GCE AS

B500U20-1

S19-B500U20-1

COMPUTER SCIENCE – AS component 2
Practical Programming to Solve Problems

FRIDAY, 24 MAY 2019 – MORNING

2 hours 15 minutes

B500U201
01

INSTRUCTIONS TO CANDIDATES

Answer **ALL** of questions 1, 2, 3 and 4.

Answer only **ONE** section of question 5. This is the section which requires you to use the Integrated Development Environment (IDE) of your chosen programming language.

You will need to record all of your answers to questions 1, 2, 3 and 4 in a **single** word processed document.

INFORMATION FOR CANDIDATES

The number of marks is given in brackets at the end of each question or part-question.

You are reminded of the need for good English and orderly, clear presentation in your answers.

The total number of marks available is 60.

You will need a computer with an installed functional copy of the Integrated Development Environment (IDE) appropriate to your chosen programming language and word processing software.

A calculator is allowed in this examination.

Remember to save your work regularly.

BLANK PAGE

Scenario

IceZone
Ice Skating Rink

IceZone is a large independent ice skating rink. The local ice hockey team play their home matches and train on the ice rink. **IceZone** also offers ice skating sessions to the general public during its daytime opening periods. A member of staff at the ice rink provides advanced lessons for competition skaters.

The manager of **IceZone** has decided to commission a new computerised system to store staff details such as staff ID, first name, surname and postcode. The system is also to be used to help monitor and record the temperature of the skating surface of the ice rink.

You have been commissioned to develop a prototype computer system.

1. **IceZone** is considering an object-oriented approach to programming its computer systems. It wishes to use a UML class diagram to describe the relationships between its classes.

IceZone would like a superclass called Person.

The Person class should have four protected attributes: firstName, surname, homeAddress, postcode, which are all of type string. The Person class should have four public methods to set each of the four attributes, which all accept a parameter.

IceZone would like a subclass called Staff which inherits from class Person.

The Staff class should have a private attribute called staffID of type Integer. The Staff class should have two public methods, one for setting the staff ID which accepts a parameter of type Integer and a method for returning the staffID which returns an Integer value.

Create a class diagram for this situation. [12]

2. **IceZone** will need to consider different user interfaces that can be provided by the operating system for its staff to use. **IceZone** is concerned that its staff are non-specialist users.

Describe two different user interfaces **IceZone** could provide for its staff. Explain why the different user interfaces are suitable for **IceZone's** staff by providing an advantage and a disadvantage of each. [8]

3. **IceZone** wishes to avoid bookings being accidently entered twice by searching for identical booking IDs in sequential time slots.

They have a pre-populated array called `bookings[5]` with the following data:

`bookings[5] = [2,3,3,7,8]`

```
1    Begin Subroutine searchForDB()
2    set i = 1
3    set Position = -1
4    set Found = FALSE
5
6    repeat
7     if bookings[i] = bookings[i - 1] then
8      set Position = i + 1
9      output “Position = ”, Position
10
11     set Found = TRUE
12     output “Found = ”, Found
13
14    else
15     set Position = i + 1
16     output “Position = ”, Position
17
18     output “Found = ”, Found
19
20    end if
21
22    set i = i + 1
23
24   until i = 5 {next loop}
25
26   End Subroutine
```

Copy and complete the table to show the outputs of the algorithm for the inputs provided. [8]

i:	Position:	Found:
1		
2		
3		
4		

4. **IceZone** has to maintain strict temperature controls over the skating surface.

Using a recognised convention, design an algorithm to help **IceZone** control the temperature of the ice.

The algorithm should repeatedly allow a user to input the temperature in degrees Celsius (°C) to the nearest 0.1 °C. The algorithm should continuously loop until a rogue value of 100 is entered. The algorithm should compare the value entered to the following set values and output the correct message:

- A positive number or above -2.1 should provide the message “Temperature too high”
- -2.1 to -4.0 should provide the message “Suitable for general skating”
- -4.1 to -5.5 should provide the message “Suitable for competition ice skating”
- -5.6 to -9.0 should provide the message “Suitable for ice hockey”
- below -9.0 should provide the message “Fault detected”.

Your algorithm should be written using self-documenting identifiers. [8]

5. Select the programming language of your choice from section *(a), (b) or (c)* and answer **all** questions in your chosen section.

(a) **Visual Basic**

IceZone wants a computer system to be developed using **Visual Basic** that meets the following requirements:

- The ability to store staff details
- The ability to recall and count specific staff details
- The ability to check the ice temperature is in a suitable range
- The ability to store and recall the last temperature input by the user.

(i) Open the file *IceStaff*
- Read through the code and familiarise yourself with its contents
- The file contains incomplete code, which is intended to allow **IceZone** to store and count staff with specific details.

Complete this code. [4]

Remember to save the changes made to the file *IceStaff*

(ii) Create a new form or program that will allow **IceZone** to implement the algorithm you designed as part of Question 4:

- Input a temperature.
- If the value is 100 the message displayed should be “program terminated”
- Output the appropriate message for the temperature input as shown in the table:

Temperature Range	Message
Above -2.1	Temperature too high
Between -2.1 and -4.0	Suitable for general skating
Between -4.1 and -5.5	Suitable for competition ice skating
Between -5.6 and -9.0	Suitable for ice hockey
Below -9.0	Fault detected

- Store the last temperature on disk in a text file called *IceResult.txt*
- Confirm storage of the last temperature to disk on screen
- Retrieve the last temperature from disk. [12]

Save your work as *IceZone*

(iii) Using the internal facility of the IDE, add annotation to your code from question **5***(a)*(ii) that would clearly explain the design of your program to another software developer. [8]

Save your annotations in the same file as 5*(a)*(ii) above.

(b) **Java**

IceZone wants a computer system to be developed using **Java** that meets the following requirements:

- The ability to store staff details
- The ability to recall and count specific staff details
- The ability to check the ice temperature ranges is in a suitable range
- The ability to store and recall the last temperature input by the user.

(i) Open the file *IceStaff*

- Read through the code and familiarise yourself with its contents
- The file contains incomplete code, which is intended to allow **IceZone** to store and count staff with specific details.

Complete this code. [4]

Remember to save the changes made to the file *IceStaff*

(ii) Create a new form or program that will allow **IceZone** to implement the algorithm you designed as part of Question 4:

- Input a temperature.
- If the value is 100 the message displayed should be "program terminated"
- Output the appropriate message for the temperature input as shown in the table:

Temperature Range	Message
Above -2.1	Temperature too high
Between -2.1 and -4.0	Suitable for general skating
Between -4.1 and -5.5	Suitable for competition ice skating
Between -5.6 and -9.0	Suitable for ice hockey
Below -9.0	Fault detected

- Store the last temperature on disk in a text file called *IceResult.txt*
- Confirm storage of the last temperature to disk on screen
- Retrieve the last temperature from disk. [12]

Save your work as *IceZone*

(iii) Using the internal facility of the IDE, add annotation to your code from question **5***(b)*(ii) that would clearly explain the design of your program to another software developer. [8]

Save your annotations in the same file as 5*(b)*(ii) above.

(c) **Python**

IceZone wants a computer system to be developed using **Python** that meets the following requirements:

- The ability to store staff details
- The ability to recall and count specific staff details
- The ability to check the ice temperature ranges is in a suitable range
- The ability to store and recall the last temperature input by the user.

(i) Open the file *IceStaff*
- Read through the code and familiarise yourself with its contents
- The file contains incomplete code, which is intended to allow **IceZone** to store and count staff with specific details.

Complete this code. [4]

Remember to save the changes made to the file *IceStaff*

(ii) Create a new form or program that will allow **IceZone** to implement the algorithm you designed as part of Question 4:

- Input a temperature.
- If the value is 100 the message displayed should be "program terminated"
- Output the appropriate message for the temperature input as shown in the table:

Temperature Range	Message
Above -2.1	Temperature too high
Between -2.1 and -4.0	Suitable for general skating
Between -4.1 and -5.5	Suitable for competition ice skating
Between -5.6 and -9.0	Suitable for ice hockey
Below -9.0	Fault detected

- Store the last temperature on disk in a text file called *IceResult.txt*
- Confirm storage of the last temperature to disk on screen
- Retrieve the last temperature from disk. [12]

Save your work as *IceZone*

(iii) Using the internal facility of the IDE, add annotation to your code from question **5***(c)*(ii) that would clearly explain the design of your program to another software developer. [8]

Save your annotations in the same file as 5*(c)*(ii) above.

END OF PAPER

BLANK PAGE

BLANK PAGE