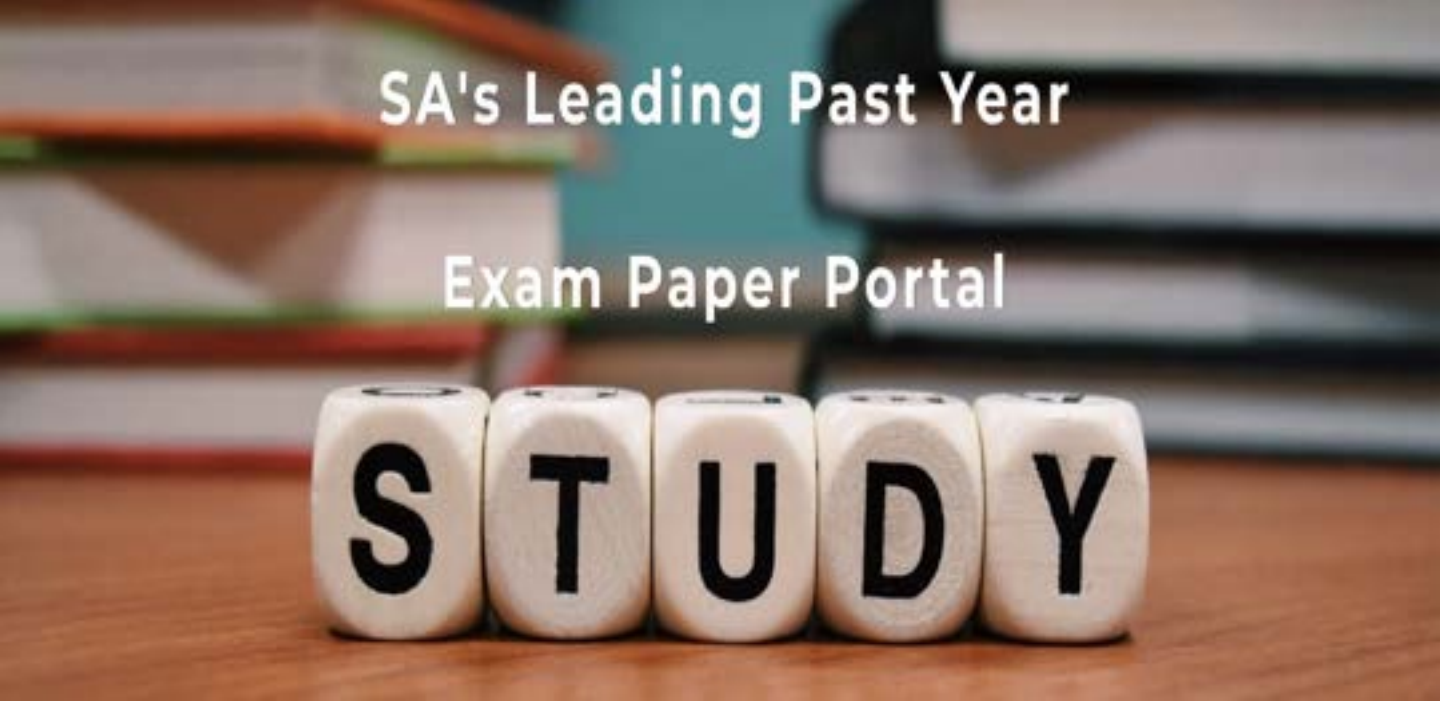


You have Downloaded, yet Another Great Resource to assist you with your Studies ☺

Thank You for Supporting SA Exam Papers

Your Leading Past Year Exam Paper Resource Portal

Visit us @ www.saexampapers.co.za

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2011

MARKS: 120

TIME: 3 hours

This question paper consists of 32 pages, 3 annexures and an information sheet.

INSTRUCTIONS AND INFORMATION

1. This is a three-hour examination. 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.

2. Answer SECTION A (for Delphi programmers) OR SECTION B (for Java programmers).

3. You require the files listed below in order to answer the questions. They are EITHER on a stiffy disk OR CD issued to you, OR the invigilator/teacher will tell you where to find them on the hard drive of the workstation you are using OR in a network folder:

QUESTION 1

Delphi:	**Java:**
Question1_P.dpr	Recruitment.class
Question1_P.res	RecruitmentDB
Question1_U.dfm	tblAgencies.txt
Question1_U.pas	tblClients.txt
RecruitmentDB	TestRecruitment.java
tblAgencies.txt	
tblClients.txt	

QUESTION 2

Delphi:	**Java:**
Jobs.txt	City.java
Question2_P.dpr	Jobs.txt
Question2_P.res	TestCity.java
Question2_U.dfm	
Question2_U.pas	
uCity.pas	

QUESTION 3

Delphi:	**Java:**
Question3_P.dpr	TestVisitors.java
Question3_P.res	
Question3_U.dfm	
Question3_U.pas	

If a disk (CD or stiffy) containing the above files was issued to you, write your examination number on the label.

4. Save your work at regular intervals as a precaution against power failures.

5. Save ALL your solutions in folders with the number of the question and your examination number as the name of the folder, for example Quest2_3020160012.

6. Type in your examination number as a comment in the first line of each program.

7. Read ALL the questions carefully. Do not do more than the questions require.

8. During the examination, you may use the manuals originally supplied with the hardware and software. You may also use the HELP functions of the software. **Java candidates may use the Java API files. You may NOT refer to any other resource material.**

9. At the end of this examination session you will be required to hand in the disk or CD with all your work saved on it OR you must make sure that all your work has been saved on the hard drive/network as explained to you by the invigilator/teacher. Ensure that all files can be read.

10. The invigilator will inform you whether you should hand in printouts of the programming code of all the questions you have done.

11. If printing is required, all printing of programming questions that you have done will take place within an hour of the completion of the examination.

12. Complete the information sheet attached to this question paper and hand it in at the end of this examination session.

SECTION A

Answer ALL the questions in this section only if you studied **Delphi**.

SCENARIO

People are always looking for better job opportunities. There are many recruitment agencies as well as training institutions involved in the employment industry. All of these agencies and institutions need appropriate software to be able to operate efficiently.

QUESTION 1: DELPHI PROGRAMMING AND DATABASE

An independent career company keeps track of a list of recruitment agencies and the details of their clients. A database named **RecruitmentDB** has been developed to store their data. An incomplete program has been developed to process queries on the data in the given database. Your task will be to complete this program.

The database named **RecruitmentDB** as well as an incomplete Delphi project named **Question1_P.dpr** are in the folder named **Question 1 Delphi**.

NOTE: The design of the tables in the **RecruitmentDB** database and sample data for this question can be found in **ANNEXURE A: Table Description Sheet**.

NOTE: If you cannot use the database provided follow the instructions in **ANNEXURE B** to create the database before you answer any of QUESTIONS 1.1 to 1.7.

NOTE: Make a copy of the **RecruitmentDB** database BEFORE you start with the solution. You will need a copy of the original database to be able to test your program thoroughly.

Do the following:

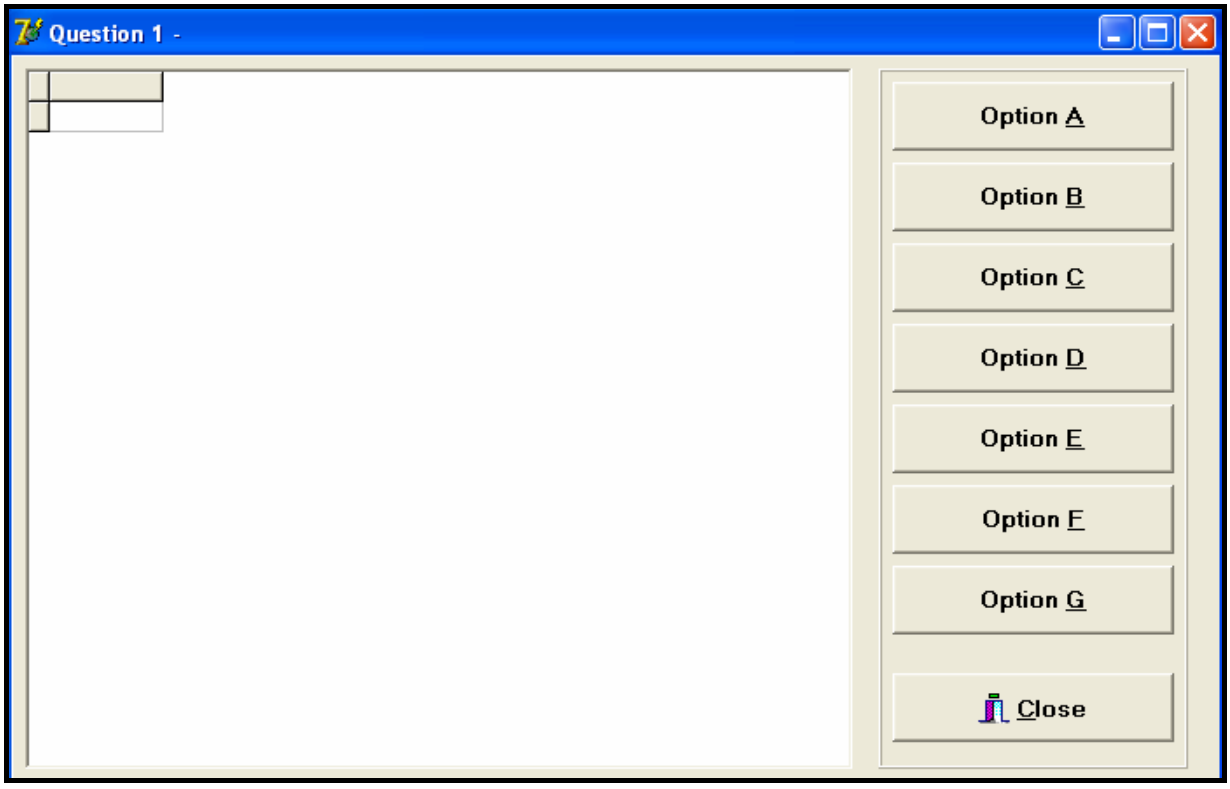
- Rename the folder **Question 1 Delphi** as **Quest1_X**, where X should be replaced with your examination number.
- Open Delphi and then open the file **Question1_P.dpr** in the folder **Quest1_X.** The program displays eight buttons as well as a DBGrid that will be used as an output component (see example on the next page).
- Add your examination number to the right of 'Question 1 –' in the caption of the form.
- Go to File/Save As … and save the unit as **Question1_UXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Go to File/Save Project As … and save the project as **Question1_PXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- The program should be able to connect to the database named **RecruitmentDB**. When you do QUESTION 1.1 (which follows on the next page) and you find that the connectivity is not in place, use the steps supplied in **ANNEXURE C** to establish connection with the database.

NOTE: If your program cannot connect to the database, make sure that the database file **RecruitmentDB** is in the same folder as your program. Your program will not work if the database file is in a folder other than the folder containing the program. If this is the case, copy the database file **RecruitmentDB** into the same folder as your program.

NOTE: If you still cannot establish connectivity with the database when you execute the program, you must still do the SQL code and submit it for marking.

Marks will only be awarded for the programming code that contains the SQL statements in the unit named Question1_UXXXX.

When you execute the program, the interface below will be displayed. An error will be displayed when the buttons are clicked, due to the incomplete SQL statements.



Do the following:

Complete the SQL statements in **Question1_UXXXX.pas** for each button as indicated in QUESTIONS 1.1 to 1.7 below. The code to execute the SQL statements and to display the contents in the DBGrid has been given to you. You only need to complete the SQL statements and some input statements, as required in the **Question1_UXXXX** unit.

1.1 A student needs to place her CV with a reputable agency. She needs a list of agencies that indicates the number of clients they have previously placed and other important information about the company. Complete the code for the **Option A** button by formulating an SQL statement to display **all the details** of agencies stored in the **tblAgencies** table. Display the results in descending order according to the **NumPrevPlacements** field.

Example of the output for the first seven records:

AgencyName	Province	International	NumPrevPlacements
Best Jobs Inc	KwaZulu-Natal	False	29058
Nellie Agencies	Gauteng	False	28167
Instant Emp Int	Western Cape	True	25627
KZN Recruitment Centre	KwaZulu-Natal	False	25145
Ronnies Agencies	Mpumulanga	False	24600
Boss Agencies	KwaZulu-Natal	True	24398
Only the Best	Mpumulanga	False	24234

:

(4)

1.2 A corporate company would like to provide bursaries to all clients that are part-time and earn a salary of less than R15 000. Complete the code for the **Option B** button by formulating an SQL statement to display **Name**, **Surname** and **Salary** of all clients that are part-time and earn a salary of less than R15 000.

Example of the output:

Name	Surname	Salary
Brianna	Amena	14328
Magee	Noah	14260
Randall	Lawrence	12526
Brent	Arden	10706
Ciara	Martina	12349
Grace	Sydnee	11215
Claudia	Logan	14724

(4)

1.3 Clients complained that there are too few agencies that operate internationally. The career company needs to provide the clients with the number of agencies that operate internationally. Complete the code for the **Option C** button by formulating an SQL statement to display the number of recruitment agencies that operate internationally.

Example of the output:

The number of agencies that offer international jobs are
11

(4)

1.4 An agency that places a client will receive 10% of the client's salary as commission. Complete the code for the **Option D** button by formulating an SQL statement that will display the **Name**, **Surname**, **Salary** and a **calculated field** for **commission**, rounded off to TWO decimal places. Name the calculated field **AgentComm**.

Example of the output for the first seven records (on the next page):

	Name	Surname	Salary	AgentComm
	Lucas	Florence	17289	1728.90
	Illana	Alyssa	27855	2785.50
	Keefe	Lawrence	28940	2894.00
	Adam	Isaiah	19142	1914.20
	Brianna	Amena	14328	1432.80
	Demetrius	Kaseem	24002	2400.20
	Garrett	Halla	17481	1748.10

: (5)

1.5 A new agency has been established. This agency operates locally and does not offer clients international job opportunities. Complete the code for the **Option E** button by formulating an SQL statement that will **insert** the following record:

AgencyName: Jobs Unlimited
Province: Western Cape
NumPrevPlacements: 0

Example of the output:

HINT: Run **Option A** to verify that the record has been inserted. (4)

1.6 The provinces of Western Cape and Gauteng would like to certify all the clients who have been placed at agencies in these provinces. Complete the code for the **Option F** button by formulating an SQL statement to display the **Name**, **Surname**, **AgencyName** and **Province** of clients who have been placed by agencies in these two provinces.

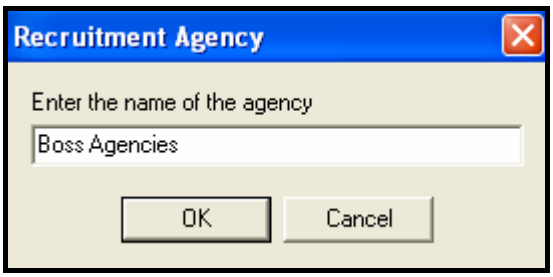
Example of the output (on the next page):

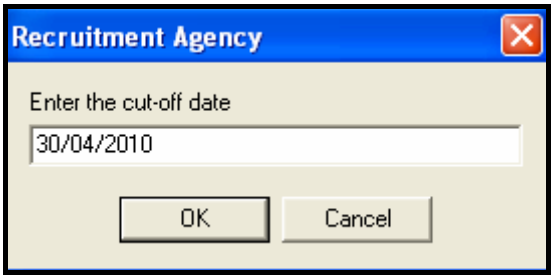
Name	Surname	AgencyName	Province
Randall	Lawrence	Apollo International	Western Cape
Ciara	Martina	Apollo International	Western Cape
Adam	Isaiah	First Choice	Western Cape
Richard	Quyn	First Choice	Western Cape
Colin	Elmo	First Choice	Western Cape
Colby	Lionel	First Choice	Western Cape
Todd	Harper	First Choice	Western Cape
Judith	Ashely	First Choice	Western Cape
Blaze	Ashton	First Choice	Western Cape
Sawyer	Elaine	Instant Emp Int	Western Cape
Damon	Teagan	Instant Emp Int	Western Cape
Nichole	Clark	Instant Emp Int	Western Cape
Claudia	Logan	Instant Emp Int	Western Cape
Demetrius	Kaseem	Nellie Agencies	Gauteng
Jenna	Chastity	Nellie Agencies	Gauteng
Grace	Sydnee	Nellie Agencies	Gauteng
Genevieve	Nicholas	Nellie Agencies	Gauteng

(6)

1.7 The career company must be able to view lists of clients who have been placed by agencies before any specific cut-off date. Complete the code for the **Option G** button by asking the user to enter the name of the agency and the cut-off date. Formulate an SQL statement to display the **Name**, **Surname** and **DatePlaced** of all the clients who have been placed by the specified agency before the specified cut-off date.

Example of the input and output of all the clients placed by **Boss Agencies** before **30/04/2010**:





Name	Surname	DatePlaced
Lucas	Florence	2010/02/21
Chaney	Akeem	2010/02/06
Lara	Norman	2010/04/01

NOTE: Any format of the date will be accepted. (8)

- Enter your examination number as a comment in the first line of the file named **Question1_UXXXX.pas** containing the SQL statements.
- Save the unit **Question1_UXXXX** and the project **Question1_PXXXX** (File/Save All).
- A printout for the code of the **Question1_UXXXX.pas** file will possibly be required (see Instruction 10 on page 3).

[35]

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

A recruitment agency has identified a need where graduates from tertiary education institutions often want to know which city offers the best job opportunities for them. This agency has asked each of its branches in various cities to supply them with a list of job opportunities available in their cities and the salaries associated with these job opportunities.

The agency needs software which will allow graduates to analyse the job opportunities of a particular city to decide whether they will be available for employment in that city.

The agency identified two categories of job opportunities for graduates, namely **diploma job opportunities** and **degree job opportunities**. A city is a good choice for a graduate if

- the number of job opportunities for their category (diploma or degree) is higher than the number of job opportunities available for the other category; and
- the average monthly salary of all job opportunities in the city is higher than the monthly salary they require to live comfortably.

EXAMPLE:

John is a graduate and has a **degree** in engineering. He requires a salary of at least **R10 000** per month to live comfortably.
Durban has five diploma and ten degree job opportunities.
The average salary of the job opportunities in Durban is R12 500 per month.

This makes Durban a suitable city for John in which to look for employment, since

- the number of degree job opportunities is higher than the number of diploma job opportunities; AND
- the average salary of the job opportunities in Durban is higher than the salary John requires.

The agency only deals with the following specific careers in each category:

Diploma job opportunities: secretary, mechanic, electrician, beautician, nurse, plumber
Degree job opportunities: doctor, programmer, architect, teacher, lawyer, engineer

The branch of the agency in each city has compiled a text file with specific job opportunities available in that city. The data stored in the text file named **Jobs.txt** contains the information on the **job opportunities available in a single city**. The format of the information in the text file is as follows (on the next page):

Name of the city
Job Title
Salary
Job Title
Salary
etc.

An example of the data in the text file is as follows:

Cape Town
Mechanic
6500
Architect
14500
Doctor
25000
etc.

You are required to complete the given program in the folder **Question 2 Delphi** (as indicated in QUESTION 2.1 and QUESTION 2.2 that follow below) to process **ONE** text file of a single city.

Do the following:

- Rename the folder **Question 2 Delphi** as **Quest2_X** (where X must be replaced with your examination number).
- Open Delphi and then open the file **Question2_P.dpr** in the folder **Quest2_X**.
- Go to File/Save As … and save the unit as **Question2_UXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the unit **uCity.pas**.
- Go to File/Save As … and save the unit as **uCityXXXX.pas** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Go to File/Save Project As … and save the project as **Question2_PXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).

You have been given a program containing incomplete code in the **Question 2 Delphi** folder. Open the program and complete the code according to the following instructions:

2.1 The object class named **uCityXXXX.pas** contains fields and methods that describe the state of available job opportunities in a single city. Note the following:

- All the fields in this class should be private and all methods public.
- Parts of this class have been placed in comment symbols in order to get it to compile. Remove the comment symbols from the statements during the completion of the program.
- Modify code in the given methods and write some additional methods, as described on the next page.

2.1.1 Create private fields **with the following names** to hold the data of a city. Choose appropriate data types for the fields:

- **cityName** – name of the city
- **diplomaJobs** – the total number of available diploma job opportunities in the city
- **degreeJobs** – the total number of available degree job opportunities in the city
- **salaryTotal** – the total of the salaries of all of the available degree and diploma job opportunities in the city added together

NOTE: It is important that you **use the field names given** in bold above in order for the given code to work correctly. (3)

2.1.2 You have been provided with a **default constructor** method. Write an **additional constructor method** which receives one parameter for the city's name. Initialise the name field using the parameter value and initialise the other fields to a value of zero.

NOTE: Add the word 'overload' to the declaration of both constructors in the 'Interface' section of the class as soon as the code for the second constructor has been done, for example:

Public Create: overload; (3)

2.1.3 You have been provided with two methods (procedure methods) that do not return any information named **addDipJob** and **addDegJob** that each receives a salary as a parameter. When called, these methods increase the number of diploma and degree job opportunities by one respectively and add the value of the salary parameter to the total salary field.

For example, calling the **addDegJob** method with a parameter value of **2000** will increase the number of degree job opportunities in the city by one and add R2 000 to the salary total for that city.

Do the following:

- Remove the comment symbols from the code of each of these two given methods.
- Write a method (function method) called **averageSalary** which calculates and returns the average salary for the city, rounded off to TWO decimal places. The average salary is calculated as follows:

  total salary/(number of diploma job opportunities + number of degree job opportunities) (5)

2.1.4 Write a 'get' method called **getCityName** to return the name of the city. (2)

2.1.5 Write a method (function method) called **isMatchCity** that returns either 'true' or 'false'. This method must receive two parameters:

- A number indicating the minimum salary required
- Text indicating a job category (''Diploma'' or ''Degree'')

The method returns **true** if **BOTH** of the following conditions are met:

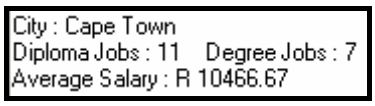
- The average salary of the available job opportunities in the city must be more than the minimum salary required which is indicated by the salary parameter.
- The city must have more job opportunities for the job category specified by the job category parameter than the other job category, for example if the job category parameter is Diploma, the city must have more diploma than degree job opportunities. (5)

2.1.6 You have been provided with a method (function method) called **toString** that constructs and returns a string with information which indicates the city name, the total number of diploma job opportunities, the total number of degree job opportunities and the average salary in that city. However, the format of the returned string is incorrect and the code has been inserted as a comment.

- Remove the comment symbols from the given code.
- Modify (Change) the given code so that it returns the information in the following format:

City : name
Diploma Jobs : diplomaJobs<tab>Degree Jobs : degreeJobs
Average Salary : R averageSalary

Example of the output:



(4)

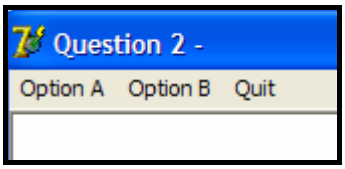
2.2 In the **Question2_UXXXX.pas** file (the main unit) you have been provided with code that includes the following:

- Two string arrays which contain the job titles that the agency deals with in each job category (diploma and degree)
- A method (function method) called **jobCategory**. This method accepts a job title as a parameter and returns a string (''Diploma'' for a diploma job and ''Degree'' for a degree job) indicating the category that the job belongs to.

EXAMPLE:

With the parameter "**Mechanic**" the method will return "**Diploma**" as a result.
With the parameter "**Doctor**" the method will return "**Degree**" as a result.

The following menu will be displayed when you execute the program:



Do the following:

- Add your examination number to the right of 'Question 2 –' in the caption of the form.
- Write code to do the following in the **Question2_UXXXX.pas** file (the main unit) in the given program:

2.2.1 Read information from the text file **Jobs.txt** according to the steps below. The code must be written in the **OnActivate** event handler of the form.

(a) Test if the text file exists. Display a suitable message if the file does not exist and terminate the program.

(b) Read the first line from the text file and store this as the city's name.

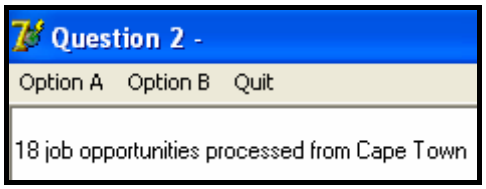
(c) Create a **single object** of type **TCity**. You do not need to create an array of these objects as the information of **only one city will be processed** each time the program is executed.

(d) Use a loop to do the following:

- Read two lines of text (the job and its salary) from the text file each time the loop executes.
- Use the function **jobCategory** to determine the category of the job ("Diploma" or "Degree").
- Based on the result of the previous step, call either one of the following methods in the **TCity** class: **addDipJob** or **addDegJob**.

(e) Use a counter to keep track of how many job opportunities were processed from the text file.

(f) Display the total number of job opportunities processed, as shown below.

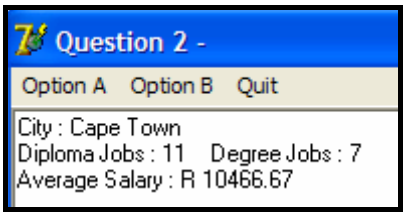


(20)

2.2.2 Complete the code for **Option A** as follows:

When the user selects this menu option, the program must display the information for the city by calling the **toString** method of the **TCity** class.

Example of the output:



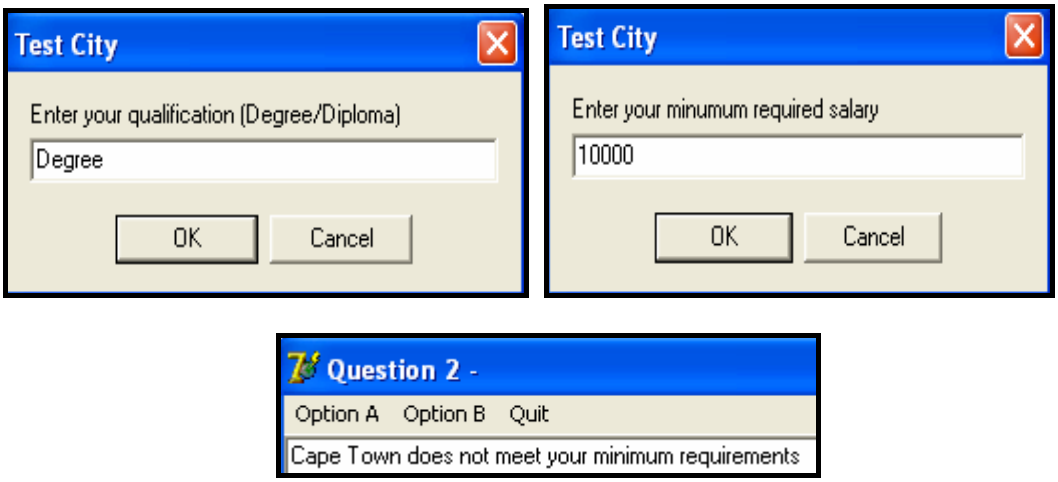
(2)

2.2.3 Complete the code for **Option B** as follows:

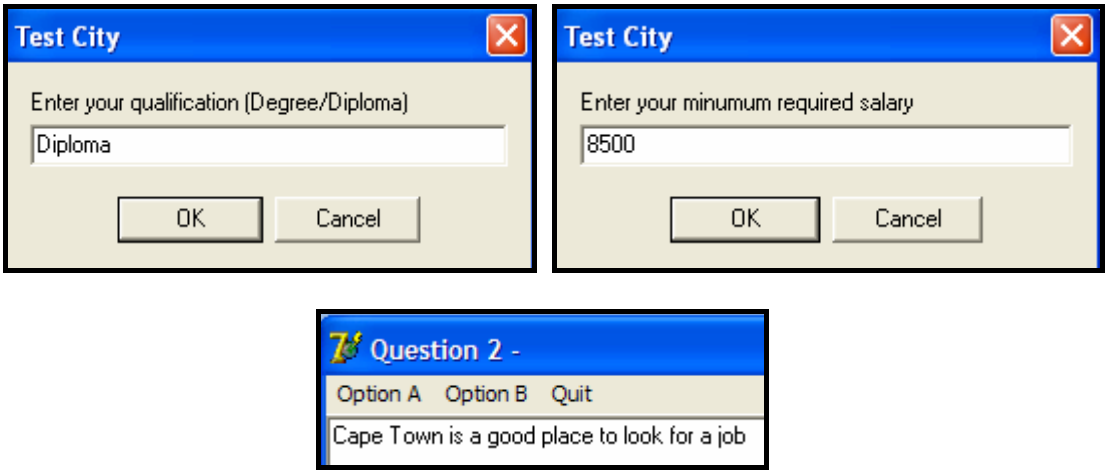
When the user selects this option, the program must

- ask the user to input what type of qualification they have (Diploma or Degree);
- ask the user to input what their minimum required salary is;
- use the **isMatchCity** method from the **TCity** class as well as the values for qualification and salary entered by the user to determine whether the city meets the candidate's job-hunting requirements (discussed in QUESTION 2.1.5). Display user-friendly output indicating the result.

Examples of the input and output (on the next page):



Another example of the input and output:



(5)

- Make sure your examination number is a comment in the first line of the main class **Question2_UXXXX.pas**, as well as the object class **uCityXXXX.pas**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **Question2_UXXXX.pas** and **uCityXXXX.pas** may be required (see Instruction 10 on page 3).

[49]

QUESTION 3: DELPHI – PROGRAMMING

Learners who want to enrol for a BSc IT degree at university were requested to e-mail their names and the name and mark of their best subject to the recruitment agency. The recruitment agency will process the information using prescribed criteria and supply a local university with the names of the candidates who met the criteria. The university wants to invite these candidates to spend a day on campus.

You have received an incomplete program in the folder named **Question 3 Delphi**. The program generates an array of 20 strings, each containing the name of a potential BSc IT student, his/her best subject and the mark for that subject in the following format:

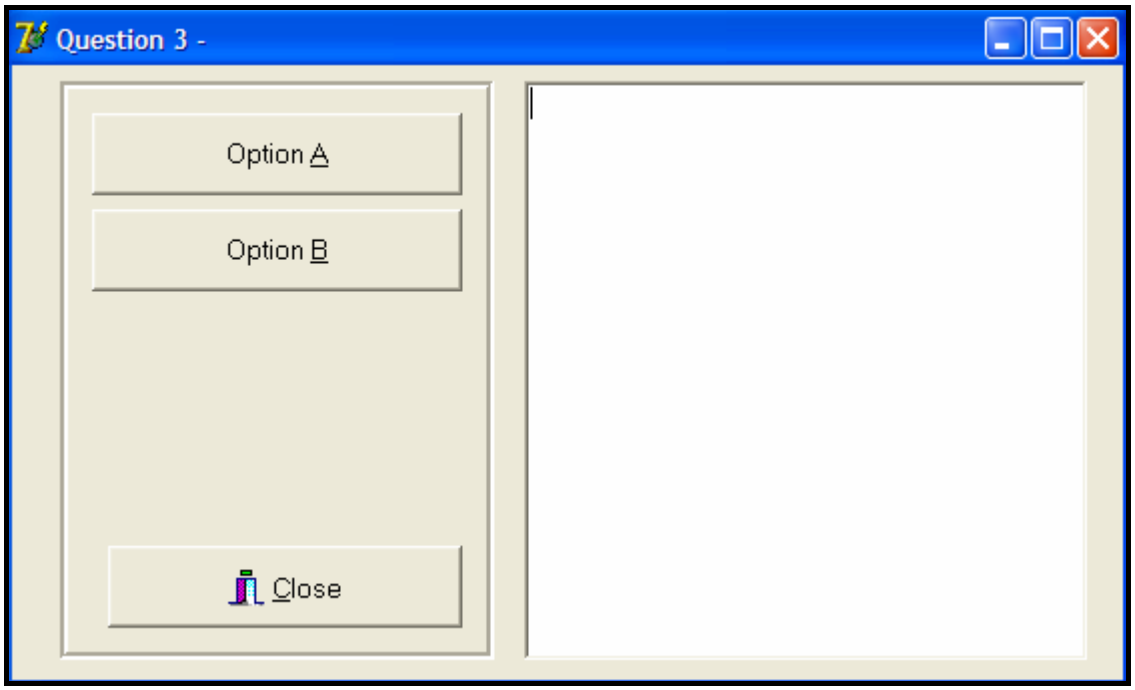
```
First name Last name, Best subject: Mark of best subject
```

Example of the first five entries in the array:

```
arrLearners[1] := 'Susan Thompson,Maths:77';
arrLearners[2] := 'Eric Ntumba,IT:89';
arrLearners[3] := 'Sean Franklin,Accounting:70';
arrLearners[4] := 'Mohammed Naidoo,Maths:68';
arrLearners[5] := 'Rowan Huntley,IT:77';
:
```

Do the following:

- Rename the folder named **Question 3 Delphi** as **Quest3_X** (where X should be replaced with your examination number).
- Open the Delphi program in this folder.
- Save the unit as ('File/Save As') **Visitors_UXXXX** and the project as ('File/Save Project As') **Visitors_PXXXX** inside the folder (XXXX should be replaced with the last FOUR digits of your examination number).
- Add your examination number to the right of 'Question 3 –' in the caption of the form.
- Execute the program. A menu with the following options will be displayed (on the next page):



Do the following:

3.1 Learners who indicated that Maths, Science or IT is their best subject and who have a mark of 70 or more for this subject, are the successful candidates and will be invited to visit the university. Write code in the **onCreate Eventhandler** to select names from the given array that meet the criteria for visiting the university and place these names in a visitors' array.

NOTE: You can assume that all the marks will contain two digits. (12)

3.2 Write a subprogram to alphabetically sort the array with the names of selected candidates. Remember that this array will contain an unknown number of elements. (6)

3.3 Write code for the **Option A** button that will display an alphabetically sorted list with the names of the selected candidates. Use the subprogram written in QUESTION 3.2 to sort the names alphabetically.

Example of the output:

```
Alphabetically Sorted List of Visitors
==============================
Bryan Smith
Camilla Johnson
Cindy Mokoena
Dwane Franklin
Eric Ntumba
Gregory Thompson
Joe Zimmerman
Mpho Anderson
Rowan Huntley
Susan Thompson
Taryn Peterson
Zane Shameez
```

(4)

3.4 Temporary student numbers must be generated for the learners who were selected to visit the university. Write code for the **Option B** button to do the following:

- Use the subprogram written in QUESTION 3.2 to sort the array of selected names.
- Generate a student number for each learner as follows:
  - Get the first three consonants from the name of the learner. The student number may contain only capital letters.
    NOTE: All the letters from the alphabet except the vowels (A, E, I, O, U) are consonants.
  - Generate a three-digit random number.
  - Join the three consonants and the randomly generated number in a string to form a student number consisting of six characters.
- Display the names of the selected learners and their temporary student numbers. Display a suitable heading and subheadings.

Example of the output:

NOTE: The student numbers will be different every time the program is executed due to the numbers being randomly generated.

```
List of Visitors with Student Numbers
=================================
Name                Student Number
=================================
Bryan Smith         BRY531
Camilla Johnson     CML150
Cindy Mokoena       CND709
Dwane Franklin      DWN392
Eric Ntumba         RCN787
Gregory Thompson    GRG579
Joe Zimmerman       JZM546
Mpho Anderson       MPH465
Rowan Huntley       RWN250
Susan Thompson      SSN939
Taryn Peterson      TRY404
Zane Shameez        ZNS545
```

(14)

- Enter your examination number as a comment in the first line of the unit **Visitors_UXXXX**, as well as any other unit(s) you may have created.
- Save the unit (or units) and the project ('File/Save All').
- A printout of the code for the unit as **Visitors_UXXXX**, as well as any other unit(s) you may have created, may be required (see Instruction 10 on page 3). **[36]**

TOTAL SECTION A: 120

SECTION B

Answer ALL the questions in this section only if you studied **Java**.

SCENARIO

People are always looking for better job opportunities. There are many recruitment agencies as well as training institutions involved in the employment industry. All of these agencies and institutions need appropriate software to be able to operate efficiently.

QUESTION 1: JAVA PROGRAMMING AND DATABASE

An independent career company keeps track of a list of recruitment agencies and the details of their clients. A database named **RecruitmentDB** has been developed to store their data. An incomplete program has been developed to process queries on the data in the given database. Your task will be to complete this program.

The database named **RecruitmentDB**, as well as an incomplete Java program, are saved in the folder named **Question 1 Java**. The folder contains a test class named **TestRecruitment.java** and an object class named **Recruitment.class** which will display the results of the queries.

NOTE: The design of the tables in the **RecruitmentDB** database and the sample data for this question can be found in **ANNEXURE A: Table Description Sheet**.

NOTE: If you cannot use the database provided follow the instructions in **ANNEXURE B** to create the database before you answer any of QUESTIONS 1.1 to 1.7.

NOTE: Make a copy of the **RecruitmentDB** database BEFORE you start with the solution. You will need a copy of the original database to be able to test your program thoroughly.

Do the following:

- Rename the folder **Question 1 Java** as **Quest1_X**, where X should be replaced with your examination number.
- Open the incomplete program **TestRecruitment.java** in the **Quest1_X** folder.
- Change the name of the class to **TestRecruitmentXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Save the class as **TestRecruitmentXXXX.java** (where XXXX must be replaced with the last FOUR digits of your examination number).

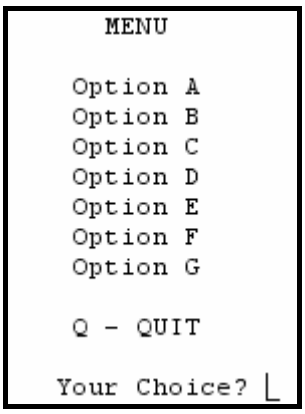
The connectivity code as well as the code to display the results have already been written as part of the given code in the file named **Recruitment.class**.

NOTE: If your program cannot connect to the database, make sure that the database file **RecruitmentDB** is in the same folder as. If this is not the case, copy the database file **RecruitmentDB** into the same folder as your program. Your program will not work if the database file is in a folder other than the folder containing your Java program.

NOTE: If you still cannot establish connectivity with the database when you execute the program, you must still do the SQL code and submit it for marking.

Marks will only be awarded for the programming code which contains the SQL statements in the file named TestRecruitmentXXXX.java.

When you compile and execute the **TestRecruitmentXXXX.java** file, the menu below will be displayed. However, if you enter any of the options (A to G), the program will not work because of the incomplete SQL statements.



Do the following:

Complete the SQL statements in **TestRecruitmentXXXX.java** for each menu option, as indicated in QUESTIONS 1.1 to 1.7 below. The code to pass the SQL statements to the relevant methods in the **Recruitment.class** file has been given to you. You need only complete the SQL statements and some input statements as required in the **TestRecruitmentXXXX.java** file.

1.1 A student needs to place her CV with a reputable agency. She needs a list of agencies that indicates the number of clients they have previously placed and other important information about the company. Complete the code for the **Option A** menu option by formulating an SQL statement to display **all the details** of agencies stored in the **tblAgencies** table. Display the results in descending order according to **NumPrevPlacements field**.

Example of the output for the first seven records (on the next page):

AgencyName	Province	International	NumPrevPlacements
Best Jobs Inc	KwaZulu-Natal	False	29058
Nellie Agencies	Gauteng	False	28167
Instant Emp Int	Western Cape	True	25627
KZN Recruitment Centre	KwaZulu-Natal	False	25145
Ronnies Agencies	Mpumalanga	False	24600
Boss Agencies	KwaZulu-Natal	True	24398
Only the Best	Mpumalanga	False	24234

: (4)

1.2 A corporate company would like to provide bursaries to all clients that are part-time and earn a salary of less than R15 000. Complete the code for the **Option B** menu option by formulating an SQL statement to display **Name**, **Surname** and **Salary** of all clients that are part-time and earn a salary of less than R15 000.

Example of the output:

Name	Surname	Salary
Brianna	Amena	14328
Magee	Noah	14260
Randall	Lawrence	12526
Brent	Arden	10706
Ciara	Martina	12349
Grace	Sydnee	11215
Claudia	Logan	14724

(4)

1.3 Clients complained that there are too few agencies that operate internationally. The career company needs to provide the clients with the number of agencies that operate internationally. Complete the code for the **Option C** menu option by formulating an SQL statement to display the number of recruitment agencies that operate internationally. Store the calculated field in **Count**.

Example of the output:

```
The number of agencies that offer international jobs are:11
```

(4)

1.4 An agency that places a client will receive 10% of the client's salary as commission. Complete the code for the **Option D** menu option by formulating an SQL statement that will display the **Name**, **Surname**, **Salary** and a **calculated field** for **commission**, rounded off to TWO decimal places. Name this calculated field **AgentComm**.

Example of the output for the first seven records:

Name	Surname	Salary	AgentComm
Lucas	Florence	17289	1728.90
Illana	Alyssa	27855	2785.50
Keefe	Lawrence	28940	2894.00
Adam	Isaiah	19142	1914.20
Brianna	Amena	14328	1432.80
Demetrius	Kaseem	24002	2400.20
Garrett	Halla	17481	1748.10

: (5)

1.5 A new agency has been established. This agency operates locally and does not offer clients international job opportunities. Complete the code for the **Option E** menu option by formulating an SQL statement that will **insert** the following record:

AgencyName: Jobs Unlimited
Province: Western Cape
NumPrevPlacements: 0

Example of the output:

```
Record inserted successfully
```

HINT: Run **Option A** to verify that the record has been inserted. (4)

1.6 The provinces of Western Cape and Gauteng would like to certify all the clients who have been placed at agencies in these provinces. Complete the code for the **Option F** button by formulating an SQL statement to display the **Name**, **Surname**, **AgencyName** and **Province** of clients who have been placed by agencies in these two provinces.

Example of the output:

Name	Surname	AgencyName	Province
Randall	Lawrence	Apollo International	Western Cape
Ciara	Martina	Apollo International	Western Cape
Adam	Isaiah	First Choice	Western Cape
Richard	Quyn	First Choice	Western Cape
Colin	Elmo	First Choice	Western Cape
Colby	Lionel	First Choice	Western Cape
Todd	Harper	First Choice	Western Cape
Judith	Ashely	First Choice	Western Cape
Blaze	Ashton	First Choice	Western Cape
Sawyer	Elaine	Instant Emp Int	Western Cape
Damon	Teagan	Instant Emp Int	Western Cape
Nichole	Clark	Instant Emp Int	Western Cape
Claudia	Logan	Instant Emp Int	Western Cape
Demetrius	Kaseem	Nellie Agencies	Gauteng
Jenna	Chastity	Nellie Agencies	Gauteng
Grace	Sydnee	Nellie Agencies	Gauteng
Genevieve	Nicholas	Nellie Agencies	Gauteng

(6)

1.7 The career company must be able to view lists of clients who have been placed by agencies before any specific cut-off date. Complete the code for the **Option G** menu option by asking the user to enter the name of the agency and the cut-off date. Formulate an SQL statement to display the **Name**, **Surname** and **DatePlaced** of all the clients who have been placed by the specified agency before the specified cut-off date.

Example of the input and output of all the clients placed by **Boss Agencies** before **30/04/2010**:

```
Enter the name of the agency
Boss Agencies
Enter the cut-off date
30/04/2010

Name                 Surname              DatePlaced
================================================
Lucas                Florence             2010-02-21
Chaney               Akeem                2010-02-06
Lara                 Norman               2010-04-01
```

NOTE: Any format of the date will be accepted. (8)

- Enter your examination number as a comment in the first line of the file named **TestRecruitmentXXXX.java** containing the SQL statements.
- Save the **TestRecruitmentXXXX.java** file.
- A printout for the code of the **TestRecruitmentXXXX.java** file will possibly be required (see Instruction 10 on page 3). **[35]**

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

A recruitment agency has identified a need where graduates from tertiary education institutions often want to know which city offers the best job opportunities for them. This agency has asked each of its branches in various cities to supply them with a list of job opportunities available in their cities and the salaries associated with these job opportunities.

The agency needs software which will allow graduates to analyse the job opportunities of a particular city to decide whether they will be available for employment in that city.

The agency identified two categories of job opportunities for graduates, namely **diploma job opportunities** and **degree job opportunities**. A city is a good choice for a graduate if

- the number of job opportunities for their category (diploma or degree) is higher than the number of job opportunities available for the other category; and
- the average monthly salary of all job opportunities in the city is higher than the monthly salary they require to live comfortably.

EXAMPLE:

John is a graduate and has a **degree** in engineering. He requires a salary of at least **R10 000** per month to live comfortably.
Durban has five diploma and ten degree job opportunities.
The average salary of the job opportunities in Durban is R12 500 per month.

This makes Durban a suitable city for John in which to look for employment, since

- the number of degree job opportunities is higher than the number of diploma job opportunities; AND
- the average salary of job opportunities in Durban is higher than the salary John requires.

The agency only deals with the following specific careers in each category:

Diploma job opportunities: secretary, mechanic, electrician, beautician, nurse, plumber
Degree job opportunities: doctor, programmer, architect, teacher, lawyer, engineer

The branch of the agency in each city has compiled a text file with specific job opportunities available in that city. The data stored in the text file named **Jobs.txt** contains the information on the **job opportunities available in a single city**. The format of the information in the text file is as follows (on the next page):

Name of the city
Job Title
Salary
Job Title
Salary
etc.

An example of the data in the text file is as follows:

Cape Town
Mechanic
6500
Architect
14500
Doctor
25000
etc.

You are required to complete the given program in the folder **Question 2 Java** (as indicated in QUESTION 2.1 and QUESTION 2.2 that follow below) to process **ONE** text file of a single city.

Do the following:

- Rename the folder **Question 2 Java** as **Quest2_X** (where X represents your examination number).
- Rename the **City.java** file in the folder **Quest2_X** as **CityXXXX.java** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the **CityXXXX.java** file.
- Change the **class name** and **constructor method** to **CityXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Add your examination number as a comment in the first line of the **CityXXXX.java** class. Save the file.
- Rename the **TestCity.java** file in the folder **Quest2_X** as **TestCityXXXX.java** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the **TestCityXXXX.java** file.
- Change the **class name** to **TestCityXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number). Save the file.

You have been given two files containing incomplete code in the **Question 2 Java** folder. Open the files and complete the code according to the following instructions:

2.1 The object class named **CityXXXX.java** contains fields and methods that describe the state of available job opportunities in a single city. Note the following:

- All the fields in this class should be private and all methods public.
- Parts of this class have been placed in comment symbols in order to get it to compile. Remove the comment symbols from the statements during the completion of the program.

- Modify code in the given methods and write some additional methods, as described below.

2.1.1 Create private fields **with the following names** to hold the data of a city. Choose appropriate data types for the fields:

- **cityName** – name of the city
- **diplomaJobs** – the total number of available diploma job opportunities in the city
- **degreeJobs** – the total number of available degree job opportunities in the city
- **salaryTotal** – the total of the salaries of all of the available degree and diploma job opportunities in the city added together

NOTE: It is important that you **use the field names given** in bold above in order for the given code to work correctly. (3)

2.1.2 You have been provided with a **default constructor** method. Write an **additional constructor method** which accepts one parameter for the city's name. Initialise the name field using the parameter value and initialise the other fields to zero. (3)

2.1.3 You have been provided with two void methods named **addDipJob** and **addDegJob** that each receives a salary as a parameter. When called, these methods increase the number of diploma and degree job opportunities by one respectively and add the value of the salary parameter to the total salary field.

For example, calling the **addDegJob** method with a parameter value of **2000** will increase the number of degree job opportunities in the city by one and add R2 000 to the salary total for that city.

Do the following:

- Remove the comment symbols from the code of each of these two given methods.
- Write a method called **averageSalary** which calculates and returns the average salary for the city, rounded off to TWO decimal places. The average salary is calculated as follows:

  total salary/(number of available diploma job opportunities + number of available degree job opportunities) (5)

2.1.4 Write a 'get' method called **getCityName** to return the name of the city. (2)

2.1.5 Write a method called **isMatchCity** that returns either 'true' or 'false'. This method must receive two parameters:

- A number indicating the minimum salary required
- Text indicating a job category (''Diploma'' or ''Degree'')

The method returns **true** if **BOTH** of the following conditions are met:

- The average salary of the available job opportunities in the city must be more than the minimum salary required which is indicated by the salary parameter.
- The city must have more job opportunities in the job category specified by the job category parameter than the other job category, for example if the job category parameter is 'Diploma', the city must have more diploma than degree job opportunities. (5)

2.1.6 You have been provided with a method called **toString** that constructs and returns a string with information which indicates the city name, the total number of diploma job opportunities, the total number of degree job opportunities and the average salary in that city. However, the format of the returned string is incorrect and therefore the code has been inserted as a comment.

- Remove the comment symbols from the given code.
- Modify (Change) the given code so that it returns the information in the following format:

City : name
Diploma Jobs : diplomaJobs<tab>Degree Jobs : degreeJobs
Average Salary : R averageSalary

Example of the output:

```
City : Cape Town
Diploma Jobs : 11       Degree Jobs : 7
Average Salary : R 10466.67
```

(4)

2.2 In the **TestCityXXXX.java** file (the main class) you have been provided with code that includes the following:

- Two string arrays which contain the job titles that the agency deals with in each job category (diploma and degree)
- A static method called **jobCategory**. This method accepts a job title as a parameter and returns a string (''Diploma'' for a diploma job and ''Degree'' for a degree job) indicating the category that the job belongs to.

EXAMPLE:

With the parameter "**Mechanic**" the method will return "**Diploma**" as a result.
With the parameter "**Doctor**" the method will return "**Degree**" as a result.

The following menu will be displayed when you execute the program:

```
     MENU

A - Option A
B - Option B

Q - QUIT

Your Choice?
```

Do the following:

- Enter your examination number as a comment in the first line of the program **TestCityXXXX.java**.
- Write code to do the following in the **TestCityXXXX.java** file in the given program:

2.2.1 Read information from the text file **Jobs.txt** according to the following steps:

(a) Test if the text file exists. Display a suitable message if the file does not exist and terminate the program.

(b) Read the first line from the text file and store this as the city's name.

(c) Create a **single object** of type **City**. You do not need to create an array of these objects as the information of **only one city will be processed** each time the program is executed.

(d) Use a loop to do the following:
- Read two lines of text (the job and its salary) from the text file each time the loop executes.
- Use the method **jobCategory** to determine the category of the job ("Diploma" or "Degree").
- Based on the result of the previous step, call either one of the following methods from the **City** class: **addDipJob** or **addDegJob**.

(e) Use a counter to keep track of how many job opportunities were processed from the text file.

(f) Display the total number of job opportunities processed, as shown below.

```
18 job opportunities processed from Cape Town
```

(20)

2.2.2 Complete the code for **Option A** as follows:

When the user selects this option, the program must display the information for the city by invoking the **toString** method of the **City** class.

Example of the output:

```
City : Cape Town
Diploma Jobs : 11       Degree Jobs : 7
Average Salary : R 10466.67
```

(2)

2.2.3 Complete the code for **Option B** as follows:

When the user selects this option, the program must

- ask the user to input what type of qualification he/she has (Diploma or Degree);
- ask the user to input what his/her minimum required salary is;
- use the **isMatchCity** method from the **City** class as well as the values for qualification and salary entered by the user to determine whether the city meets the candidate's job hunting requirements (discussed in QUESTION 2.1.5). Display user-friendly output indicating the result.

Example of the input and output:

```
Enter your qualification (Degree/Diploma) : Degree
Enter your minimum required salary : 10000
Cape Town does not meet your minimum requirements
```

Another example of the input and output:

```
Enter your qualification (Degree/Diploma) : Diploma
Enter your minimum required salary : 8500
Cape Town is a good place to look for a job
```

(5)

- Make sure your examination number is entered as a comment in the first line of the class **TestCityXXXX.java**, as well as the object class **CityXXXX.java**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **TestCityXXXX.java** and **CityXXXX.java** may be required (see Instruction 10 on page 3).

[49]

QUESTION 3: JAVA – PROGRAMMING

Learners who want to enrol for a BSc IT degree at university were requested to e-mail their names and the name and mark of their best subject to the recruitment agency. The recruitment agency will process the information using prescribed criteria and supply a local university with the names of the candidates who met the criteria. The university wants to invite these candidates to spend a day on campus.

You have received an incomplete program in the folder named **Question 3 Java**. The program generates an array of 20 strings, each containing the name of a potential BSc IT student, his/her best subject and the mark for that subject in the following format:

```
First name Last name, Best subject: Mark of best subject
```

Example of the first five entries in the array:

```
arrLearners[0] = "Susan Thompson,Maths:77";
arrLearners[1] = "Eric Ntumba,IT:89";
arrLearners[2] = "Sean Franklin,Accounting:70";
arrLearners[3] = "Mohammed Naidoo,Maths:68";
arrLearners[4] = "Rowan Huntley,IT:77";
  :
```

Do the following:

- Rename the folder named **Question 3 Java** as **Quest3_X** (where X should be replaced with your examination number).
- Rename the file **TestVisitors.java** in this folder as **TestVisitorsXXXX.java** (XXXX should be replaced with the last FOUR digits of your examination number).
- Open the file (incomplete program) **TestVisitorsXXXX.java**. Change the name of the class to **TestVisitorsXXXX**.
- Add your examination number as a comment in the first line of the program.
- Execute the program. A menu with the following options will be displayed:

```
      MENU

A - Option A
B - Option B

Q - QUIT

Your Choice?
```

Do the following (on the next page):

3.1 Learners who indicated that Maths, Science or IT is their best subject and who have a mark of 70 or more for this subject, are the successful candidates and will be invited to visit the university. Write code to be executed before the menu is displayed to select names from the given array that meet the criteria for visiting the university and place these names in a visitors' array.

NOTE: You can assume that all the marks will contain two digits. (12)

3.2 Write a method to alphabetically sort the array with the names of selected candidates. Remember that this array will contain an unknown number of elements. (6)

3.3 Write code for the **Option A** button that will display an alphabetically sorted list with the names of the selected candidates. Use the method written in QUESTION 3.2 to sort the names alphabetically.

Example of the output:

```
Alphabetically Sorted List of Visitors
======================================
Bryan Smith
Camilla Johnson
Cindy Mokoena
Dwane Franklin
Eric Ntumba
Gregory Thompson
Joe Zimmerman
Mpho Anderson
Rowan Huntley
Susan Thompson
Taryn Peterson
Zane Shameez
```

(4)

3.4 Temporary student numbers must be generated for the learners who were selected to visit the university. Write code for the **Option B** button to do the following:

- Use the method written in QUESTION 3.2 to sort the array of selected learners.
- Generate a student number for each learner as follows:
  - Get the first three consonants from the name of the learner. The student number may contain only capital letters.
    NOTE: All the letters from the alphabet except the vowels (A, E, I, O, U) are consonants.
  - Generate a three-digit random number.
  - Join the three consonants and the randomly generated number in a string to form a student number consisting of six characters.
- Display the names of the selected learners and their temporary student numbers. Display a suitable heading and subheadings.

Example of the output (on the next page):

NOTE: The student numbers will be different every time the program is executed due to the numbers being randomly generated.

```
List of Visitors with Student Numbers
=====================================
Name                    Student Number
=====================================
Bryan Smith             BRY641
Camilla Johnson         CML370
Cindy Mokoena           CND260
Dwane Franklin          DWN705
Eric Ntumba             RCN634
Gregory Thompson        GRG279
Joe Zimmerman           JZM750
Mpho Anderson           MPH932
Rowan Huntley           RWN297
Susan Thompson          SSN773
Taryn Peterson          TRY506
Zane Shameez            ZNS237
```

(14)

- Enter your examination number as a comment in the first line of the class **TestVisitorsXXXX.java**, as well as any other class(es) you have created with code.
- Save the class(es).
- A printout of the code for the class **TestVisitorsXXXX.java**, as well as any other class(es) you have created, may be required (see Instruction 10 on page 3).

[36]

TOTAL SECTION B: 120
GRAND TOTAL: 120

ANNEXURE A: Table Description Sheet

This sheet shows the data structure and sample data for the tables used in the **RecruitmentDB** database in **Question 1.**

tblAgencies Table Structure

tblAgencies : Table

Field Name	Data Type	Description
AgencyName	Text	Unique name of the agency
Province	Text	Province where the agency is located
International	Yes/No	Yes means international job opportunities are offered
NumPrevPlacements	Number	The number of clients who were placed by the agency previously

tblAgencies Table Sample Data

tblAgencies : Table

AgencyName	Province	International	NumPrevPlacements
Anderson Agencies	Mpumalanga	☑	11552
Apollo International	Western Cape	☑	18204
Best Jobs Inc	KwaZulu-Natal	☐	29058
Boss Agencies	KwaZulu-Natal	☑	24398
Cool Jobs Int	Mpumalanga	☑	20443
First Choice	Western Cape	☑	10409
Instant Emp Int	Western Cape	☑	25627
Kaiyal Recruits	Mpumalanga	☐	13097
King Employment	Gauteng	☑	20887
KZN Recruitment Centre	KwaZulu-Natal	☐	25145
Mchunu International	Mpumalanga	☑	20994
Mpu Job Centre	Mpumalanga	☑	16924
Ndwedwe Job Find	Mpumalanga	☑	21579
Nellie Agencies	Gauteng	☐	28167
Nkosi Recruitment Co	Limpopo	☑	22586
Only the Best	Mpumalanga	☐	24234
Quick Find	KwaZulu-Natal	☐	22850
Recruitment for Leaders	KwaZulu-Natal	☐	18523
Ronnies Agencies	Mpumalanga	☐	24600
Tash for Cash Agencies	Mpumalanga	☐	10778

tblClients Table Structure

tblClients : Table

Field Name	Data Type	Description
ClientNo	AutoNumber	Unique number assigned to the client
Name	Text	First name of the client
Surname	Text	Surname of the client
DatePlaced	Date/Time	Date the client was placed
PlacedBy	Text	Name of the agency that placed the client
Salary	Currency	Salary the client earns
FullTime	Yes/No	Yes if the client is full time employed

tblClients Table Sample Data

tblClients : Table

ClientNo	Name	Surname	DatePlaced	PlacedBy	Salary	FullTime
1	Lucas	Florence	2/21/2010	Boss Agencies	R17,289.00	☐
2	Illana	Alyssa	3/27/2010	Cool Jobs Int	R27,855.00	☐
3	Keefe	Lawrence	4/8/2010	Mchunu International	R28,940.00	☑
4	Adam	Isaiah	2/14/2010	First Choice	R19,142.00	☐
5	Brianna	Amena	4/25/2010	Kaiyal Recruits	R14,328.00	☐
6	Demetrius	Kaseem	3/30/2010	Nellie Agencies	R24,002.00	☑
7	Garrett	Halla	2/8/2010	Nkosi Recruitment Co	R17,481.00	☐
8	Chadwick	Steven	3/12/2010	Mchunu International	R26,626.00	☑
9	Kerry	Cullen	2/18/2010	Quick Find	R19,805.00	☐
10	Magee	Noah	1/23/2010	Best Jobs Inc	R14,260.00	☐
11	Galena	Brody	3/29/2010	Anderson Agencies	R24,457.00	☑
12	Serina	Keefe	2/14/2010	Nkosi Recruitment Co	R19,657.00	☐
13	Richard	Quyn	4/2/2010	First Choice	R28,069.00	☐
14	Sade	Caesar	2/21/2010	Kaiyal Recruits	R27,219.00	☐
15	Malcolm	Ferdinand	3/19/2010	KZN Recruitment Centre	R22,844.00	☐
16	Karyn	Cameran	3/11/2010	Tash for Cash Agencies	R11,814.00	☑
17	Tad	Amelia	2/5/2010	Cool Jobs Int	R10,087.00	☑
18	Amal	Callum	4/4/2010	KZN Recruitment Centre	R19,480.00	☐
19	Chaney	Akeem	2/6/2010	Boss Agencies	R28,674.00	☑
20	Karyn	Maxine	1/29/2010	Cool Jobs Int	R14,946.00	☑

ANNEXURE B: Instructions to create the database RecruitmentDB.mdb

If you cannot use the database provided, do the following:

- Use the two text files named **tblAgencies** and **tblClients** that have been supplied. Create your own database with the name **RecruitmentDB** with a table named **tblAgencies** and another table named **tblClients** in the folder called **Question 1 Delphi** or **Question 1 Java**.
- Change the data types and the sizes of the fields in the two tables according to the specifications given below.

The **tblAgencies** table stores data on the agencies that recruit clients. The fields in the **tblAgencies** table are defined as follows:

Field Name	Type	Size	Description
AgencyName	Text	30	A unique name entered for each agency
Province	Text	30	Province where the agency is located
International	Yes/No		Yes means the agency offers international jobs
NumPrevPlacements	Number	Longint	The number of clients who were placed by the company previously

See ANNEXURE A for an example of the data in the **tblAgencies** table.

The **tblClients** table stores data on the clients of each agency. The fields in the **tblClients** table are defined as follows:

Field Name	Type	Size	Description
ClientNo	AutoNumber	Longint	A unique number assigned to each client
Name	Text	30	First name of the client
Surname	Text	30	Surname of the client
DatePlaced	Date/Time	ShortDate	Date the client was replaced
PlacedBy	Text	30	Name of the agency that placed the client
Salary	Currency		Salary the client earns
FullTime	Yes/No		Yes if the client is employed full-time

See ANNEXURE A for an example of the data in the **tblClients** table.

ANNEXURE C: Instructions to connect to the database in Delphi

If you cannot use the database provided, do the following:

- Click on the ADOQuery component named **qryRec**.
- Click on the Ellipse button (three dots) to the right of the 'ConnectionString' property in the Object Inspector.
- Click on the Build button which takes you to the Data Link Properties dialogue box.
- Click on the Provider tab to open the Provider tab sheet and select Microsoft Jet 4.0 OLE DB Provider. Click on the Next button.
- The Connection tab sheet will be displayed. The first option on the Connection tab sheet provides an Ellipse button (three dots) that allows you to browse and look for the **RecruitmentDB** file. You will find this file in the **Question 1 Delphi** folder. Once you have found it, select the **RecruitmentDB** file and click on the Open button.
- Remove the user name Admin.
- Click on the Test Connection button.
- Click OK on each one of the open dialogue windows.

INFORMATION TECHNOLOGY PAPER 1
FEBRUARY/MARCH 2011

INFORMATION SHEET *(to be completed by candidates)*

120	

NAME OF PROVINCE _____

CENTRE NUMBER _____

EXAMINATION NUMBER _____

WORKSTATION NUMBER _____

DATE OF EXAMINATION _____

Programming language used
(Mark appropriate box with a cross (X).)

Delphi	Java

FOLDER NAME _____

Enter the file name used for each answer and tick if saved.

Question number	**File name**	**Saved (*tick* ✓)**	**Maximum mark**	**Mark achieved**	**Marker initial/ code**
1			35		
2			49		
3			36		
TOTAL			**120**		

Comment *(for official use only)*

_____

_____

_____

_____

_____

_____