

SA's Leading Past Year

Exam Paper Portal



You have Downloaded, yet Another Great  
Resource to assist you with your Studies 😊

Thank You for Supporting SA Exam Papers

Your Leading Past Year Exam Paper Resource Portal

Visit us @ [www.saexampapers.co.za](http://www.saexampapers.co.za)



SA EXAM  
PAPERS



# basic education

Department:  
Basic Education  
**REPUBLIC OF SOUTH AFRICA**

## **SENIOR CERTIFICATE EXAMINATIONS**

**INFORMATION TECHNOLOGY P1**

**2017**

**MARKS: 150**

**TIME: 3 hours**

**This question paper consists of 21 pages.**

**INSTRUCTIONS AND INFORMATION**

1. This question paper is divided into THREE sections. Candidates must answer ALL THREE sections.
2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.
3. This question paper is set with programming terms that are not specific to any particular programming language (Delphi/Java (using the Netbeans IDE)).
4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.
5. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.
6. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.
7. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of a programming language for any of these routines.
8. All data structures must be defined by you, the programmer, unless the data structures are supplied.
9. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.
10. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.
11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.
12. At the end of this examination session you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

13. The files that you need to complete this question paper have been given to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

**NOTE:**

- Delphi candidates must use the file **DelphiDataENGJune2017.exe**.
- Java candidates must use the file **JavaDataENGJune2017.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the extract button.
- Enter the following password: **AiR%\$Port**

Once extracted, the following list of files will be available in the folder **DelphiDataENGJune2017/JavaDataENGJune2017**:

**DELPHI FILES****Question1:**

Gold.jpg  
NonMember.jpg  
Platinum.jpg  
Question1\_P.dpr  
Question1\_P.dprj  
Question1\_P.res  
Question1\_U.dfm  
Question1\_U.pas  
Silver.jpg

**Question2:**

DataQ2.txt  
Flight\_U.pas  
Question2\_P.dpr  
Question2\_P.dprj  
Question2\_P.res  
Question2\_U.dfm  
Question2\_U.pas

**Question3:**

Question3\_P.dpr  
Question3\_P.dprj  
Question3\_P.res  
Question3\_U.dfm  
Question3\_U.pas

**JAVA (NETBEANS) FILES****Question1:**

Gold.jpg  
NonMember.jpg  
Platinum.jpg  
Question1.form  
Question1.java  
Silver.jpg

**Question2:**

DataQ2.txt  
Flight.java  
Question2.form  
Question2.java

**Question3:**

PopulateArrays.java  
Question3.form  
Question3.java

**SCENARIO**

A number of different software programs are used at an airport to assist staff in confirming and validating details of flights and passengers.

**SECTION A****QUESTION 1: GENERAL PROGRAMMING SKILLS**

A software program is used by the airline to capture initial passenger registration information. This information is used to update the passenger profile and to assist the airline to cater for the passengers on board.

**INSTRUCTIONS:**

<b>DELPHI PROGRAMMERS</b>	<b>JAVA PROGRAMMERS</b>
<ul style="list-style-type: none"><li>• Open the incomplete project file called <b>Question1_P.dpr</b> in the <b>Question1</b> folder.</li><li>• Enter your examination number as a comment in the first line of the main form unit called <b>Question1_U.pas</b>.</li></ul>	<ul style="list-style-type: none"><li>• Open the incomplete class called <b>Question1.java</b> in the folder <b>Source Packages (src)</b>, <b>Question1Package</b> in the <b>Question1</b> folder.</li><li>• Enter your examination number as a comment in the first line of the class called <b>Question1.java</b>.</li></ul>

Do the following:

- Compile and execute the program. The graphical user interface (GUI) displays five sections named Question 1\_1 to Question 1\_5. Currently the program has no functionality.
- Complete the code for the program, as described in QUESTION 1.1 to QUESTION 1.5 on the next page.

Example of graphical user interface (GUI):

The GUI is divided into several sections:

- Question1\_1:** Contains a text box for 'Age of passenger', two checkboxes for 'Valid passport' and 'Accompanied by adult', and a button labeled 'Question 1.1'.
- Question1\_2:** Contains a text box for 'Luggage weight in kg', a list of airlines with their maximum weights (ZAA#25kg, UKAL#30kg, MAO#20kg, KAA#15kg, SA#5kg, ETTAL#100kg), and a button labeled 'Question 1.2'.
- Question1\_3:** Contains a text box for 'Number of passengers' and a button labeled 'Question 1.3'.
- Question1\_4:** Contains text boxes for 'Time of departure' and 'Boarding time', and a button labeled 'Question 1.4'.
- Question1\_5:** Contains a text box for 'Distance in kilometres', radio buttons for 'Flyer Card' (Silver, Gold, Platinum), and a button labeled 'Question 1.5'.

A large box on the right side of the GUI displays the text 'Not a member'.

- 1.1 For the purpose of this question you may assume that all passengers are travelling on international flights. Passengers on international flights must have a valid passport and passengers younger than 16 years must be accompanied by an adult.

Write code to obtain the passenger's age from the text box provided, test whether the correct check boxes are ticked and display a suitable message to indicate whether the passenger's boarding is confirmed, or not.

Example of output if the passenger is 23 years old and has a valid passport:

The output shows the following state:

- Question1\_1:** 'Age of passenger' is 23, 'Valid passport' is checked, 'Accompanied by adult' is unchecked.
- Question 1.1:** The button is highlighted, and the output field displays 'Boarding is confirmed.'

Example of output if the passenger is 14 years old and has a valid passport, but is not accompanied by an adult:

Example of output if the passenger is 14 years old, has a valid passport and is accompanied by an adult:

(7)

- 1.2 A list box is provided with the names of different airlines and the maximum luggage weight allowed per passenger. The format of the text in the list box is as follows:

```
<airline name>#<maximum weight>kg
```

An amount of R50,00 per kilogram has to be paid if the passenger's luggage weight exceeds the maximum weight specified by the airline. The amount to be paid per kilogram has been declared as a constant in the provided code.

Write code to do the following:

- Obtain the passenger's luggage weight from the text box and the maximum luggage weight allowed for the airline selected from the list box.
- Use the information obtained to calculate the excess weight, if any.
- Use the constant variable and the calculated excess weight to determine the cost of the excess weight of the passenger's luggage.
- Display the excess weight and the cost of the excess weight. The cost must be formatted to a currency with TWO decimal places.

Example of output if the passenger's luggage weighs 35,89 kg (35.89kg) and UKAL airline has been selected in the list box:

Question1\_2

Luggage weight in kg    Airline#Maximum weight

35.89

ZAA#25kg  
UKAL#30kg  
MAO#20kg  
KAA#15kg  
SA#5kg  
ETTAL#100kg

Excess weight: 5.89kg  
Cost: R294.50

Question 1.2

Example of output if the passenger's luggage weighs 12,5 kg (12.5kg) and KAA airline has been selected in the list box:

Question1\_2

Luggage weight in kg    Airline#Maximum weight

12.5

ZAA#25kg  
UKAL#30kg  
MAO#20kg  
KAA#15kg  
SA#5kg  
ETTAL#100kg

Excess weight: 0.00kg  
Cost: R0.00

Question 1.2

(10)

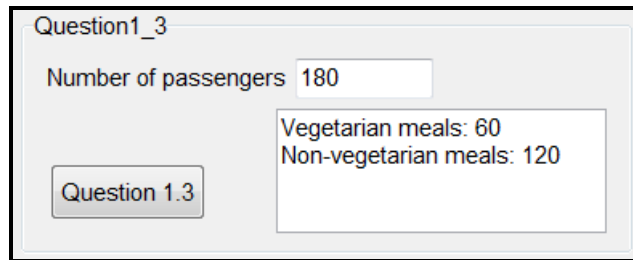
- 1.3 The flight catering company prepares passenger meals in advance to cater for the number of passengers on the flight. The meal options are vegetarian and non-vegetarian. The standard policy of the catering company is to prepare one third vegetarian meals and two thirds non-vegetarian meals. If the number of passengers does not divide exactly into three parts, the remaining number of meals prepared after dividing by three must be non-vegetarian.

Write code to do the following:

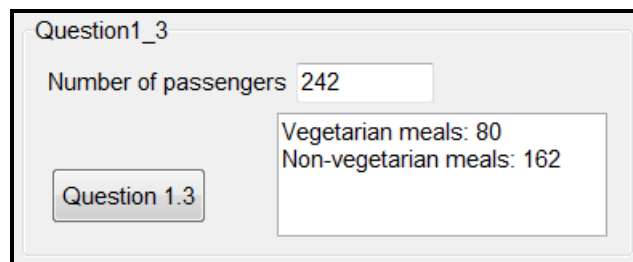
- Obtain the number of passengers from the text box provided.
- Determine and display the number of vegetarian meals and the number of non-vegetarian meals that must be prepared for the passengers.



Example of output if there are 180 passengers:



Example of output if there are 242 passengers:



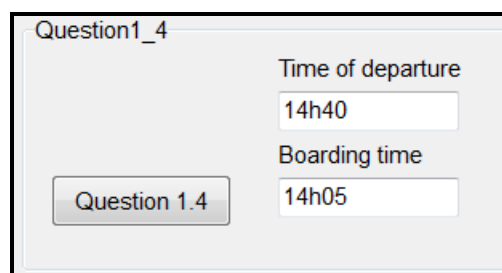
(5)

- 1.4 The boarding time of flights is thirty-five minutes before the time of departure. The user has to enter the time of departure in the format <hour>h<minute>.

Write code to do the following:

- Obtain the time of departure from the text box provided.
- Validate the time of departure to ensure that the following applies:
  - The format must be <hour>h<minute>.
  - The hour value must be less than or equal to 23 and must consist of two digits.
  - The minute value must be less than or equal to 59 and must consist of two digits.
- If the time of departure is invalid, use a dialog box to display a suitable message and clear the time of departure text box.
- If the time of departure is valid, calculate and display the boarding time in the text box provided.

Example of output if the time of departure is 14h40:



Example of output if the time of departure is 17h15:

Example of output if the time of departure is invalid or entered in the incorrect format, for example 15:67:

(17)

- 1.5 A passenger who has a flyer card can earn points, depending on the distance travelled and whether the passenger has a silver, gold or platinum card. The list of benefits for each type of flyer card is saved as separate image files. The names of the image files are **Silver.jpg**, **Gold.jpg**, **Platinum.jpg** and **NonMember.jpg**. The default image file, **NonMember.jpg**, is displayed in the image component provided.

Write code to do the following:

- Obtain the distance travelled from the text box provided.
- Select the type of flyer card from the radio buttons provided.
- Display the corresponding image file for the flyer card selected.
- Calculate the points earned as follows:
  - All card passengers will earn one point for every 1,6 kilometres travelled.
  - Gold card passengers will earn additional bonus points, calculated at 15% of the distance travelled.
  - Platinum card passengers will earn additional bonus points, calculated at 20% of the distance travelled.
  - The total points must be rounded down to the nearest whole number.
- The panel named **pnlPoints** must be shown if a card was selected. The total points earned must be displayed in this panel.

Example of output if the distance is 1 233 kilometres and the passenger has a silver flyer card:

Question1\_5

Distance in kilometres

Flyer Card

☒ Silver

☐ Gold

☐ Platinum

**Silver card member**

- Priority boarding
- Reserved seat
- Free miles

Points earned: 770

Example of output if the distance is 1 560 kilometres and the passenger has a gold flyer card:

Question1\_5

Distance in kilometres

Flyer Card

☐ Silver

☒ Gold

☐ Platinum

**Gold card member**

- Priority boarding
- Reserved seat
- Access to business class lounge
- Free miles and bonus miles
- Priority security checks

Points earned: 1209

(11)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Make a printout of the code if required.

**TOTAL SECTION A: 50**

**SECTION B****QUESTION 2: OBJECT-ORIENTATED PROGRAMMING**

Airlines store details of flights and passengers to assist staff in verifying information promptly.

Do the following:

DELPHI PROGRAMMERS	JAVA PROGRAMMERS
<ul style="list-style-type: none"> <li>Open the incomplete project file called <b>Question2_P.dpr</b> and the class called <b>Flight_U.pas</b>.</li> <li>Enter your examination number as a comment in the first line of both files <b>Question2_U.pas</b> and <b>Flight_U.pas</b>.</li> </ul>	<ul style="list-style-type: none"> <li>Open the incomplete classes called <b>Question2.java</b> and <b>Flight.java</b> in the folder <b>Source Packages (src)</b>, <b>Question2Package</b>.</li> <li>Enter your examination number as a comment in the first line of both classes <b>Question2.java</b> and <b>Flight.java</b>.</li> </ul>

- Complete the code for this program as specified in QUESTION 2.1 and QUESTION 2.2 below.

- 2.1 A partially completed object class called **TFlight/Flight** has been provided. It contains the attributes of a flight and the code for one method.

Complete the code in the given flight class called **TFlight/Flight**, as described in QUESTION 2.1.1 to QUESTION 2.1.6 that follow.

The table below contains descriptions of the attributes of a flight object.

NAMES OF ATTRIBUTES		DESCRIPTION
Delphi	JAVA	
fFlightNumber	flightNumber	Flight number
fCity	city	Name of the destination city
fDate	date	Date of the flight in the format YYYY/MM/DD
fNumPassengers	numPassengers	Number of passengers booked on the flight

- 2.1.1 Write code for a constructor to do the following:
- Receive the flight number, destination city and date of the flight as parameters.
  - Assign the relevant parameter values to the respective attributes.
  - Call the given **setNumPassengers** method to set the **fNumPassengers/NumPassengers** attribute to 0. (5)
- 2.1.2 Write accessor methods for the **fFlightNumber/flightNumber** and **fNumPassengers/numPassengers** attributes. (4)
- 2.1.3 Write a method called **increasePassengers** to increase the attribute of the number of passengers by 1. (2)
- 2.1.4 Write a method called **calcPercBooked** that will receive the maximum number of passengers for the flight as a parameter and calculate and return the percentage of seats booked.

Use the formula:

$$\text{passengers booked} / \text{maximum number of passengers} * 100 \quad (4)$$

- 2.1.5 Write a **toString** method to display the details of a flight in the following format:

```
Flight number: <flight number>
Destination: <city>
Departure date: <date>
Number of passengers booked: <number of passengers booked>
```

Example of output:

```
Flight number: SA528
Destination: Johannesburg
Departure date: 2017-07-22
Number of passengers booked: 13
```

(5)

## 2.2 Graphical user interface (GUI):

**Data supplied:**

A text file called **DataQ2.txt** has been supplied. Each line of text in the file contains data of a passenger booked for a flight in the following format:

<code>&lt;flight number&gt;-&lt;passenger number&gt;four spaces&lt;passenger name&gt;</code>
--

Example of the first six lines of text stored in the text file:

SA528-01	Gregory Thomas
SA528-02	Henry Kensington
MA230-01	Sebastian Johnson
KU137-01	Henrietta Botha
BA630-01	Samson Nduli
BA630-02	Mary Nduli

The data in the first two lines of text can be interpreted as follows:

**Line 1:** On flight **SA528**, the name of passenger number **1** is **Gregory Thomas**

**Line 2:** On flight **SA528**, the name of passenger number **2** is **Henry Kensington**

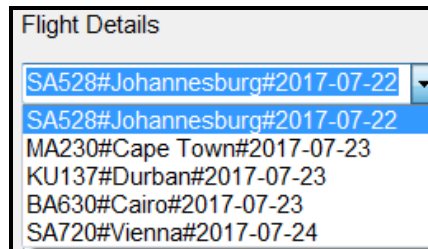
Do the following to complete the code for the buttons in the main form unit (Delphi)/GUI class (Java) as described below.

**NOTE:** The object **objFlight** has been declared in the program.

### 2.2.1 Button [Question 2.2.1]

A combo box containing the details of the different flights is given.

The contents of the combo box is as follows:



The format of each line of text in the combo box is as follows:

**<flight number>#<destination city>#<date of flight>**

The data in the first line of the combo box can be interpreted as follows:

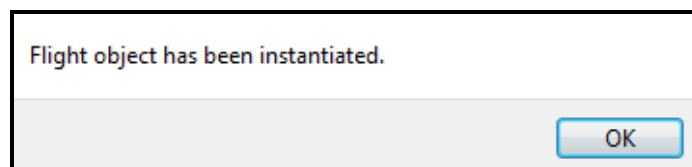
**Line 1:** Flight **SA528** to **Johannesburg** will depart on **2017-07-22**.

The user must select a flight from the combo box.

Write code to do the following:

- Use the selected flight details to instantiate the object **objFlight**.
- Enable the buttons for QUESTION 2.2.2 and QUESTION 2.2.3.
- Display a message to indicate that the flight object has been instantiated.

Example of output:



(12)

### 2.2.2 Button [Question 2.2.2]

Write code to do the following:

- Check whether or not the **DataQ2.txt** text file exists.
- If the text file does NOT exist, display a suitable message and close the program.

- Do the following if the text file exists:
  - Loop through the file.

Read a line of text (passenger details) in the file.

If the passenger read in the file is on the selected flight:

    - Display the line of text as read in the text file.
    - Use the **increasePassengers** method to update the number of passengers.
  - Use the **toString** method to display the updated information of the flight.

Example of output for flight MA230 to Cape Town:



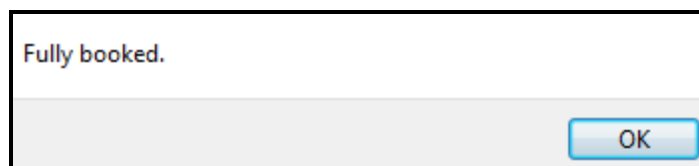
```
List of passengers
MA230-01 Sebastian Johnson
MA230-02 Jeffrey Gordan
MA230-03 Jenny Gordan
MA230-04 Mala Govender
MA230-05 Mannie Govender
MA230-06 Tyeel Govender
MA230-07 Kaitlyn Green
MA230-08 Stefan Green
MA230-09 Johannes Mbatha

Flight number: MA230
Destination: Cape Town
Departure date: 2017-07-23
Number of passengers booked: 9
```

(15)

### 2.2.3 Button [Question 2.2.3]

Use a dialog box to ask the user to enter the maximum number of passengers for the flight. Use the **calcPercBooked** method to determine the percentage of seats booked for the flight. Display a suitable message if the flight is fully booked, for example:



If the flight is NOT fully booked, display the booking percentage to ONE decimal place and use a dialog box to enter the name a new passenger. The new passenger's details must be saved to the text file and the number of passengers booked must be updated. The updated information must be displayed in the output area.



The format of all lines in the text file **DataQ2.txt** must be as follows:

**<flight number>-<passenger number>4 spaces<passenger name>**

**NOTE:** The first passenger on a flight has the number 01, the second passenger has the number 02, and so on.

Example of input from the user and output if flight MA230 is selected, the maximum number of passengers is 50 and the flight is not fully booked. Graham Barkley is added as a passenger to the flight:

List of passengers

MA230-01	Sebastian Johnson
MA230-02	Jeffrey Gordan
MA230-03	Jenny Gordan
MA230-04	Mala Govender
MA230-05	Mannie Govender
MA230-06	Tyeel Govender
MA230-07	Kaitlyn Green
MA230-08	Stefan Green
MA230-09	Johannes Mbatha
MA230-10	Graham Barkley

Flight number: MA230  
Destination: Cape Town  
Departure date: 2017-07-23  
Number of passengers booked: 10

(13)

- Enter your examination number as a comment in the first line of the class and the form.
- Save your program.
- Print the code contained in both the class and the form if required.

**TOTAL SECTION B: 60**

**SECTION C****QUESTION 3: PROBLEM-SOLVING PROGRAMMING**

The new airline company, Soaring Eagles, wants to optimise the check-in process of their passengers. The number of counters that will be opened and manned by their staff will depend on the number of customers in the queue. If a flight is delayed, the passengers on the delayed flight will be requested to queue at a separate counter.

Do the following:

<b>DELPHI PROGRAMMERS</b>	<b>JAVA PROGRAMMERS</b>
<ul style="list-style-type: none"><li>• Open the incomplete project file called <b>Question3_P.dpr</b> in the <b>Question3</b> folder.</li><li>• Enter your examination number as a comment in the first line of the main form unit called <b>Question3_U.pas</b>.</li></ul>	<ul style="list-style-type: none"><li>• Open the incomplete class called <b>Question3.java</b> in the folder <b>Source Packages (src)</b>, <b>Question3Package</b> in the <b>Question3</b> folder.</li><li>• Enter your examination number as a comment in the first line of the class called <b>Question3.java</b>.</li></ul>

Read the following sections before attempting the solution:

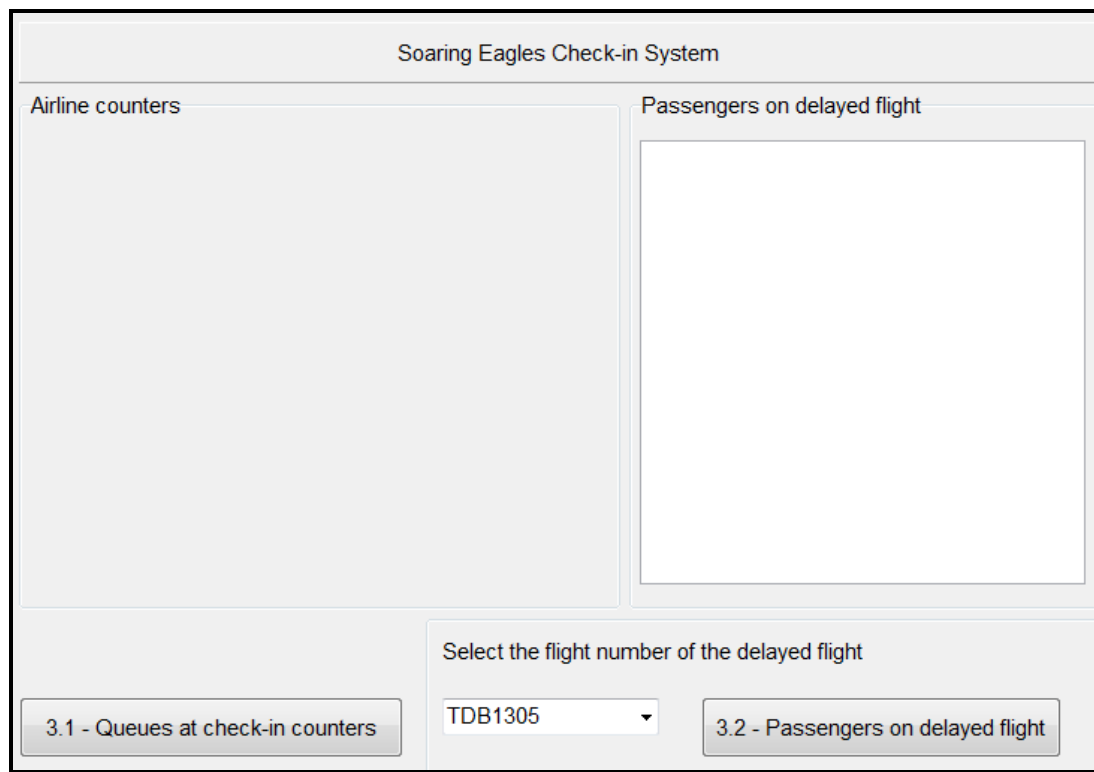
- GUI and data supplied
- Program requirements
- Mark allocation

**NOTE:**

- You may NOT modify data supplied manually. Code must be added to manipulate the data supplied according to the requirements.
- The use of good programming techniques and modular design must be applied in the design and coding of your solution.

**Supplied graphical user interface (GUI):**

The supplied GUI contains components for events that need to take place.



Use the program requirements in the questions that follow to decide on a suitable output component to be placed in the output area labelled **Airline counters** in the GUI.

**Supplied data:**

A one-dimensional array called **arrPassengers** has been declared and will contain the information of passengers who must collect their boarding passes at the check-in counters at the airport. Supplied code will populate the **arrPassengers** array with an unknown number of elements obtained from the **arrPosPassengers** array when the program is executed. The number of elements allocated to the **arrPassengers** array will be in the range from 1 to 35 and will be determined randomly.

You must use the **arrPassengers** array in your code.

The format of the data in the **arrPassengers** array is as follows:

(B – business class or E – economy class), the position of the passenger in the queue (starting at 01) and the flight number, as shown below:

**<Passenger class><position of passenger in the queue>;<flight number>**

Example of passenger information in the **arrPassengers** array:

```
E01;TDB2506
B02;TDB2506
E01;TDB1305
```

A two-dimensional array called **arrGrid** with nine rows and four columns has been declared. In your solution this array will not necessarily contain 36 elements, as the number of passengers contained in the **arrPassengers** array will determine the number of elements contained in the **arrGrid** array.

### Program requirements:

#### 3.1 Button [3.1 – Queues at check-in counters]

The program must place the passengers waiting for their boarding passes (contained in the **arrPassengers** array) in queues at the check-in counters. The number of passengers contained in the **arrPassenger** array determines the number of open check-in counters by using the following criteria:

NUMBER OF OPEN CHECK-IN COUNTERS	NUMBER OF PASSENGERS TO RECEIVE BOARDING PASSES
1	Less than 10
2	10 to 16
3	17 to 24
4	More than 24

Do the following:

- Use the **arrGrid** two-dimensional array to represent the open check-in counters and passengers allocated to these counters. Passengers must be allocated to counters as follows:
  - Business class passengers must be placed first in the queues at the check-in counters, followed by economy class passengers.
  - The passengers must be distributed as evenly as possible amongst the queues at the check-in counters, as shown in the examples of output that follow.
- Display suitable headings for the check-in counters.
- Display the class and number of the passengers.

Example of output if 16 passengers are waiting to receive boarding passes:

Counter1	Counter2
B06	B08
B10	B12
B13	B14
E01	E02
E03	E04
E05	E07
E09	E11
E15	E16

Example of output if 33 passengers are waiting to receive boarding passes:

Counter 1	Counter 2	Counter 3	Counter 4
B06	B08	B10	B12
B13	B14	B22	B23
B27	B28	B31	E01
E02	E03	E04	E05
E07	E09	E11	E15
E16	E17	E18	E19
E20	E21	E24	E25
E26	E29	E30	E32
E33			

**NOTE:** The output is determined by the number of elements that the **arrPassengers** array contains.

### 3.2 Button [3.2 – Passengers on delayed flight]

When a flight is delayed the passengers booked on the delayed flight must be removed from the queues at the open check-in counters and placed in a queue at a separate counter. A combo box with flight numbers is provided to select the flight number of a delayed flight.

The user must select the delayed flight and click on the QUESTION 3.2 button.

The program must do the following:

- Remove the information of all passengers booked on the delayed flight from the queues at the check-in counters and display their details in the output area provided.
- Update the display of the queues at the check-in counters. The passengers at the check-in counters must still be distributed evenly amongst the counters; first the business class passengers and then the economic class passengers.

Example of output if flight TDB2506 was selected from the combo box as a delayed flight and the **arrPassengers** array contains 33 elements:

Counter 1	Counter 2	Counter 3	Counter 4	Flight number: TDB2506
B06	B08	B12	B14	B10
B22	B27	B28	B31	B13
E02	E03	E07	E09	B23
E11	E16	E18	E19	E01
E20	E21	E24	E25	E04
E26	E29			E05
				E15
				E17
				E30
				E32
				E33

**Mark allocation:**

REQUIREMENTS	MAXIMUM MARKS
<b>QUESTION 3.1</b>	
Determine number of open counters and number of passengers per row	6
Allocate position according to passenger class	5
Copy information from given array to correct data structure	8
Display passengers in queues at correct number of counters	7
<b>QUESTION 3.2</b>	
Determine and display passengers on the delayed flight	6
Remove passengers on delayed flight from queues	6
Update information in data structures and output	2

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Make a printout of the code if required.

**TOTAL SECTION C: 40**  
**GRAND TOTAL: 150**