2020 HSC Software Design and Development Marking Guidelines

Section I

Multiple-choice Answer Key

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

Section II

Question 21 (a)

Criteria	Marks
• Describes how to address TWO relevant issues	3
• Describes how to address a relevant issue OR • Outlines TWO relevant issues	2
• Identifies a relevant issue	1

Sample answer:

The user interfaces could be designed with appropriate font types, font sizes and layout for ease of legibility for visually impaired customers. A range of languages could be offered for restaurants in non-English speaking areas.

Since the system allows online payment, it is critical that the system stores customers' banking details securely. A firewall would prevent unauthorised access to the server, and passwords would protect individual customer accounts.

Question 21 (b)

Criteria	Marks
• Describes TWO uses of CASE tools relevant to the development of this application	3
• Describes ONE use of a CASE tool relevant to the development of this application OR • Identifies TWO uses of CASE tools relevant to the development of this application	2
• Demonstrates a basic understanding of CASE tools	1

Sample answer:

Version control allows team members to work on the latest versions of common modules, and prevents members from mistakenly making changes to older versions. The CASE software can also 'roll back' to a previous working version if necessary. Generation of test data such as restaurant locations and delivery locations will help to ensure that the fastest-route calculations are correct over a wide range of data.

Question 21 (c)

Criteria	Marks
• Outlines the impact on the development process	2
• Demonstrates some understanding of the Gantt chart	1

Sample answer:

If Task 1 runs late, Task 2 (to be done by Bill) and Task 3 (to be done by Adam) will be delayed. If Adam cannot complete Task 3 by the end of week 2 because he is still working on Task 1, he may not be available to work with Cara in Task 5. The whole project may be delayed.

Question 21 (d)

Criteria	Marks
• Shows a thorough understanding of how benchmarking could be used to assess the performance of the application	3
• Shows some understanding of how benchmarking could be used to assess performance	2
• Shows a basic understanding of benchmarking	1

Sample answer:

Benchmarking involves looking at the performance of systems using similar technologies under controlled conditions. For this system, the developers could look at the relative performance of a range of fastest route determining algorithms, thereby assessing their system in terms of speed of food delivery.

Question 22

Criteria	Marks
• Explains why the software development cycle is called a 'cycle' with reference to relevant stages	3
• Shows some understanding of why the software development cycle is called a 'cycle' with reference to some stages	2
• Shows a basic understanding of the software development cycle	1

Sample answer:

During the testing and evaluating stage, problems may be identified that require changes to the software. There may be modifications to the logic, interfaces or functionality. Similarly in the maintenance stage, new requirements may be identified. In either case the modifications will require clarification, which is achieved during the defining and understanding stage, beginning the next cycle. Hence it is called a cycle.

Question 23

Criteria	Marks
• Provides a thorough justification of the use of an interpreter and/or a compiler referencing both development and distribution of a software package	4
• Shows a sound understanding of the use of an interpreter and/or a compiler in both the development and distribution of a software package	3
• Shows some understanding of an interpreter and/or a compiler	2
• Shows a basic understanding of an interpreter or a compiler	1

Sample answer:

During development an interpreter should be used as it enables execution of partially completed code. Because interpreters translate code line by line, they identify syntax errors as they are encountered. If a compiler is used, a list of all syntax errors is displayed, which may be overwhelming for the developer.

The developer should compile the software for distribution. Executable programs are distributed in machine code form, which executes faster and helps to minimise piracy. Users do not require additional programs to run the package.

Question 24 (a)

Criteria	Marks
• Explains TWO benefits of consistency in interface design for the games package	3
• Explains ONE benefit of consistency in interface design OR • Outlines TWO benefits of consistency in interface design	2
• Identifies a benefit of consistency in interface design	1

Sample answer:

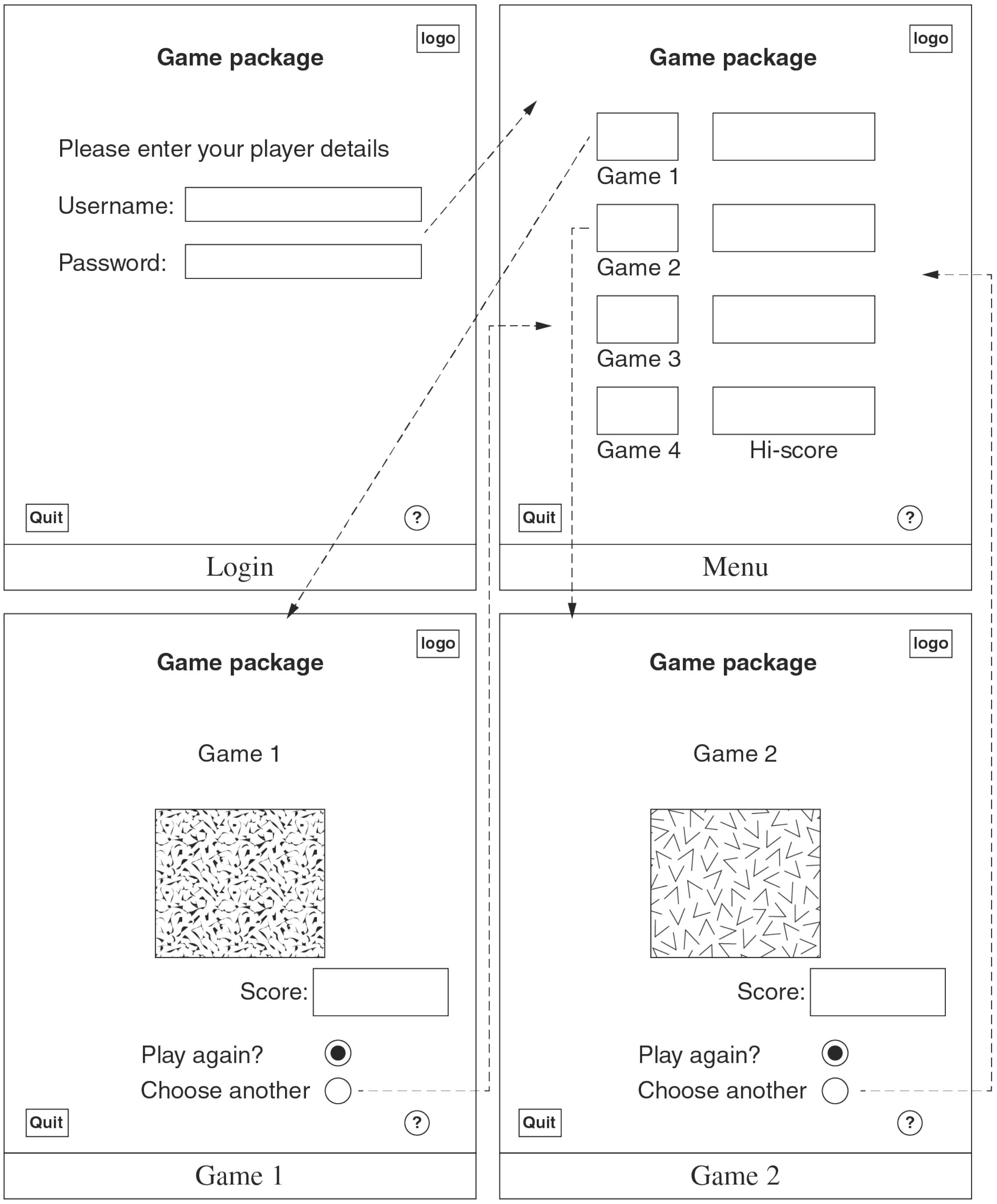
Consistency ensures all common buttons / clickable areas are named and located similarly across all relevant screens making the games easier to use.

By using a common banner with the same colours, font types, styles and colours on each screen, development time is reduced as a common module can be called to display this information.

Question 24 (b)

Criteria	Marks
• Produces a substantially correct storyboard that demonstrates consistent interface design and appropriate navigation between the screens	3
• Produces a storyboard that demonstrates a sound understanding of the interfaces and navigation required	2
• Shows some understanding of a storyboard or the interfaces required	1

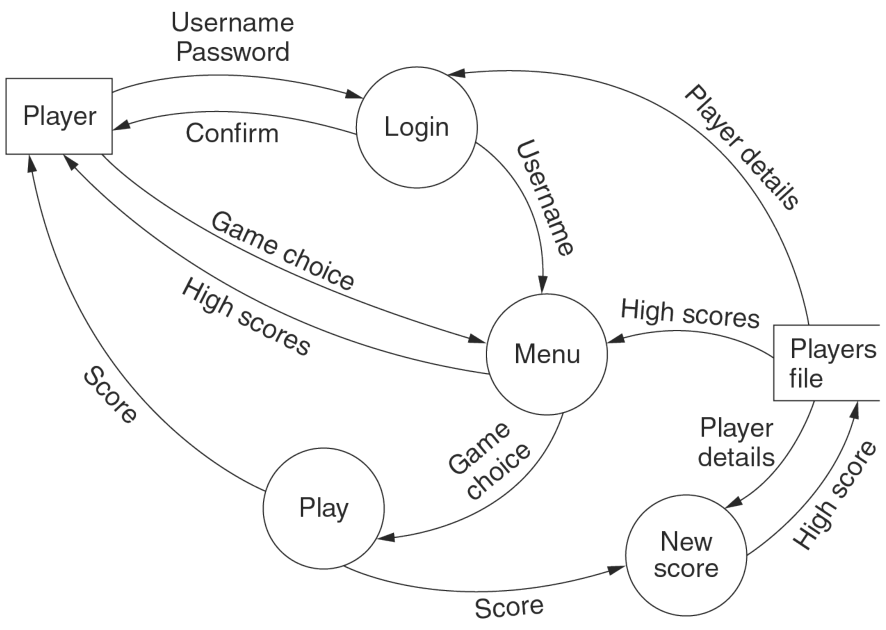
Sample answer:



Question 24 (c)

Criteria	Marks
• Provides a substantially correct data flow diagram that includes several processes, data flows, storage and external entity	4
• Provides a data flow diagram that is relevant to the games package • Includes some processes, data flows and external entity	3
• Provides a data flow diagram with some elements relevant to the games package	2
• Shows a basic understanding of data flow diagrams	1

Sample answer:



Question 25 (a)

Criteria	Marks
• Describes ONE benefit of using a stub in the scenario	2
• Shows a basic understanding of a stub	1

Sample answer:

The CalculateCost module can be executed and tested without having to write and test the FindDiscount module. This will help if the necessary FindDiscount logic is complex or incorrect or the file does not yet exist.

Question 25 (b)

Criteria	Marks
• Produces a substantially correct data dictionary	3
• Produces a data dictionary with several correct entries	2
• Shows a basic understanding of a data dictionary	1

Sample answer:

Data item	*Data type*	*Number of bytes required for storage*
cost_of_item	Floating point	4
total	Floating point	4
more	Text	3
CustID	Long integer	4
discount	Floating point	4

Answers could include:

CustID may be text.

Question 25 (c)

Criteria	Marks
• Identifies issues and provides points for and against the use of a sequential file and a relative file in the scenario	3
• Shows some understanding of a sequential and/or relative file	2
• Shows a basic understanding of a sequential and/or relative file	1

Sample answer:

If there are a large number of stored customer records, the use of a sequential file may take too long to find a particular customer. If the details in the customers' records are updated often, a sequential file will also not be appropriate, as it will have to be rewritten each time changes are made. It is much more efficient to use a relative file as each record can be more quickly retrieved and modified in place within the file.

To access a relative file, the CustID must be converted to an appropriate integer denoting the relative position of the record in the file. If it is not possible to design and implement appropriate logic to do this, a relative file cannot be used.

Question 26 (a)

Criteria	Marks
• Performs correct desk checking	2
• Shows a basic understanding of desk checking	1

Sample answer:

x	y	message	Display
		“”	enter 2 multiples of 10
60	70	Boundary	Boundary
		“”	enter 2 multiples of 10
150	30		

Question 26 (b)

Criteria	Marks
• Provides TWO appropriate sets of coordinates that test different conditions and provides justification	3
• Provides ONE appropriate set of coordinates that tests a different condition with justification OR • Provides TWO appropriate sets of coordinates that test different conditions without justification	2
• Shows a basic understanding of the problem	1

Sample answer:

(100,200) – This is outside the screen area. It tests another pathway.

(80,60) – This is inside the target area. It tests another pathway.

Question 26 (c)

Criteria	Marks
• Provides a substantially correct algorithm that includes the following features: – inputs – a loop – selection – appropriate error messages	4
• Provides an algorithm that addresses the key aspects of the problem and contains several of the features	3
• Provides an algorithm that shows some understanding of the problem	2
• Shows a basic understanding of the problem	1

Sample answer:

```
BEGIN GetValid(x,y)
    valid = FALSE
    REPEAT
        Input x
        Input y
        IF x mod 10 <> 0 OR y mod 10 <> 0 THEN
            print “Coordinates must be multiples of 10”
        ELSE
            IF x < 0 OR x > 200 THEN
                Print “outside screen”
            ELSE
                IF y < 0 OR y > 100 THEN
                    Print “outside screen”
                ELSE
                    valid = TRUE
                ENDIF
            ENDIF
        ENDIF
    UNTIL valid = TRUE
END GetValid(x,y)
```

Question 27 (a)

Criteria	Marks
• Justifies an appropriate software development approach	3
• Outlines the features of a relevant software development approach	2
• Shows some understanding of a software development approach	1

Sample answer:

Prototyping allows the school executive to be involved in the development, by providing opportunities to give the developers feedback on functionality and interface. The developers can adjust the design of the software to better meet the school's needs. The 12 month time frame allows time to gather and respond to this feedback.

Question 27 (b)

Criteria	Marks
• Describes methods to both isolate the specific modules and the specific lines of code	4
• Describes a method to isolate the specific modules OR the specific lines of code, and shows some understanding of the other method	3
• Outlines how to isolate the specific modules and/or the specific lines of code	2
• Shows a basic understanding of debugging	1

Sample answer:

The developer reads the code to determine which modules are involved in printing tickets and calculating the amount charged. A series of debugging output statements can be inserted in these modules to reveal the current values of critical variables to identify the variables that contain incorrect values. Code in the module can be examined to see where the variables were incorrectly set, to identify the lines causing the issue.

The developer may also use breakpoints at the start of the suspect modules, followed by single line stepping to show the outcome of each line of code, to identify how the variable contents are changing, thus determining the line(s) of code causing the error.

Question 27 (c)

Criteria	Marks
• Provides a substantially correct algorithm that addresses the following features: – accepts input – searches the array for appropriate seats – updates the array – displays allocated seats – provides suitable messages	5
• Provides an algorithm that addresses most of the booking and seat allocation requirements	4
• Provides an algorithm that addresses some booking and seat allocation requirements	3
• Provides an algorithm that addresses an aspect of the booking and seat allocation requirements	2
• Shows some understanding of the problem	1

Sample answer:

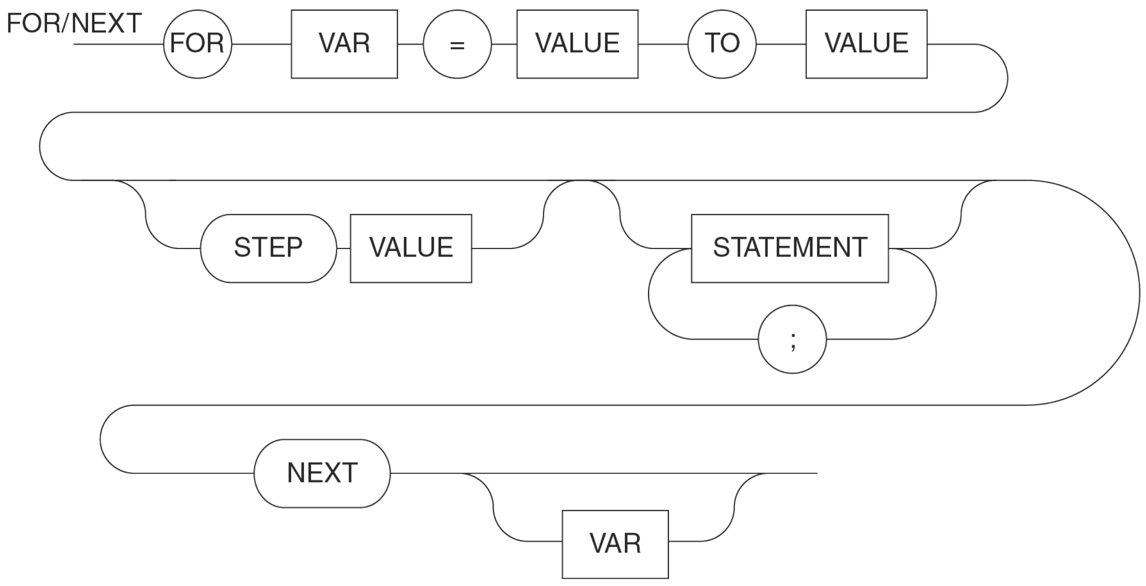
R=row, S=seat, N=number of seats needed

```
BEGIN
    REPEAT
        R = 0
        input N
        Found = FALSE
        WHILE R < 14 AND Found = FALSE
            R = R + 1
            S = 0
            WHILE S < = 10-N AND Found = FALSE
                S = S + 1
                Total = 0
                Index = 0
                WHILE Index < N AND Found = FALSE
                     IF Grid(R,S+index) = “” THEN
                     Total = Total + 1
                    END IF
                     Index = Index+1
                     IF Total = N THEN
                     Found = TRUE
                     ENDIF
                ENDWHILE
            ENDWHILE
        ENDWHILE
        IF Found = FALSE THEN
            print “Sorry – that number of seats is not available”
            again = TRUE
        ELSE
            FOR loop = 0 TO N–1
                print Grid(R,0), Grid(0,S+loop)
                Grid(R,S+loop)=”X”
            NEXT
            again = FALSE
        ENDIF
    UNTIL again = FALSE
END
```

Question 28

Criteria	Marks
• Produces a substantially correct railroad diagram that caters for all structures	3
• Produces a partially correct railroad diagram that caters for some of the structures	2
• Shows a basic understanding of railroad diagrams relevant to the programming language	1

Sample answer:



Section III

Question 29 (a)

Criteria	Marks
• Explains the relevance of OOP in developing the computer game	3
• Demonstrates some understanding of the use of OOP in developing the computer game	2
• Shows a basic understanding of OOP	1

Sample answer:

Since the computer game consists of various characters, these characters can be defined as objects. These characters or objects can be given common characteristics and they may inherit their attributes from a parent class. These characters or objects can also interact with each other in specified ways through defined methods.

Question 29 (b) (i)

Criteria	Marks
• Provides the correct fact and rule	3
• Provides a substantially correct extension to the code	2
• Shows some understanding of constructing a fact or a rule	1

Sample answer:

leader(benchmarking, lina)
two-managers(A, B, C) :- manager(A, C), manager(B, C), A ≠ B

Question 29 (b) (ii)

Criteria	Marks
• Describes how the query would be evaluated with correct reference to the relevant facts and rules	3
• Shows some understanding of how the query would be evaluated with reference to relevant facts and rules	2
• Shows some understanding of the problem	1

Sample answer:

To evaluate manager(X, adam) it scans the given facts but does not find a matching fact. It scans the rules and identifies from the manager rule provided that X is a manager of adam if X is a leader of a team and adam is a member of that team.

It then scans the member facts and finds that adam is a member of the prototyping team and also of the testing team.

Scanning the leader facts, it finds that prototyping is led by tracey and testing by sam. Tracey and sam are outputted as the result of the query.

Question 29 (c)

Criteria	Marks
• Shows a clear understanding of the differences between developing an expert system using the imperative paradigm and the logic paradigm	3
• Shows some understanding of how the imperative and/or the logic paradigm may be used to develop an expert system	2
• Shows a basic understanding of the imperative or the logic paradigm	1

Sample answer:

When using the imperative paradigm, a developer needs to code each conclusion available for the expert system explicitly. However, when using the logic paradigm, the developer need only define rules and facts for the expert system. The developer would then input queries into the system, which would use forward chaining to produce a list of possible outcomes based on its rules and facts.

Question 29 (d)

Criteria	Marks
• Explains how encapsulation can assist a team of programmers in a software project	3
• Shows some understanding of how encapsulation can be used in a software project	2
• Shows a basic understanding of encapsulation	1

Sample answer:

Encapsulation is used to hide components or values of an object within a class from other objects, which helps to restrict access to the object. This means data inside an object is not modified unexpectedly by external code in a completely different part of the program, written by a different programmer. Also, other programmers need only know what that object's methods will produce, without needing to know details about the object's internals in order to use or refer to it in their code.

Question 29 (e) (i)

Criteria	Marks
• Describes how polymorphism is used in the code with reference to line numbers	3
• Shows some understanding of polymorphism	2
• Shows a basic understanding of polymorphism	1

Sample answer:

The methods on lines 3 and 8 have the same name, but act differently when given different parameters. Calling the method 'operator' on line 16 executes different logic from calling the method 'operator' on line 17 as the parameters have different data types. Therefore, the processing of the method 'operator' changes based on the parameters provided when the method is called.

Question 29 (e) (ii)

Criteria	Marks
• Provides a substantially correct method	2
• Shows some understanding of the method required	1

Sample answer:

```
operator (int n, string str1) {
    outStr = “”
    FOR count = 1 TO n
        outStr = outStr + str1
    NEXT count
    Return outStr
}
```

Question 30 (a)

Criteria	Marks
• Provides a valid explanation	2
• Shows some understanding of ASCII or Unicode	1

Sample answer:

Unicode has the capacity to represent a much greater number of characters than ASCII, as it allocates more bits per character. Therefore it is preferable for representing the larger number of different characters that exist in a wide range of languages.

Question 30 (b) (i)

Criteria	Marks
• Provides a thorough description of how the security system operates	3
• Shows a sound understanding of how the security system operates	2
• Shows some understanding of how the security system operates	1

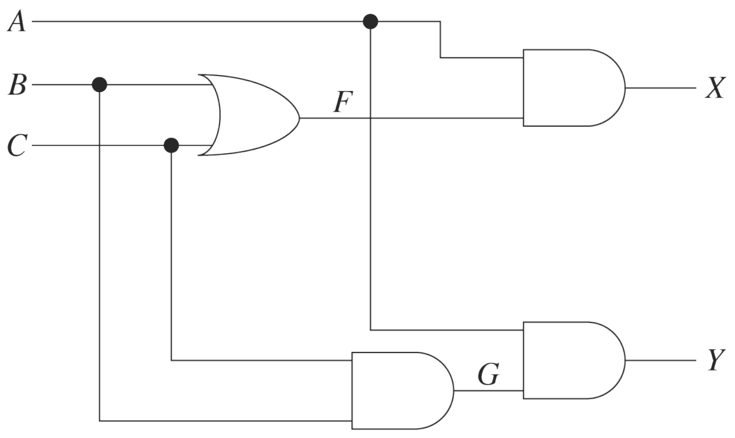
Sample answer:

The light turns on when the power is on and either or both of the sensors are on. The siren will only sound when the power is on and both sensors are on. Nothing happens if the power is off.

Question 30 (b) (ii)

Criteria	Marks
• Provides a correct circuit	3
• Provides a partially correct circuit	2
• Shows a basic understanding of the problem	1

Sample answer:



Question 30 (c) (i)

Criteria	Marks
• Provides a correct data stream	3
• Provides a data stream that partially addresses the problem	2
• Shows some understanding of the problem	1

Sample answer:

A 15 bit data stream:

Bit 1:	Start bit, always 0
Bits 2 & 3:	2 bits, identifies refrigerator
Bits 4 to 9:	6 bits, signed number representing temperature
Bits 10 to 14:	5 bits, number representing the hour
Bit 15:	Stop bit, always 1

Question 30 (c) (ii)

Criteria	Marks
• Explains how a flip-flop can be used in the situation	3
• Shows some understanding of how flip-flops work	2
• Identifies a feature of a flip-flop	1

Sample answer:

A flip-flop involves a feedback component in its circuitry, which ensures that the output will remain unchanged until a reset occurs.

The inputs for the flip-flop will be the single bit from the computer indicating that the critical temperature has been exceeded (SET) and a bit from the on/off switch that allows workers to turn off the light (RESET).

The output of the flip-flop is a signal to turn the light on (1) or off (0).

Question 30 (d) (i)

Criteria	Marks
• Provides a correct conversion with working	3
• Shows a sound understanding of the conversion process	2
• Shows some understanding of the conversion process	1

Sample answer:

Sign	1 means negative number	
Exponent	10000010	= 130
	– 127 bias	= 3
Mantissa	110100000…,	
Therefore number		$= 1.1101 \times 2^3$
		= 1110.1
		= 14.5

Number is –14.5

Question 30 (d) (ii)

Criteria	Marks
• Provides a substantially correct algorithm	3
• Provides a partially correct algorithm	2
• Shows some understanding of the problem	1

Sample answer:

```
BEGIN
    Value = 0
    extract characters 2 to 9 from String into Exp
    FOR Place = 1 to 8
        IF the Place^th character of Exp = "1" THEN
            Value = Value + 2^(8 - Place)
        ENDIF
    NEXT Place
    Value = Value – 127
    display Value
END
```

2020 HSC Software Design and Development Mapping Grid

Section I

Question	Marks	Content	Syllabus outcomes
1	1	9.2.1 Data types	H1.3, H4.2
2	1	9.2.3 Program documentation	H5.2
3	1	9.1.2 Software development approaches	H1.2
4	1	9.2.1 Development stages – boundaries	H4.1
5	1	9.2.2 Binary search	H4.2
6	1	9.2.2 Sorting	H4.2
7	1	9.2.2 Algorithm output	H4.2
8	1	9.2.3 Types of error	H4.2
9	1	9.2.4 Levels of testing and use of driver	H4.2
10	1	9.2.3 Metalanguage	H4.2
11	1	9.2.1 Quality assurance	H1.3
12	1	9.2.3 Railroad diagram and EBNF	H4.2
13	1	9.2.5 Maintenance	H4.2
14	1	9.2.2 Data structure	H1.3, H4.2
15	1	9.2.2 File handling	H4.2
16	1	9.2.3 Dynamic link libraries	H4.2
17	1	9.1.1 Decompilation	H3.1, H4.2
18	1	9.2.2 Flowchart and pseudocode	H4.2
19	1	9.2.3 Fetch – Execute cycle – code interpretation	H1.3
20	1	9.2.3 Fetch – Execute cycle – program counter	H1.3

Section II

Question	Marks	Content	Syllabus outcomes
21 (a)	3	9.1.1 Social issues	H3.1
21 (b)	3	9.1.2 CASE tools	H1.2
21 (c)	2	9.3 Project management – Gantt chart	H5.1
21 (d)	3	9.2.4 Benchmarking	H5.2
22	3	9.2 Software development cycle	H4.2
23	4	9.2.3 Interpretation and compilation	H1.2
24 (a)	3	9.2.2 Design consistency	H6.3
24 (b)	3	9.3 Documentation – storyboard	H5.2
24 (c)	4	9.2.1 Documentation – DFD	H5.2
25 (a)	2	9.2.3 Use of stubs	H4.2
25 (b)	3	9.2.1 Data dictionary	H5.2
25 (c)	3	9.2.2 File types	H4.2

Question	Marks	Content	Syllabus outcomes
26 (a)	2	9.2.2 Desk checking	H4.2
26 (b)	3	9.2.4 Test data	H4.2
26 (c)	4	9.3 Validation subroutine	H4.2
27 (a)	3	9.1.2 Software development approach	H1.2
27 (b)	4	9.2.3 Debugging methods	H4.2
27 (c)	5	9.3 Algorithm design	H4.2
28	3	9.2.3 Metalanguage	H4.2

Section III

Question	Marks	Content	Syllabus outcomes
29 (a)	3	9.4.1 OOP – games	H4.2
29 (b) (i)	3	9.4.1 Logic programming	H2.1
29 (b) (ii)	3	9.4.1 Logic programming	H4.2
29 (c)	3	9.4.1 Imperative vs logic – expert system	H4.2
29 (d)	3	9.4.1 OOP – encapsulation	H4.2
29 (e) (i)	3	9.4.1 OOP – polymorphism	H2.1
29 (e) (ii)	2	9.4.1 OOP – method	H2.1
30 (a)	2	9.4.2 Data representation – ASCII – Unicode	H1.3
30 (b) (i)	3	9.4.2 Interpreting a truth table	H1.3
30 (b) (ii)	3	9.4.2 Designing a circuit	H1.1
30 (c) (i)	3	9.4.2 Constructing a data stream	H1.1
30 (c) (ii)	3	9.4.2 Use of a flip-flop	H1.1
30 (d) (i)	3	9.4.2 Floating point to decimal conversion	H1.1
30 (d) (ii)	3	9.4.2 Design an algorithm for a partial conversion	H4.2