-left:50px}


/*Home Page*/
.imageLargeLeft
{width:300px;height:225px;float:left;margin-right:10px;margin-bottom:10px}

/* History Page  */
#historyLeft{float:left; width:477px}
#historyRight{float:right; width:477px}
#historyBot{clear:both}
.imageCardsRight {width:120px;height:90px; float:right}
```

```
68  /* Patience Games Page  */
69  .imageMediumRight
    {width:120px;height:90px;float:right;margin-left:10px;margin-bottom:10px}
70  .patienceButtons {width:250px;height:170px;margin-left:30px;margin-right:30px}
71  .patiencePhotos
    {width:180px;height:135px;float:right;margin-left:10px;margin-bottom:20px}
72  .patienceSections {display:none}
73
74  /*Single Page*/
75  .imageLargeRight
    {width:300px;height:225px;float:right;margin-left:10px;margin-bottom:10px}
76
77  /*Multi Page*/
78  .multiPhotos {width:330px;height:200px;margin-bottom:5px}
79  .firstTwoColumns {float:left;width:330px;margin-right:5px}
80  .thirdColumn {float:left;width:329px}
81
82
83
84
85  /* Footer */
86  footer {height:50px}
87  footer p
    {font-family:Verdanda;text-align:left;color:White;font-size:10pt;margin-top:0px}
88
89
90
```

```
<!DOCTYPE html>
<html>

<head>
<title>Patience</title>
<link rel="stylesheet" type="text/css" href="../CSS/styles.css">

<script language="javascript">
    function showKlondike() {
                    document.getElementById("Klondike").style.display="block";
                    document.getElementById("FreeCell").style.display="none";
                    document.getElementById("Pyramid").style.display="none";
                        }
    function showFreeCell() {
                    document.getElementById("Klondike").style.display="none";
                    document.getElementById("FreeCell").style.display="block";
                    document.getElementById("Pyramid").style.display="none";
                        }
    function showPyramid()  {
                    document.getElementById("Klondike").style.display="none";
                    document.getElementById("FreeCell").style.display="none";
                    document.getElementById("Pyramid").style.display="block";
                        }
</script>
</head>

<body>

<!-- Page Header -->
<header>
<img class="imageBannerLeft" src="../images/bannerCardsLeft.png">
<img class="imageBannerRight" src="../images/bannerCardsRight.png">
<h1>Playing Cards</h1>

</header>

<!-- Navigation Bar -->
<nav>
<ul>
<li><a href="home.html">Home</a></li>
<li><a href="history.html">History</a></li>
<li><a href="multi.html">Multi-player</a></li>
<li><a href="single.html">Single-player</a></li>
<li><a href="register.html">Free Cards</a></li>
</ul>
</nav>

<!-- The main content of the page -->
<main>
<section>
<h2>Popular Patience Games</h2>
<p>Click on the names below to read more about each game of patience.</p>
<p>
<img class="patienceButtons" src="../images/klondikeButton.png"
onClick="showKlondike()">
<img class="patienceButtons" src="../images/freecellButton.png"
onClick="showFreeCell()">
<img class="patienceButtons" src="../images/pyramidButton.png"
onClick="showPyramid()">
</p>
</section>

<section class="patienceSections" id="Klondike">
<img class="patiencePhotos" src="../images/klondikePhoto.jpeg">
<h2>Klondike</h2>
<p>Klondike is the best known of the patience games.  It was invented in the
late 19th century and was named after the place Klondike, the gold rush area in
Canada. Some think it may have been invented by the gold prospectors of the
time. </p>
<p>Klondike is played by laying 7 piles of cards face down with each pile having
```

```
one more card than the pile to its left.  The top card of each pile is turned
face up.  The remaining cards are placed face down in a separate playing pile.</p>
65  <p>The player must build new 4 piles (one for each suit) starting with the ace,
then 2, then 3 and so on.  The piles are created as the next cards required to
build each pile are discovered.</p>
66  </section>
67
68  <section class="patienceSections" id="FreeCell">
69  <img class="patiencePhotos" src="../images/freecellPhoto.jpeg">
70  <h2>Freecell</h2>
71  <p>FreeCell is played using the standard 52-card deck. Compared to most patience
games it is very easy to complete the game. All cards are dealt face-up at the
beginning of the game.</p>
72  <p>FreeCell has been included as part of Windows since 1995 making it very
popular.  Many other platforms have attempted to replicate the MicroSoft version
as players don't tend to like variations from that version.</p>
73  <p>The game begins with four open cells and four foundations.  A standard 52
card deck of cards is used.</p>
74  <p>Cards are dealt face-up into eight columns positioned so that every card can
be seen.  Four columns have seven cards and four have six cards.</p>
75  <p>The top card of any colum can be covered with a card one lower in value and
the opposite colour. Foundations are built up by suit.</p>
76  <p>The four cells can be used to temporarly store cards while moving other cards
around.</p>
77  <p>The game is complete when all four foundation piles are complete from Ace to
King in their correct suits. </p>
78  </section>
79
80
81  <section class="patienceSections" id="Pyramid">
82  <img class="patiencePhotos" src="../images/pyramidPhoto.jpeg">
83  <h2>Pyramid</h2>
84  <p>The object is to remove all the cards from the pyramid to the foundation. </p>
85  <p>The game begins by laying out a pyramid of cards.  One card at the top
covered by two cards on the next row down, covered by three cards on the next
row and so on up to the last row of seven cards (28 cards in total).  The
remaining cards are placed face down in a pile.</p>
86
87  <p>When using a a standard deck, Kings score 13, Queens 12 and Jacks 11.</p>
88
89  <p>Play begins by turning over the top card in the face down pile.  Players try
to match pairs of cards to a score of 13 (or the King, which itself is worth
13).  When a pair is identified the cards are removed from the game.  If no pair
is possible the turned card is placed face up in a discarded cards pile next to
the face down pile. A pair may be made from any visible card in the pyramid, the
turned card or the last discarded card.</p>
90
91  <p>When the pile is complete the discarded cards are turned over and becomes the
new pile.</p>
92
93  <p>The game is scored by counting the number of cards left in the pyramid at the
end of the game. A perfect score is 0 as the whole pyramid has been removed.</p>
94  </section>
95
96
97
98  </main>
99
100 <!-- Page Footer -->
101 <footer>
102 <p> &copy; Card Foundation <br> 2017 <br> Please provide feedback to:
jlongridge@jlinternet.co.uk </p>
103 </footer>
104
105 </body>
106 </html>
```

3b(ii) Cont.

Playing Cards

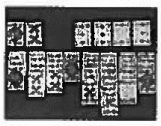
Home History Multi-player Single-player Free Cards

Popular Patience Games

Click on the names below to read more about each game of patience

KLONDIKE FREECELL PYRAMID

Freecell



FreeCell is played using the standard 52-card deck. Compared to most patience games it is very easy to complete the game. All cards are dealt face-up at the beginning of the game.

FreeCell has been included as part of Windows since 1995 making it very popular. Many other platforms have attempted to replicate the MicroSoft version as players don't tend to like variations from that version.

The game begins with four open cells and four foundations. A standard 52 card deck of cards is used.

Cards are dealt face-up into eight columns positioned so that every card can be seen. Four columns have seven cards and four have six cards.

The top card of any colum can be covered with a card one lower in value and the opposite colour. Foundations are built up by suit.

The four cells can be used to temporarily store cards while moving other cards around.

The game is complete when all four foundation piles are complete from Ace to King in their correct suits.

Playing Cards

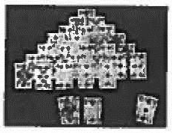
Home History Multi-player Single-player Free Cards

Popular Patience Games

Click on the names below to read more about each game of patience

KLONDIKE FREECELL PYRAMID

Pyramid



The object is to remove all the cards from the pyramid to the foundation.

The game begins by laying out a pyramid of cards. One card at the top covered by two cards on the next row down, covered by three cards on the next row and so on up to the last row of seven cards (28 cards in total). The remaining cards are placed face down in a pile.

When using a a standard deck, Kings score 13, Queens 12 and Jacks 11.

Play begins by turning over the top card in the face down pile. Players try to match pairs of cards to a score of 13 (or the King, which itself is worth 13). When a pair is identified the cards are removed from the game. If no pair is possible the turned card is placed face up in a discarded cards pile next to the face down pile. A pair may be made from any visible card in the pyramid, the turned card or the last discarded card.

When the pile is complete the discarded cards are turned over and becomes the new pile.

The game is scored by counting the number of cards left in the pyramid at the end of the game. A perfect score is 0 as the whole pyramid has been removed.

Candidate Name: Candidate Number:

Question 3(c)

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)

In testing the form's inputs we want to ensure that all validation for the form is working as it should. This includes:

Ensuring length validation is obeyed. This can be done by typing answers into the texts fields to ensure their max length is not exceed. This includes the first name's 15 character limit, Town's 20 character limit and the emails 70 character limit to name a few.

Ensuring presense validation is obeyed. This can be done by submitting a form with each of the required fields blank one at a time hopefully prompting a reminder to fill them in. This systematically ensures that when submitting a form these inputs cannot be left blank.

Ensuring range validation is obeyed. This is restricted to the "number of years you have been playing" and it can be proven by trying to exceed 100 years and have less that 0 years. If the answer is restricted appropriately then the input is working as intended.

Candidate name: Candidate number: