

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



SA EXAM
PAPERS



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2018

MARKS: 150

TIME: 3 hours

This question paper consists of 17 pages.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into THREE sections. Candidates must answer ALL the questions in 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 specific to Delphi programming language.
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 Delphi 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. Make sure 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:

Candidates must use the file **DataENGMarch2018.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the extract button.
- Enter the following password: **Sp@cE&18**

Once extracted, the following list of files will be available in the folder **DataENGMarch2018**:

SUPPLIED FILES**Question 1:**

Image1.jpg
Image2.jpg
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

Question 2:

Centaurus.jpg
Crux.jpg
Orion.jpg
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas
Scorpio.jpg
Star_U.pas
StarData.txt

Question 3:

Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas

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

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The program has no functionality currently.
- The program contains FIVE tab sheets with different unrelated questions.
- Follow the instructions below to complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5.

1.1 Button [1.1 – Total area]

The user must use the edit boxes provided to enter the radius of the circle and the base and height of one of the triangles in the figure.

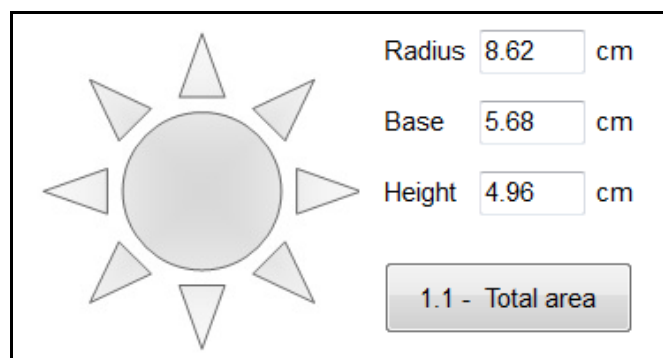
Write code to do the following:

- Extract the values entered by the user.
- Calculate the area of the circle, the area of the eight triangles in total and the total area of the figure using the following formulae:

Area of circle = $\pi \times \text{radius}^2$ where the value of $\pi = 3.14159$
Area of a triangle = $\frac{1}{2} \times \text{base} \times \text{height}$

- Display the area of the circle, area of the eight triangles in total and the total area of the figure in the output component provided. The total area must be formatted to TWO decimal places.

Example of input:



Radius	<input type="text" value="8.62"/>	cm
Base	<input type="text" value="5.68"/>	cm
Height	<input type="text" value="4.96"/>	cm

Example of output:

Area of circle = 233.433959996
Total area of triangles = 112.6912
Total area = 346.13

(12)

1.2 Button [1.2 – Next blue moon]

The first blue moon was seen in the year 1862. Thereafter, a blue moon was seen every three years.

Extract the year in which the first blue moon was seen from the **lblInfo** label. Use the year from the system date and the year extracted from the label to determine the year when the next blue moon will appear (excluding the current year). Display the output in the edit box provided.

Example of output:

Next appearance of a blue moon 2021

(9)

1.3 Button [1.3 – Highest common factor]

The user must enter two integer numbers. The highest common factor is the largest number that can divide into the two numbers without a remainder.

Write code to do the following:

- Extract the numbers entered by the user from the edit boxes provided.
- Determine the highest common factor of the two numbers.
- Display the highest common factor in the edit box provided.

Example of input and output:

Number 1	<input type="text" value="12"/>
Number 2	<input type="text" value="15"/>
<input type="button" value="1.3 - Highest common factor"/>	
	<input type="text" value="3"/>

(10)

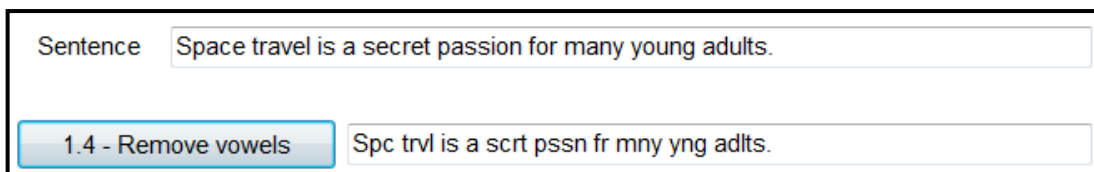
1.4 Button [1.4 – Remove vowels]

The user must enter a sentence in the edit box provided.

Write code to do the following:

- Delete ALL the vowels from the sentence entered by the user, except where the vowel is the first letter of the word.
- Display the sentence after deleting the vowels in the edit box provided.

Example of output:



The screenshot shows a web interface with a label 'Sentence' and a text input box containing 'Space travel is a secret passion for many young adults.' Below this is a button labeled '1.4 - Remove vowels'. To the right of the button is another text input box containing the result: 'Spc trvl is a scrt pssn fr mny yng adlts.' The vowels have been removed from the original sentence, except for the first letter of each word.

(11)

1.5 Button [1.5 – Slide show]

A space exhibition centre presents a slide show to visitors in their conference centre which seats 100 people. Groups of people can attend the slide show as long as there are seats available for **all the people** in the group. Groups who can be accommodated will each be assigned a number, starting with the value 1 for the first group, 2 for the second group, and so on. If large groups are rejected, smaller groups can still be accommodated until all the seats are taken.

Write code to do the following:

- Use an input box to prompt the user to enter the number of people in the group. The user must be able to keep on entering the number of people for new groups until the total number of attendees reaches 100.
- For each group that is accepted, display the group number and the number of people in the group.
- If the number of people in the group exceeds the number of seats available, display a suitable message indicating that the group cannot be accommodated. The number of seats still available must be part of the message.

NOTE: Even though some of the groups will not be accepted to attend the slide show, the program must continue to allow the user to enter groups until a total of 100 people are accepted to attend the slide show.

Example of input and output if the number of people in the first group is 27:

Enter the number of people in the group

Group number	Number of people
1	27

Example of input and output if the number of people in the second group is 34:

Enter the number of people in the group

Group number	Number of people
1	27
2	34

Example of output if the next group that wants to attend the slide show consists of 50 people:

Cannot accept a group of 50 people
Number of seats available is 39

OK

(12)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION A: 54

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

A constellation is a group of related stars that covers the night sky. Some stars are considered to be navigational, while others are passive. A navigational star is used to assist with direction and movement.

Do the following:

- Open the incomplete program in the **Question 2** folder.
- Open the incomplete object class **Star_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question2_U.pas** file and the **Star_U.pas** file.
- Compile and execute the program. Currently the program has no functionality.

The following user interface is displayed:

The screenshot shows a graphical user interface for a program titled "STARS". At the top left, the title "STARS" is displayed in a large, bold, black font. Below the title, there is a label "Select a star" followed by a standard Windows-style dropdown menu. In the center of the interface is a large, empty rectangular box with a thin black border. Below this box is a button with the text "2.2.1 - Instantiate object". In the bottom right corner, there is a button with a small icon of a window and the text "Close". The entire interface is set against a light gray background.

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

- 2.1 An incomplete object class called **TStar**, which represents a celestial star, has been provided.

The declaration of the attributes and an accessor method for the `fName` attribute (`getName`) has been provided.

The attributes for the **Star** object have been declared as follows:

NAMES OF ATTRIBUTES	DESCRIPTIONS
<code>fName</code>	Name of the star
<code>fMagnitude</code>	The magnitude of a star is related to its brightness. The magnitude of the brightest star is -1. As the magnitude of a star increases, the star gets dimmer. Example: The magnitude of a dim star, such as Mimosa, is 1.25.
<code>fDistance</code>	The distance of a star from Earth is an integer value measured in light years. Example: The distance of the star Mimosa from Earth is 279 light years.
<code>fConstellation</code>	The name of the constellation the star belongs to
<code>fNavigationalStatus</code>	A Boolean value which indicates whether the star is regarded as a navigational star or not

- 2.1 Complete the code in the object class, as described in QUESTION 2.1.1 to QUESTION 2.1.5 below.

- 2.1.1 Write code for a constructor method that will receive the name of the star, its magnitude, its distance from Earth and the constellation it belongs to as parameters. Set the FOUR respective attributes to the received parameter values and initialise the **fNavigationalStatus** attribute to 'false'. (4)
- 2.1.2 Write code to create an accessor method for the constellation attribute **fConstellation**. (2)
- 2.1.3 Write code for a mutator method called **setNavigationalStatus**, which will receive a Boolean value as a parameter and set the navigational status attribute to the received value. (3)

- 2.1.4 Write code for a method called **determineVisibility** that will determine and return a description of the visibility of the star. The visibility of a star depends on its distance from Earth in light years and its magnitude.

Use the following criteria to determine the description of visibility that applies to a star:

DISTANCE	MAGNITUDE	DESCRIPTION OF VISIBILITY
Fewer than 80 light years	Any value	Clearly visible
Between 80 and 900 light years (inclusive)	Up to 2	Hardly visible to the naked eye
	Larger than 2	Visible by means of standard optical aid
More than 900 light years	Any value	Only visible by means of specialised optical aid

(11)

- 2.1.5 Write code to create a **toString** method which returns a string formatted as follows, depending on whether the star is a navigational star or not:

<name of star> belongs to the **<constellation>** constellation.

The star has a magnitude of **<magnitude>** and is **<distance from Earth>** light years away from Earth.

If the star is a navigational star, add the following line:

<name of star> is a navigational star.

If the star is NOT a navigational star, add the following line:

<name of star> is a passive star.

(6)

- 2.2 An incomplete unit, **Question2_U**, has been provided which contains code for the following:

- The declaration of the object variable **objStarX**
- An array called **arrNavigationStars** which contains the names of stars that are navigational stars.

A text file called **StarData.txt** contains the data of an unknown number of stars (both passive and navigational). The details of each star appears in the following format in the file:

```
<common name of the star>
<magnitude of the star>
<distance of the star from Earth in light years>
<name of the constellation the star belongs to>
<blank line>
```

Example of the details of the first two stars in the text file:

```
Acrux
1.25
321
Crux

Mimosa
1.25
279
Crux
```

Do the following to complete the code for the buttons in the main form unit, as described below:

2.2.1 Button [2.2.1 – Instantiate object]

The user is required to select the name of a star in the combo box.

Write code to do the following:

- Extract the name of the selected star from the combo box.
- Use a conditional loop and search in the text file for the name of the selected star. The loop must stop when the name of the star has been found in the file.

If the name of the star has been found, do the following:

- Instantiate a **TStar** object using the **objStarX** object variable that has been declared globally as part of the given code.
- Test whether the star is a navigational star using the **arrNavigationStars** array and set the value for the navigational status attribute accordingly.
- Enable the **pnlButtons** panel.

If the name of the star has NOT been found in the text file, do the following:

- Display a message to indicate that the star was not found.
- Disable the **pnlButtons** panel.

(24)

2.2.2 Button [2.2.2 – Display]

Display the details of the star in the rich edit component **redQ2** using the **toString** method.

Load the image of the constellation that the star belongs to into the **imgQ2** image component. The file name of the image to be displayed is the name of the constellation the star belongs to. The image files have the extension **.jpg**.

Example of output if the selected star is Mimosa:

Select a star Mimosa

Mimosa belongs to the Crux constellation.

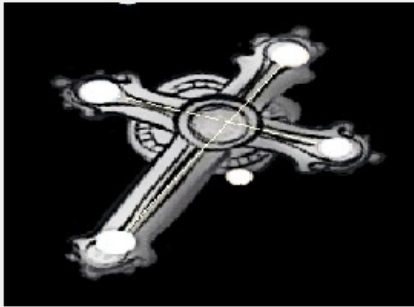
The star has a magnitude of 1.25 and is 279 light years away from Earth.

Mimosa is a passive star.

2.2.1 - Instantiate object

2.2.2 - Display

2.2.3 - Visibility



(3)

2.2.3 Button [2.2.3 – Visibility]

The brightness and visibility of a star is dependent on the magnitude and the distance of the star from Earth. Call the relevant methods to display the name and visibility of the star.

Example of output if Mimosa has been selected:

```
Star:      Mimosa
Visibility: Hardly visible to the naked eye
```

Example of output if Rigil Kent has been selected:

```
Star:      Rigil Kent
Visibility: Clearly visible
```

Example of output if Mintaka has been selected:

```
Star:      Mintaka
Visibility: Only visible by means of specialised optical aid
```

(3)

- Ensure that your examination number has been entered as a comment in the first line of the object class and the form class.
- Save your program.
- Print the code of both the object class and form class, if required.

TOTAL SECTION B: 56

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

A new game called Galaxy Explore is planned and needs to be developed. The purpose of the game is to prepare a grid with a number of randomly placed planets that are not visible to the player. The player must then guess the position of the planets on the grid. The grid will be referred to as the 'Game board'.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Enter your examination number as a comment in the first line of the **Question3_U.pas** file.
- Compile and execute the program. Currently the program has no functionality.

Supplied GUI:

The GUI below represents the interface of the program.

The GUI is a standard Windows-style application window. On the left, there's a panel titled 'Levels of difficulty' containing three radio buttons: 'Level 1' (selected), 'Level 2', and 'Level 3'. Below this panel is a large rectangular area labeled 'Game board'. To the right of the 'Game board' is a vertical stack of controls. At the top is a 'Number of guesses' text box. Below it is an 'Incorrect guesses' list box. Further down are three buttons: '3.1 - Start game', '3.2 - Play' (which contains two dropdown menus for 'Row' and 'Column', both currently showing '1'), and '3.3 - Reveal planets'. At the bottom right is a 'Close' button with a small icon of a window with an 'X'.

- Complete the code for each question, as described in QUESTION 3.1 to QUESTION 3.3.

NOTE:

- Good programming techniques and modular design must be applied in the design and coding of your solution.
- You may NOT change the code provided.

Supplied code:

The program contains the following code for the declaration of a two-dimensional array called **arrGame**:

```
arrGame: array [1..9, 1..9] of char;
```

The program must do the following:

- Populate the two-dimensional array when the game starts
- Allow the user to guess the positions of invisible planets placed randomly on the game board
- Determine whether the player has won or lost and terminate the game. The player wins when he/she identifies two planets on the game board within five guesses.

3.1 Button [3.1 – Start game]

A dash character (-) represents an open space in the two-dimensional array **arrGame** and a hash character (#) represents a planet.

The two-dimensional array must first be populated with open-space characters. Its content must be displayed in the game board area.

The level of difficulty selected from the radio group **rgbQ3** determines the number of planets to be placed randomly in **arrGame**. The following applies:

- Difficulty level 1: 50 positions in the array must be replaced by planets (#)
- Difficulty level 2: 40 positions in the array must be replaced by planets (#)
- Difficulty level 3: 30 positions in the array must be replaced by planets (#)

The value '0' must be displayed on panel **pnlQ3NumberOfGuesses**, which indicates the total number of guesses.

The '**Play**' button must be enabled and the rich edit components cleared.

NOTE: The positions of the planets on the game board should NOT be visible (not displayed) to the player.

Example of output when the **Start game** button is clicked:

The screenshot shows a game interface with the following elements:

- Levels of difficulty:** Three radio buttons for Level 1 (selected), Level 2, and Level 3.
- Number of guesses:** A text box displaying '0'.
- Incorrect guesses:** A large empty rectangular area for displaying incorrect guesses.
- Game board:** A 10x10 grid of dashed lines representing the game board.
- Row and Column selection:** Two dropdown menus, both set to '1'.
- Buttons:**
 - 3.1 - Start game:** A button at the top right.
 - 3.2 - Play:** A button below the row and column dropdowns.
 - 3.3 - Reveal planets:** A button below the 'Play' button.
 - Close:** A button at the bottom right with a close icon.

(22)

3.2 Button [3.2 – Play]

The player must guess the position of a planet by selecting a row number and a column number from the combo boxes provided. Then click the '**Play**' button to see whether the position of a planet was guessed correctly.

If the position of a planet is guessed correctly:

- Update the display to show the position of the planet (#) that was guessed correctly.
- Update the number of correct guesses on the panel **pnIQ3NumberOfGuesses**.

HINT: Replace the planet character (#) that was guessed correctly with another character to show which planets were identified during the game play session. This information is required for Button 3.3 ('Reveal planets').

If the position guessed is NOT the position of a planet:

- Display the row and column values of the incorrect guess in the **redQ3Incorrect** output area.

The player must be able to guess the positions of planets repeatedly until he/she wins or loses the game.

NOTE: A game is won as soon as the positions of two planets are correctly guessed within the allowed five guesses.

Use a message box to display a suitable message based on the outcome of the game, either 'Game won' or 'Game lost'.

Disable the play button when the game is over.

Example of output if the positions of two planets were guessed correctly within two guesses:

The screenshot shows a game interface with the following elements:

- Levels of difficulty:** Level 1 is selected (radio button).
- Number of guesses:** A text box containing the number 2.
- Incorrect guesses:** An empty text box.
- Game board:** A 10x10 grid with two '#' symbols at row 3, column 5 and row 4, column 3.
- Row/Column selection:** Row is set to 3 and Column is set to 5.
- Buttons:** '3.1 - Start game' (disabled), '3.2 - Play' (disabled), '3.3 - Reveal planets' (disabled), and a 'Close' button with a small icon.

Example of output if the player lost the game. The position of only one planet was guessed correctly within five guesses:

The screenshot shows the same game interface as above, but with the following changes:

- Number of guesses:** A text box containing the number 5.
- Incorrect guesses:** A text box containing the list: R1, C7; R8, C1; R9, C3; R3, C2.
- Game board:** A 10x10 grid with one '#' symbol at row 3, column 5.
- Row/Column selection:** Row is set to 3 and Column is set to 2.
- Buttons:** '3.1 - Start game' (disabled), '3.2 - Play' (disabled), '3.3 - Reveal planets' (disabled), and a 'Close' button with a small icon.

(13)

3.3 Button [3.3 – Reveal planets]

Write code to display the game board with all the randomly placed planets revealed.

Example of the output if the planets were randomly placed but the player has not played yet:

Levels of difficulty

☒ Level 1

☐ Level 2

☐ Level 3

Number of guesses

0

Incorrect guesses

Game board

```

- - # # - - # # -
- # - # - # # # #
# # - - - # # - #
# # - # # - - # #
# # - # # - - # #
- - # # # # - -
# # - # # - - # #
# # # # - - - # #

```

Row 1 Column 1

3.1 - Start game

3.2 - Play

3.3 - Reveal planets

Close

Example of output if the player won by identifying the positions of two planets within two guesses:

Levels of difficulty

☒ Level 1

☐ Level 2

☐ Level 3

Number of guesses

2

Incorrect guesses

Game board

```

- - # # - - # # -
- # - $ - # # # #
# # - - - # # - #
# # $ - - - # # -
# # - # # - - # #
- - # # # # - -
# # - # # - - # #
# # # # - - - # #

```

Row 4 Column 3

3.1 - Start game

3.2 - Play

3.3 - Reveal planets

Close

NOTE: The array in the sample output may be different because the indices for the planet symbols are randomly generated.

(5)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION D: 40
GRAND TOTAL: 150