# 2019 HSC Software Design and Development Marking Guidelines

## Section I

**Multiple-choice Answer Key**

| Question | Answer |
|---|---|
| 1 | D |
| 2 | B |
| 3 | C |
| 4 | C |
| 5 | A |
| 6 | A |
| 7 | C |
| 8 | A |
| 9 | C |
| 10 | B |
| 11 | D |
| 12 | D |
| 13 | A |
| 14 | A |
| 15 | C |
| 16 | B |
| 17 | D |
| 18 | B |
| 19 | B |
| 20 | C |

## Section II

### Question 21

| Criteria | Marks |
|---|---|
| • Shows a thorough understanding of relevant interface considerations | 3 |
| • Shows some understanding of interface considerations | 2 |
| • Shows a basic understanding of interfaces | 1 |

***Sample answer:***

A developer will need to consider how screen elements can be designed to fit small screens and still remain accessible and readable, what input methods are available on these devices (pinch, tap, shake) and what output methods are available (vibrate, audio feedback, pop-up messages).

### Question 22

| Criteria | Marks |
|---|---|
| • Identifies TWO advantages and TWO disadvantages | 3 |
| • Demonstrates some understanding of the issues | 2 |
| • Identifies a relevant issue | 1 |

***Sample answer:***

A large customer base (with a dominant developer) results in potentially more support being available through user groups. Because more clients use the product, most errors have been addressed.

However, less competition leads to less incentive for innovative creative ideas or maintenance of the product. Also, there is less incentive to respond to user requests for support.

## Question 23 (a)

| Criteria | Marks |
|---|---|
| • Outlines a similarity AND a difference | 2 |
| • Outlines a similarity OR a difference | 1 |

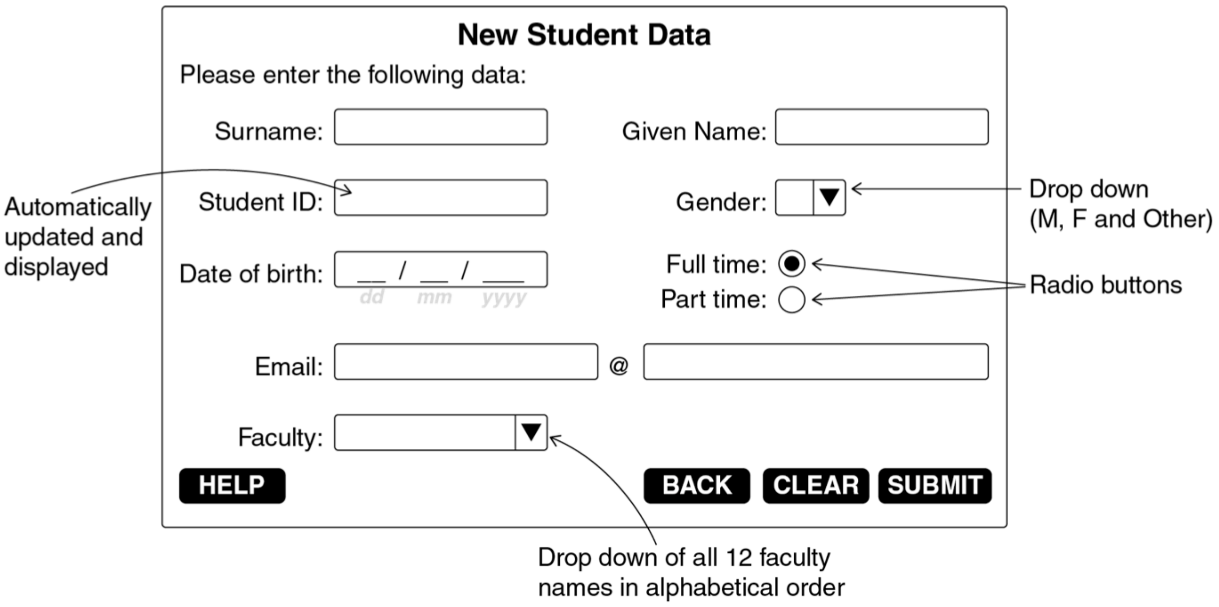
***Sample answer:***

Both arrays and records enable the grouping and storage of related data. However, all elements in an array must be of the same data type (eg integers or strings) while different fields in a record can be of different data types.

## Question 23 (b) (i)

| Criteria | Marks |
|---|---|
| • Provides an annotated interface that includes all relevant screen elements with appropriate validation | 4 |
| • Provides an annotated interface that provides most required elements, including validation | 3 |
| • Provides an interface that includes several of the required elements | 2 |
| • Shows some understanding of an interface for this scenario | 1 |

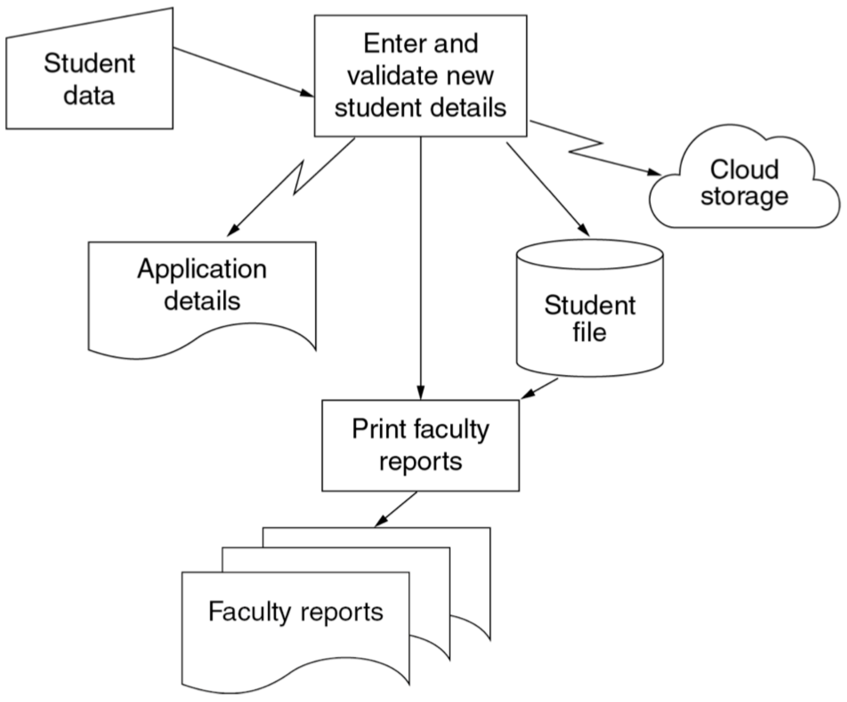
***Sample answer:***


New Student Data
Please enter the following data:
Surname:
Given Name:
Student ID:
Gender:
Date of birth: __ / __ / ___
dd mm yyyy
Full time:
Part time:
Email:
@
Faculty:
HELP
BACK
CLEAR
SUBMIT
Automatically updated and displayed
Drop down (M, F and Other)
Radio buttons
Drop down of all 12 faculty names in alphabetical order

## Question 23 (b) (ii)

| Criteria | Marks |
|---|---|
| • Provides a substantially correct system flowchart using appropriate symbols and labels | 3 |
| • Provides a system flowchart that shows some understanding of the process | 2 |
| • Shows some understanding of system flowcharts | 1 |

***Sample answer:***



## Question 23 (b) (iii)

| Criteria | Marks |
|---|---|
| • Provides a relevant explanation of when each search method is appropriate for the student records | 3 |
| • Explains when a linear OR a binary search is appropriate<br>OR<br>• Shows some understanding of both linear and binary searches | 2 |
| • Shows some understanding of either search method | 1 |

***Sample answer:***

A binary search requires the data to be sorted while a linear search does not. It can take time to re-sort 20 000 records each time a new search with a different target field is required. Thus it may be faster to perform multiple linear searches.

If there are several searches required, all based on the same target field, it is faster to sort the records on that target field once and then perform multiple binary searches.

A complex search based on multiple criteria (eg gender = “F” and faculty = “maths”) needs to be done using a linear search.

## Question 24 (a)

| Criteria | Marks |
|---|---|
| • Provides a substantially correct desk check | 2 |
| • Shows some understanding of desk checking | 1 |

***Sample answer:***

| *X* | *Y* | *Z* | *Output* |
|---|---|---|---|
| 1 | 2 | 2 | |
| 2 | | | 4 |
| 3 | | 2 | |
| 4 | | | 8 |
| 5 | | 2 | |
| 6 | | | 12 |
| 7 | | 2 | |
| 8 | | 2 | |
| 9 | | | Done |

## Question 24 (b)

| Criteria | Marks |
|---|---|
| • Provides various required changes, with explanation | 3 |
| • Provides some required changes OR explains one change | 2 |
| • Identifies a required change | 1 |

***Sample answer:***

No need to keep setting Z to Y (remove line 4 and change line 2 to Get Z).

We could add 2 to X in one statement, rather than two separate addition operations. Therefore line 6 becomes X = X + 2 (remove line 11 and change line 1 to X = 0).

There is no need to recheck if X < 8 as it is already known to be less than 7 (remove lines 7 and 9).

## Question 25 (a)

| Criteria | Marks |
|---|---|
| • Provides an approach for this project, with thorough justification | 3 |
| • Provides an approach for this project, with some relevant justification | 2 |
| • Identifies a development approach | 1 |

***Sample answer:***

As the system is constantly evolving, and the future needs may not yet be fully known, an agile approach will allow fast delivery of an initial version, followed by easy implementation of upgraded versions as users' needs become more apparent.

## Question 25 (b)

| Criteria | Marks |
|---|---|
| • Describes various ways relevant to this application | 3 |
| • Outlines ways relevant to this application | 2 |
| • Outlines one way of evaluating | 1 |

***Sample answer:***

The team can confirm (using a checklist) that the functionality and quality of the current live application meet all of the requirements.

The team can check the performance of the current live system by measuring typical response times, and times to load videos and images.

A quick online survey can be regularly delivered to gain feedback from a range of users, particularly after each upgrade, to identify areas of concern/satisfaction.

## Question 25 (c)

| Criteria | Marks |
|---|---|
| • Justifies a variety of ways relevant to the evolving social-networking application | 4 |
| • Justifies ways relevant to the evolving social-networking application<br>OR<br>• Justifies a variety of ways | 3 |
| • Justifies ways<br>OR<br>• Outlines a variety of ways | 2 |
| • Shows some understanding of software maintenance | 1 |

***Sample answer:***

Each team member should write to defined standards so that the code is uniform (consistent naming and use of language) and therefore easier to maintain.

There are a variety of logical tasks to code (eg logon, load, message, share, interface design), so each should be in one identifiable module. This makes it easier to modify individual modules as new needs are identified.

Similar applications already exist so the code should be written to allow ease of incorporation of required existing modules.

Version control should be set up from the start to handle the incorporation of any changes.

To adapt easily to new devices there should be a separate module dealing with interface design for maximum flexibility.

## Question 26

| Criteria | Marks |
|---|---|
| • Provides a substantially correct description that includes reference to loops, decisions and parameters | 4 |
| • Provides a relevant description that refers to the meaning of some of the symbols used | 3 |
| • Demonstrates some understanding of the structure chart provided | 2 |
| • Demonstrates a basic understanding of structure charts | 1 |

***Sample answer:***

The system calls a Login module which calls the EnterMemberDetails&Validate module, which repeatedly executes until the details are valid. It returns MemberID to the Login module and then back to the HotelBookingSystem mainline.

A choice is then continually offered of BrowseHotels or Book until the user chooses to exit. If they choose to book, their MemberID is passed into the Book module where booking details are entered. If a booking is successful, the flag BookingMade is set to true and returned with a generated BookingID to the HotelBookingSystem mainline.

Once the user has exited the Browse/Book modules, details of the newly made booking are passed into the DisplayNewBookings module which will display all bookings just made. This only occurs if the BookingMade flag is true.

## Question 27 (a)

| Criteria | Marks |
|---|---|
| • Justifies THREE relevant considerations for this system | 3 |
| • Justifies TWO relevant considerations | 2 |
| • Shows a basic understanding of the system's requirements | 1 |

***Sample answer:***

The developer needs to ensure that the interface design caters for the disabled customers in order to prevent unintended discrimination. Relevant functionality and data are important in order to offer tours relevant to disabled customers to cater for their needs. The proposed system must be compatible with their existing hardware and software to minimise costs.

***Answers could include:***

- Adequate security to protect sensitive customer data because the system is going online
- Providing alternative date and currency formats because the customer base may be global.

## Question 27 (b)

| Criteria | Marks |
|---|---|
| • Outlines THREE relevant factors | 3 |
| • Outlines TWO relevant factors | 2 |
| • Shows a basic understanding of programming language features | 1 |

***Sample answer:***

The programming language should be compatible with the existing/proposed architecture and operating system, and able to interface with the unusual hardware devices. It should fit in with the company's existing programming environment.

***Answers could include:***

- Allow easy access to support when required
- Be one that the developers are experienced in using.

## Question 27 (c)

| Criteria | Marks |
|---|---|
| • Provides an explanation of ONE advantage AND ONE disadvantage relevant to this upgrade | 3 |
| • Demonstrates good understanding of EITHER ONE advantage OR ONE disadvantage | 2 |
| • Demonstrates basic understanding of ONE advantage or ONE disadvantage | 1 |

***Sample answer:***

A single developer leads to minimised communication issues, saving time and avoiding misinterpretation. The single developer does not need to explain to team members specific requirements for the design and implementation. This is particularly relevant for the range of disabled users.

However, the single developer may not be skilled in all required areas (eg security of member data, designing inclusive interfaces) leading to a poorer quality system.

***Answers could include:***

Advantage of a single developer
Easier management of the process – a single developer manages their own timeline, independent of others. Less stress on the developer.

Disadvantage of a single developer
A single developer may not be able to complete the project within the required time frame required by the travel company.

## Question 28

| Criteria | Marks |
|---|---|
| • Provides a substantially correct algorithm that addresses the following features:<br>– Input a string<br>– Determine its length<br>– Calculate number of characters each side of the middle 10<br>– Use appropriate routines<br>– Use correct parameters<br>– Display encrypted string | 4 |
| • Provides a substantially correct algorithm that addresses most of the features | 3 |
| • Provides an algorithm that attempts to address some of the features | 2 |
| • Shows some understanding of the problem | 1 |

***Sample answer:***

```
BEGIN Encrypt
    Get string
    len = Length(string)
    Step1i = Middle(string, 10)
    Step1ii = Flip(Step1i)
    side = (len-10)/2
    Step2a = Left(string, side)
    Step2bi = Flip(Step2a)
    Step2bii = Append(Step2bi, Step1ii)
    Step3a = Right(string, side)
    Step3bi = Flip(Step3a)
    Step3bii = Append(Step2bii, Step3bi)
    Display Step3bii
END
```

## Question 29 (a)

| Criteria | Marks |
|---|---|
| • Describes TWO methods relevant to this subroutine | 3 |
| • Describes ONE method relevant to this subroutine<br>OR<br>• Outlines TWO methods | 2 |
| • Shows some understanding of error identification | 1 |

***Sample answer:***

Insert debugging output statements at appropriate points in the code to display the value of Swapped and the two elements being compared, inside the loop at line 55 and after the comparison at line 115.

Insert a breakpoint at the end of each pass (line 135) so that the contents of the array can be checked at that point.

## Question 29 (b)

| Criteria | Marks |
|---|---|
| • Justifies relevant changes | 2 |
| • Shows some understanding of the sort process | 1 |

***Sample answer:***

Change line 30 to **Set Swapped to True** to ensure the loop is entered at the start of the sort.

Insert line 45 **Set Swapped to False** to reset the Swapped flag at the start of each pass.

Remove line 130 to ensure array index doesn't exceed length of array.

## Question 30

| Criteria | Marks |
|---|---|
| • Provides a substantially correct algorithm that addresses the following features:<br>– Loop through all the player scores<br>– Effectively process a new player<br>– Keep track of the indices for both arrays<br>– Transfer data appropriately | 5 |
| • Provides a substantially correct algorithm that addresses most of the features | 4 |
| • Provides an algorithm that addresses some of the features | 3 |
| • Provides an algorithm that addresses one of the features | 2 |
| • Shows some understanding of the problem | 1 |

***Sample answer:***

```
 'Note:   FRow = current row in First
 '        SRow = current row in Second
 '        Column = position of current score along Second for this current player

BEGIN Transpose
        InitFirstElement
        For FRow = 2 to 50
            CheckIfNextPlayerSame
            MoveThisScore
        NEXT FRow
END Transpose

BEGIN   InitFirstElement
        FRow = 1
        SRow = 1
        Column = 2
        MovePlayerId
End InitFirstElement


BEGIN MovePlayerId
    Second (SRow, 1) = First (FRow, 1)
    MoveThisScore
END MovePlayerId

BEGIN CheckIfNextPlayerSame
        IF First (FRow, 1) = First (FRow – 1, 1) THEN
            Column = Column + 1
        ELSE
            SRow = SRow + 1
            Column = 2
            MovePlayerId
        END IF
END CheckIfNextPlayerSame

BEGIN MoveThisScore
            Second (SRow, Column) = First (FRow, 2)
END MoveThisScore
```

## Section III

### Question 31 (a)

| Criteria | Marks |
|---|---|
| • Demonstrates understanding of relevant OOP concept(s) with respect to maintenance of software | 2 |
| • Demonstrates some understanding of the OOP paradigm | 1 |

***Sample answer:***

Due to encapsulation, any objects not affected by changes in requirements do not need to be considered by the programmer. This allows them to focus specifically on the methods or attributes requiring change.

### Question 31 (b) (i)

| Criteria | Marks |
|---|---|
| • Distinguishes between polymorphism and inheritance in this system, with relevant examples | 4 |
| • Shows some understanding of polymorphism AND inheritance, with relevant example(s) | 3 |
| • Shows some understanding of polymorphism AND/OR inheritance | 2 |
| • Shows some understanding of the code provided | 1 |

***Sample answer:***

Inheritance allows the definition of a subclass such as the TICKETTYPE that has the same properties as the super class TICKET with additional properties as required, such as discounted tickets.

On the other hand, polymorphism, while still a characteristic of OOP, allows the same method to process data differently, depending on the circumstances. Although we call the method GetDiscount(), the logic will depend on the specific type of discount available for the ticket being purchased.

***Answers could include:***

GetEvent_name() occurs in both classes but the logic is different.

## Question 31 (b) (ii)

| Criteria | Marks |
|---|---|
| • Provides correct code | 2 |
| • Provides some relevant code | 1 |

***Sample answer:***

```
212  IF Evidence_sighted is FALSE THEN
214          Discount = 0
216  END IF
```

## Question 31 (c) (i)

| Criteria | Marks |
|---|---|
| • Provides a correct definition and relevant example for each of the three concepts | 3 |
| • Shows an understanding of most concepts | 2 |
| • Shows some understanding of one concept | 1 |

***Sample answer:***

Facts are statements that are assumed to be true, such as “Mary has a temperature”.

Rules are used to determine new facts, based on existing facts and user input, such as “IF Mary has a temperature AND Mary has a rash THEN Mary has measles”.

A query is used to determine relevant goals using the inference engine, based on user input, for example “What disease does Mary have?”

## Question 31 (c) (ii)

| Criteria | Marks |
|---|---|
| • Provides a thorough description of the use of pattern matching in this system | 3 |
| • Demonstrates understanding of the use of pattern matching in this system | 2 |
| • Demonstrates some understanding of pattern matching | 1 |

***Sample answer:***

Pattern matching could be used to compare stored images of typical rashes with a photo of a patient’s rash. The software recognises commonalities in all previously scanned images of identified rashes. The new unknown rash is scanned to see if the same commonalities exist, and the probability of a match with an identified rash can be determined. This allows probable diagnoses (eg 90% chance of measles) to be made.

## Question 31 (d)

| Criteria | Marks |
|---|---|
| • Provides a description of the potential use of AI in this context | 3 |
| • Demonstrates some understanding of AI | 2 |
| • Demonstrates some understanding of the role of the software | 1 |

***Sample answer:***

AI can make use of input (from the car's sensors as it moves along the route) and use the logic paradigm based software to compare it to large amounts of previously collected data (where observed good driving practice is expressed as facts and rules), so that the car appears to make appropriate decisions while driving.

## Question 31 (e)

| Criteria | Marks |
|---|---|
| • Provides a thorough justification for their response (yes or no) | 3 |
| • Provides justification for their response<br>OR<br>• Shows understanding of the imperative paradigm | 2 |
| • Shows limited understanding of the imperative paradigm | 1 |

***Sample answer:***

Yes. Not all systems lend themselves to being encoded using alternative paradigms such as OOP or logic. There may be no need to identify objects and their classes, or it may not be possible to solve the problem by specifying facts and rules. It may be more productive to use the imperative paradigm to code each of the logical steps required, such as an encryption process.

## Question 32 (a)

| Criteria | Marks |
|---|---|
| • Provides the correct calculation with carry bits shown | 2 |
| • Shows some understanding of the process | 1 |

***Sample answer:***

$$\begin{array}{ccccc} & 1^{1} & 0 & 1^{1} & 1 \\ & 0 & 1 & 1 & 1 \\ & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 \end{array}$$

## Question 32 (b)

| Criteria | Marks |
|---|---|
| • Provides an outline of the processes used in all three operations with reference to flipping, shifting and adding | 3 |
| • Provides an outline of the processes used in two of the operations | 2 |
| • Shows limited understanding of one operation | 1 |

***Sample answer:***

Subtraction can be done by adding the two’s complement (obtained by flipping bits and adding 1).

Multiplication is achieved by shifting and adding the bits of the multiplier.

Division is achieved by repetitive shifting of the divisor and subtraction to produce remainders.

## Question 32 (c)

| Criteria | Marks |
|---|---|
| • Provides a detailed explanation for all three interpretations | 4 |
| • Provides a detailed explanation for two interpretations<br>OR<br>• Provides an explanation for three interpretations | 3 |
| • Provides a detailed explanation for one interpretation<br>OR<br>• Provides an explanation for two interpretations | 2 |
| • Shows some understanding of one interpretation | 1 |

***Sample answer:***

Each group of 8 bits (1 byte) can represent an ASCII character, with the first byte specifying three characters in the string to follow.

The bytes can represent a single long integer, with the first bit possibly being the sign. The equivalent decimal value is calculated by adding powers of 2.

The bytes can represent a single precision floating point number consisting of a sign bit, exponent and mantissa.

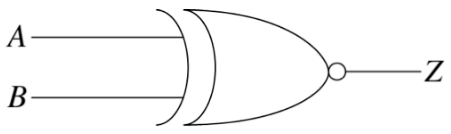
## Question 32 (d)

| Criteria | Marks |
|---|---|
| • Provides a correct circuit with only one gate and a correct truth table | 3 |
| • Provides a substantially correct simpler circuit with a relevant truth table | 2 |
| • Shows some understanding of logic circuits | 1 |

***Sample answer:***

| A | B | V | W | X | Y | Z |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |

| A | B | NOT XOR |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



## Question 32 (e) (i)

| Criteria | Marks |
|---|---|
| • Provides a correct data stream | 2 |
| • Provides a partially correct data stream | 1 |

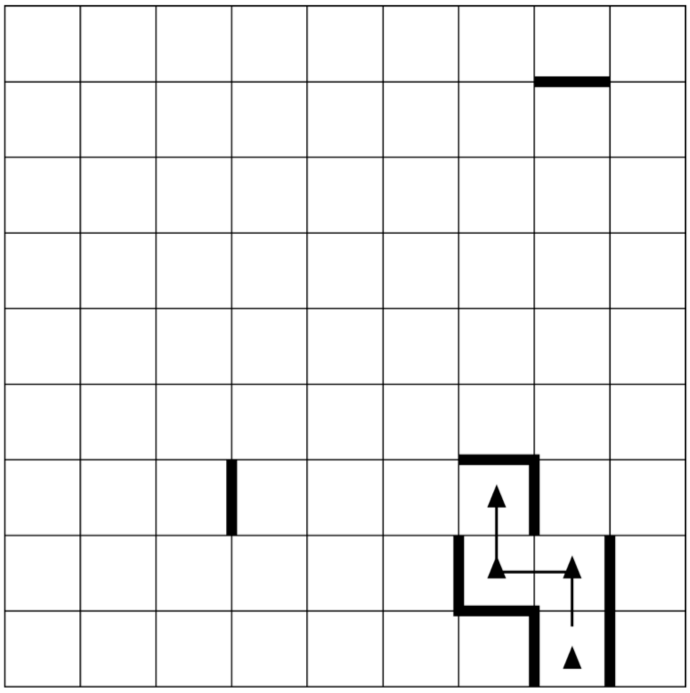
***Sample answer:***

1 0011 0011 0101 1111 1

## Question 32 (e) (ii)

| Criteria | Marks |
|---|---|
| • Provides a correct drawing, showing all movements and known walls | 3 |
| • Provides a partially correct drawing | 2 |
| • Demonstrates some understanding of the problem | 1 |

***Sample answer:***



## Question 32 (e) (iii)

| Criteria | Marks |
|---|---|
| • Explains the required changes for the drone and both data streams | 3 |
| • Identifies some relevant changes to the drone and/or data streams | 2 |
| • Shows some understanding of the requirements | 1 |

***Sample answer:***

The drone will require two more sensors to detect walls above and below, as well as changes to the motor to allow up and down movement.

The data stream sent from the drone requires two more 4-bit components to transmit the distance to the nearest wall above and below the drone.

The data stream sent from the computer requires the direction component to be increased to 3 bits to allow up and down movement, for example

000 – back 001 – right 010 – left 011 – forward 100 – up 101 – down

# 2019 HSC Software Design and Development Mapping Grid

## Section I

| Question | Marks | Content | Syllabus outcomes |
|---|---|---|---|
| 1 | 1 | 9.1.2 Approaches | H1.2 |
| 2 | 1 | 9.2.4 Volume testing | H4.2 |
| 3 | 1 | 9.1.2 Outsourcing | H1.2 |
| 4 | 1 | 9.2.4 Post-implementation review | H5.2 |
| 5 | 1 | 9.1.2 CASE tools | H1.2 |
| 6 | 1 | 9.2.2 Data dictionary | H5.2 |
| 7 | 1 | 9.2.2 Search/complete the algorithm | H4.2 |
| 8 | 1 | 9.2.3 Use of flags | H4.2 |
| 9 | 1 | 9.2.3 Compilation/interpretation | H4.2 |
| 10 | 1 | 9.2.2 Control structure (convert post-test to pre-test) | H4.2 |
| 11 | 1 | 9.1.2 Methods of installation | H1.2 |
| 12 | 1 | 9.2.3 Fetch-execute cycle | H1.3 |
| 13 | 1 | 9.2.2 Converting documentation | H5.2 |
| 14 | 1 | 9.2.4 Test data | H4.2 |
| 15 | 1 | 9.2.2 Sequential vs relative files | H1.3 |
| 16 | 1 | 9.2.2 Local/global variables | H4.2 |
| 17 | 1 | 9.2.3 Syntax (English to EBNF) | H4.2 |
| 18 | 1 | 9.2.2 Algorithm, random number generator | H4.2 |
| 19 | 1 | 9.2.2 Algorithm for data validation | H4.2 |
| 20 | 1 | 9.2.2 Sort | H4.2 |

## Section II

| Question | Marks | Content | Syllabus outcomes |
|---|---|---|---|
| 21 | 3 | 9.2.2 Developing software for specific devices | H2.2 |
| 22 | 3 | 9.1.1 Software market – dominant players | H3.1 |
| 23 (a) | 2 | 9.2.2 Scenario – student enrolment<br>Compare and contrast arrays and records | H4.2 |
| 23 (b) (i) | 4 | 9.2.2 Documentation – screen design | H6.4, H5.2 |
| 23 (b) (ii) | 3 | 9.2.1, 9.2.2 Documentation – system flowchart | H5.2 |
| 23 (b) (iii) | 3 | 9.2.2 Justification of search methods | H4.2 |
| 24 (a) | 2 | 9.2.2 Algorithm – simple desk check | H4.2 |
| 24 (b) | 3 | 9.2.2 Improving code efficiency | H4.2 |
| 25 (a) | 3 | 9.1.2 Development approach | H1.2 |
| 25 (b) | 3 | 9.2.4 Software evaluation | H4.3, H5.1 |
| 25 (c) | 4 | 9.2.5 Maintenance | H4.2, H5.1 |

| Question | Marks | Content | Syllabus outcomes |
|---|---|---|---|
| 26 | 4 | 9.2.1 Documentation – interpreting a structure chart | H5.2 |
| 27 (a) | 3 | 9.1.1, 9.2.1 Scenario – travel company<br>Meeting client needs | H4.1 |
| 27 (b) | 3 | 9.2.2, 9.2.3 Choosing a programming language | H2.1 |
| 27 (c) | 3 | 9.3, 9.1.2 Team of developers vs single developer | H6.3 |
| 28 | 4 | 9.2.2 Algorithm based on subroutines provided | H4.2 |
| 29 (a) | 3 | 9.2.3 Sorting subroutine – debugging methods | H4.2 |
| 29 (b) | 2 | 9.2.2 Fixing errors in subroutine | H4.2 |
| 30 | 5 | 9.2.2 Algorithm – processing 2D array | H4.2 |

**Section III**

| Question | Marks | Content | Syllabus outcomes |
|---|---|---|---|
| 31 (a) | 2 | 9.4.1 Implication of OOP for testing/maintenance | H1.2, H2.1 |
| 31 (b) (i) | 4 | 9.4.1 Distinguish between inheritance and polymorphism | H1.2, H2.1 |
| 31 (b) (ii) | 2 | 9.4.1 Modify OOP code | H4.2 |
| 31 (c) (i) | 3 | 9.4.1 Distinguish between facts, rules and queries | H4.1 |
| 31 (c) (ii) | 3 | 9.4.1 Use of pattern matching in described system | H4.2 |
| 31 (d) | 3 | 9.4.1 Use of AI | H4.2 |
| 31 (e) | 3 | 9.4.1 Current role of the imperative paradigm | H2.1, H4.1 |
| 32 (a) | 2 | 9.4.2 Binary arithmetic – calculation | H1.1, H1.3 |
| 32 (b) | 3 | 9.4.2 Binary arithmetic – role of shift, add, flip | H1.1, H1.3 |
| 32 (c) | 4 | 9.4.2 Data representation | H1.1, H1.3 |
| 32 (d) | 3 | 9.4.2 Simplifying a provided circuit | H1.1, H1.3 |
| 32 (e) (i) | 2 | 9.4.2 Construct a data stream | H1.1, H1.3 |
| 32 (e) (ii) | 3 | 9.4.2 Interpret data streams | H1.1, H1.3 |
| 32 (e) (iii) | 3 | 9.4.2 Modify a data stream | H1.1, H1.3 |