Diploma Programme
Programme du diplôme
Programa del Diploma

Markscheme

May 2022

Computer science

Higher level

Paper 1

18 pages

Subject details: Computer science HL paper 1 markscheme

Mark allocation

Section A: Candidates are required to answer **all** questions. Total 25 marks.
Section B: Candidates are required to answer **all** questions. Total 75 marks.
Maximum total = 100 marks.

General

A markscheme often has more specific points worthy of a mark than the total allows. This is intentional. Do not award more than the maximum marks allowed for that part of a question.

When deciding upon alternative answers by candidates to those given in the markscheme, consider the following points:

Each statement worth one point has a separate line and the end is signified by means of a semi-colon (;).

An alternative answer or wording is indicated in the markscheme by a "/"; either wording can be accepted.

Words in (…) in the markscheme are not necessary to gain the mark.

If the candidate's answer has the same meaning or can be clearly interpreted as being the same as that in the markscheme then award the mark.

Mark positively. Give candidates credit for what they have achieved and for what they have got correct, rather than penalizing them for what they have not achieved or what they have got wrong.

Remember that many candidates are writing in a second language; be forgiving of minor linguistic slips. In this subject effective communication is more important than grammatical accuracy.

Occasionally, a part of a question may require a calculation whose answer is required for subsequent parts. If an error is made in the first part then it should be penalized. However, if the incorrect answer is used correctly in subsequent parts then **follow through** marks should be awarded. Indicate this with "**FT**".

General guidance

Issue	Guidance
Answering more than the quantity of responses prescribed in the questions	In the case of an "identify" question, read all answers and mark positively up to the maximum marks. Disregard incorrect answers. In the case of a "describe" question, which asks for a certain number of facts *eg* "describe two kinds", mark the first two correct answers. This could include two descriptions, one description and one identification, or two identifications. In the case of an "explain" question, which asks for a specified number of explanations *eg* "explain two reasons …", mark the first two correct answers. This could include two full explanations, one explanation, one partial explanation *etc*.

Section A

1. *Award **[2 max].***
Language differences / lexical differences present across datasets to be merged;
Data representation differences / different data structures (e.g., date format, incompatible file formats);
Incompatible hardware;
Incompatible operating systems / different software versions;

[2]

2. *Award **[2 max]***
User efficiency;
To ensure that users know how to use the system correctly;

Support/Troubleshoot;
To provide users help when they encounter errors;

Accuracy;
To ensure the correct methods are used to enable reliable output;

Improved user experience;
the user is aware of all available features, so they can make the most out of the system;

[2]

3. *Award **[2 max]***
WAN covers a much larger area (national/international), LANs usually cover a smaller area (such as a single site);
Nodes connected to WANs often make use of connections through public networks (such as the telephone system), nodes connected to LANs are usually connected through private infrastructure;
LANs are more secure than WANs (due to how WANs transmit the data /how far the data would need to travel);
A higher bandwidth is available for transmission in a LAN than a WAN/ LANs can have a higher data transfer rate than a WAN;
LAN is typically cheaper than WAN to implement/ maintain (as the equipment required for LAN is less expensive);
WAN includes a large number of devices connected together, LAN includes less;
WANs are typically slower than LANs due to the distance data must travel;
WAN requires hardware to connect different networks, such as a router, LAN can be simple and does not need to connect to other networks;

[2]

4. *Award **[2 max]***
The reason for compression when transmitting data is to save on transfer times;
As it reduces the number of bits needed to represent data (when compared with the original data);

Compressing data involves modifying/restructuring files, so that they take up less space;
And this results in cost savings in cloud storage;

To take up less bandwidth;
Because data compression reduces the size of files to be transmitted over a network;

*Note to examiners: Award **[1]** for a reason (for example, to save data usage for sending files over the internet, to save storage capacity, to speed up file transfer, to decrease costs for network bandwidth, etc.), and award **[1]** for an expansion.*

[2]

5. *Award **[2 max]***
(The program written in HLL must be translated into machine code) so that the computer can execute the program;
as the computer only understands machine language / as code written in HLL can only be understood by humans and cannot be interpreted by the computers (which work in binary);

[2]

6. *Award **[4 max]***
*Award **[1]** for every two correct rows*

A	B	C	P	Q	X
0	0	0	0	1	1
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	1

[4]

7. (a) *Award **[2 max]***
*Award **[1]** for showing workings.*

(0)1101001;

[2]

(b) *Award **[2 max]***
*Award **[1]** for showing workings.*

C8;

[2]

8. *Award **[3 max]***
scheduling;
policies;
multitasking;
virtual memory;
paging;
interrupt;
polling;

[3]

9. (a) *Award **[1 max]**.*
A linear (abstract) data structure;
First in First out/FIFO;
Elements can only be added at one end(rear/tail) and removed from the other(front/head);
Once a new element is inserted into the queue (enqueue), all the elements inserted before the new element in the queue must be removed (dequeue), to remove the new element;

[1]

(b) *Award **[1 max]**.*
Print queue;
CPU task scheduling;
Simulation of a real-life scenario (call centre, supermarket queue);
Playlist queue;
Interrupt queues;
Queues in routers and switches;
Mail queues;
Buffers in devices like keyboard;
Website traffic;

Note to examiners: Reward other reasonable responses.

[1]

10. *Award **[2 max]***
Child (is either null or) is a node that has up to two references /links to other nodes;
and has (only) one predecessor (parent) node;

"Child" is every node in a binary tree which is descendant/ the node which has a link from its predecessor (the node which is a predecessor of any node is called as parent node);
The child node can have up to two descendants (child nodes);
(All the nodes except the node which is the origin of the binary tree data structure (called root) are child nodes;)

[2]

Section B

11. (a) (i) *Award **[1 max]***
End users/ employees/ customers/ community members/ media/ suppliers;
Business owners/ managers/ shareholders/ investors;

[1]

(ii) *Award **[2 max]***
User dissatisfaction;
because the system does not meet user requirements;

Developers not being paid for the final product;
as the business owner requests are not evident in the final product/ or outside of the project's scope;

Unsuccessful final product;
the developed system may either solve a different problem/ is not user friendly as compared to the existing system;

Note to examiners: Reward other reasonable responses.

[2]

(b) *Award **[4 max]***
Examining current systems (using interviews/ surveys/direct observation);
To compare the existing system against possible requirements to identify missing features;

Examining competing products;
To compare own system with competitors to enable decisions on features to add;

Review of organizational capabilities;
To determine how well the organization manages resources to gain an advantage over competitors;

Literature searches;
To research current methods and to help inform development choices;

Mark as 2 and 2

[4]

(c) *Award **[2 max]***
Testing is important (at every stage) to make sure the system operates in line with user requirements/as intended;
To prevent the end user being dissatisfied with the final system;

Testing is important to enable early discovery of errors;
to reduce time delay/ using more resources / avoid higher cost;

[2]

(d) *Award **[6 max]***
*Award **[1]** for method, award **[1]** for benefit and award **[1]** for drawback.*

Parallel running;
The old and new systems run together, so if a problem is found with the new system, it can be repaired/the old system can take over;
This is expensive as duplicate systems and staff are needed;

Pilot running;
The new system is only implemented in one branch of the organization so disruption is kept to a minimum;
It can take a long time for the new system to be fully implemented / two systems are still in operation within the organization, leading to duplication and possible errors;

Direct changeover;
The new system is implemented overnight so the changes happen very quickly;
If the new system fails, the company has no working system to fall back on;

Phased conversion;
Only one area/department/part of the system is updated at a time, so the disruption is kept to a minimum;
Multiple systems which may not be compatible with each other will be running at the same time;

Note to examiners: Accept other suitable examples of benefits and drawbacks.

Mark as 3 and 3.

[6]

12. (a) *Award **[2 max]***

```
if;
then;
else;
```

Note to examiners: allow an alternative descriptive version such as:
test/condition;
action/consequence;
(optional) alternative action/consequence;

[2]

(b) *Award **[5 max]***
*Award **[1]** for an appropriate loop with correct loop parameters to cover 30 array elements/all students*
*Award **[1]** for correct use of indexes in two arrays* (`MARK` *and* `GRADE`)
*Award **[1]** for each if statement with correct condition and grade assignment up to **[4]**.*

Note to examiners: *Award **[4]** if candidate has correctly used an alternative conditional statement such as switch/ case.*

Example answer 1:

```
loop COUNTER from 0 to 29
    if MARK[COUNTER] >= 80
      then GRADE[COUNTER] = "Distinction"
      else
        if MARK[COUNTER]>= 60
          then GRADE[COUNTER] = "Merit"
          else
            if MARK[COUNTER} >= 40
               then GRADE[COUNTER] = "Pass"
               else
                    GRADE[COUNTER] = "Fail"
            end if
        end if
    end if
end loop
```

Example answer 2:

```
COUNTER = 1
loop while COUNTER <= 30
    if MARK[COUNTER-1] >= 80
      then GRADE[COUNTER-1] = "Distinction"
    end if
    if MARK[COUNTER-1] >= 60 and MARK[COUNTER-1] < 80
          then GRADE[COUNTER-1] = "Merit"
    end if
    if MARK[COUNTER-1} >= 40 and MARK[COUNTER-1] < 60
            then GRADE[COUNTER-1] = "Pass"
    end if
    if MARK[COUNTER-1} < 40
              GRADE[COUNTER-1] = "Fail"
    end if
    COUNTER = COUNTER + 1
end loop
```

[5]

(c) *Award **[2 max]***
Three arrays are parallel/ they have the same number of elements/ the same length;
the same array index can be used to represent name, grade and mark of the same student/ the array index makes sure that data from the three arrays lines up;

[2]

(d) *Award **[3 max]***
*Award **[1]** for correct loop to check all students*
*Award **[1]** for correct conditional statement checking correct array*
*Award **[1]** for correct output*

Example answer 1:

```
loop COUNTER from 0 to 29
    if MARK[COUNTER] >= 60 then
      output NAME[COUNTER], GRADE[COUNTER]
    end if
end loop
```

Example answer 2:

```
loop C from 0 to 29
  if GRADE[C].equals("Merit")OR GRADE[C].equals("Distinction")
       then
          output NAME[C], GRADE[C]
    end if
end loop
```

[3]

(e) *Award **[3 max]***
*Award **[1]** for an input statement before the loop;*
*Award **[1]** for changing the conditional statement so that it checks the* `GRADE[]` *array for the* `GRADE` *input (using the same variable)*
*Award **[1]** for outputting the name and marks of the student who has achieved the inputted grade*

Note to examiners: Accept a written explanation or an amended algorithm that corresponds to candidate's answer to part(d).

Example 1:

```
G=input()
COUNTER = 0
loop while COUNTER < 30
    if GRADE[COUNTER] = G
      then
           output(NAME[COUNTER], MARK[COUNTER])
    end if
    COUNTER = COUNTER + 1
end loop
```

[3]

13. (a) *Award **[1 max]***
photoelectric sensor;
light sensor;

[1]

(b) *Award **[6 max]***
Sensor (continuously) collecting data (related to the distance between the two vehicles);
Data is converted to digital using ADC…;
…and sent to the processor;
Processor has access to pre-set data (minimum distance required between the two cars);
Processor compares the input data against stored/pre-set data;
If the vehicle is too close, the processor sends a signal to an actuator to apply the brakes/ an output signal (for example, warning light on the dashboard/ audible warning) to the driver to apply break;
If the car has fallen back from the car in front, the processor sends a signal to an actuator to apply acceleration/ an output signal to the driver to apply acceleration;
This process is constantly looped (feedback loop);

[6]

(c) *Award **[5 max]***
*Award **[2 max]** for centralised processing*
Centralized processing would allow the different features of the car to be controlled by a single processor;
… this means that different features of the car are less likely to impact on the operation of other features, as the central processor can coordinate them;

*Award **[2 max]** for decentralised / distributed processing*
Decentralized processing allows each of the car's features to have its own processor;
… enabling each feature to function independently/more quickly;

*Award **[1 max]** for concluding statement*
Centralized processing is more complex for the manufacturer to program, but is possibly safer as the different components coordinate;
Centralised processing is more expensive to achieve as the whole system needs to be coordinated in real time;
Decentralised processing has a faster response (time due to the use of multiple smaller PLCs);
Decentralised processing is simpler/cheaper for the car manufacturer as they can use standard parts that are already programmed, but it is more difficult to get them to coordinate with each other;
Decentralised processing allows scalability, the subsystems can be easily customised;
A crash in a centralized system would result in failure of the whole system / a critical system failure so would have far greater consequences than in a decentralized system where the system may be able to partially function hence are riskier than decentralized ones;

[5]

(d) *Award **[3 max]***

Autonomous vehicles can be seen as more dangerous to pedestrians/cyclists;
…as they have to be programmed to react to avoid unpredictable movements from them;
The ethical consideration is how might the processor decide whether to avoid the cyclist to hit a pedestrian instead (trolley problem);

When autonomous vehicle control systems are perfected;
… they should be safer than human drivers;
… it may then be an ethical consideration as to whether a human who cannot drive should be the responsible person in the autonomous vehicle?;

An autonomous driving system could collect detailed information about the movements of the car;
… this information could reveal where individuals go, and raise privacy concerns;
… which could be exploited for commercial gain/ be considered as a form of surveillance (an ethical consideration);

There may be times when an accident is unavoidable (the trolley problem);
The autonomous vehicle has to make a ‘decision’ about who to hit;
In this case is the driver, the manufacturer, the software engineer or the third party accountable?;

Note to examiners: accept other reasonable answers.

[3]

14. (a) (i) *Award **[2 max]***
*Award **[1]** for a correct formula that uses the passed arguments;*
*Award **[1]** for a return statement;*

Example 1:

```
SUB_PERIMETER(NUM_SIDES, LENGTH_SIDE)
      //Pseudocode to be added
      return NUM_SIDES * LENGTH_SIDE
End SUB_PERIMETER
```

[2]

(ii) *Award **[4 max]***
*Award **[1]** for* a correct sub-program (function) defined with end statement;
*Award **[1]** for* the correct arguments passed;
*Award **[1]** for* the correct use of return;
*Award **[1]** for* the correct formula used;

Example 1:

```
SUB_AREA(PERIM, APOTHEM)
  AREA = PERIM * APOTHEM / 2
  return AREA
end SUB_AREA
```

[4]

(iii) *Award **[5 max]***
*Award **[1]** for* all three correct inputs;
*Award **[1]** for* all three inputs with appropriate prompts;
*Award **[1]** for* the correct definition of `PERIM`;
*Award **[1]** for* the correct call of perimeter sub-program, including variables/arguments passed;
*Award **[1]** for* the correct call of area sub-program, including variables/arguments passed;
*Award **[1]** for* the return/output of both values;

Example 1:

```
//Main Program
output "Number of sides?"
input NUM_SIDES
output "Length of each side?"
input LENGTH_SIDE
output "Length of apothem?"
input APOTHEM
PERIM = SUB_PERIMETER(NUM_SIDES, LENGTH_SIDE)
output "Perimeter is ", PERIM
output "Area is ", SUB_AREA(PERIM, APOTHEM)
```

[5]

(b) *Award **[4 max]***

*Award **[1]** for determining whether the shape is a polygon or a circle*
*Award **[1]** for inputting the radius (if a circle);*
*Award **[1]** for the creation of new subprogram (for area) **OR***
*changing values of input parameters for the existing one (SUB_AREA) **OR***
*the calculation using correct formula (as given in the question paper PI * RADIUS * RADIUS)*
*Award **[1]** for the creation of a new subprogram (for circumference) **OR***
*appropriately changing values of input parameters to the existing one (SUB_PERIMETER) **OR***
*the calculation using formula (as given in the question paper 2*PI * RADIUS)*
*Award **[1]** for calling both subprograms / outputting both values*
*Award **[1]** if after the alteration of the algorithm, it should still be able to calculate the area and the perimeter of the polygon as well as the area and the circumference of the circle*

Example answer 1 *(with existing subprograms, after inputting the three values: NUM_SIDES, LENGTH_SIDE, APOTHEM):*
check if the inputted value for the NUM_SIDES is 1 or 0 (then the shape is a circle);
then inputted value for LENGTH_SIDE is to be used as radius;
2*3.14 should be set as NUM_SIDES and LENGTH_SIDE as APOTHEM;
call already written sub-programs(SUB_AREA and SUB_PERIMETER) so that algorithm calculates/finds correct area of circle and circumference;
if the shape is not a circle then code (as in Part(a)) for the polygon (no other changes needed);

Example answer 2 *(with new subprograms for area and circumference created):*
construct two new sub-programs (for example, getCircleArea and getCircumference) ;
(in the main program) ask the user if the shape is a polygon or a circle;
if it is a polygon then code is as written in part a (no changes);
if it is a circle then ask the user to input the radius;
call both new subprograms to output area and circumference/ call the getCircleArea(radius) to find area and getCircumference(radius) sub-program to find circumference;

Example answer 3 *(with simple if statement and formulas given in the question paper):*
ask user to input if the shape is circle and use if statement to check this input value;
if not circle then code for the polygon (as written in part a);
otherwise (it is circle!) input radius;
calculate area;
and circumference using formulas given in the question;

Note to examiners: there may be other examples of correct answers.

[4]

15. (a) *Award **[6 max]**.*
*Award **[1]*** for correct values in VALUE column;
*Award **[1]*** for correct values of OPERAND2, OPERAND1 and NEW_VAL when VALUE is '+';
*Award **[1]*** for correct values of OPERAND2, OPERAND1 and NEW_VAL when VALUE is '-';
*Award **[1]*** for correct values of OPERAND2, OPERAND1 and NEW_VAL when VALUE is '*';
*Award **[1]*** for correct values of OPERAND2, OPERAND1 and NEW_VAL when VALUE is '/';
*Award **[1]*** correct output: 'The result is: 21';

VALUE	OPERAND2	OPERAND1	NEW_VAL	Output
5				
2				
+	2	5	7	
25				
16				
-	16	25	9	
*	9	7	63	
3				
/	3	63	21	
				The result is: 21

***Note**: The trace table may be differently presented.*

***Note**: Allow Follow Through.*

[6]

(b) *Award **[3 max]***
A stack is a last in first out (LIFO) / first in last out (FILO) data structure;
…which means data is popped off the stack in the reverse order to which it was pushed;
In the expression, it is important to evaluate some of the values in a certain order (to obtain the correct result);
For example, 25 – 16 would give the wrong value if it was evaluated as 16 – 25;

Pushing items onto a stack and then popping them off reverses the order;
To evaluate e.g. "10 2 /" we must treat it as "do the operation on the operands 10 and 2"/ i.e. we have to access the operator *before* we can apply it to the operands;
A stack achieves the reversing of the order of the operators and their operands as a group (but it also reverses the operands which must be fixed);

[3]

(c) *Award **[4 max]***
Start from the root node;
If the root is null, return immediately;
Traverse left subtree;
Traverse right subtree;
Visit root;

start from the root node (for example traverse (root)) ;
terminate traversal (and backtrack) when/if root equals null;
(before visiting the root node) traverse (and visit/process/output) each node in the left subtree (i.e., traverse(root.left));
(before visiting the root node) traverse (and visit/process/output) each node in the right subtree (i.e., traverse(root.right));
visit/output/process root (for example, output(root));

[4]

(d) *Award **[2 max]***
*Award **[1]** if the answer contains no more than one error.*

```
(5+2)*(25-16)/3
```

Note to examiners: Ignore the brackets – these represent the completely correct mathematical expression, but they are not read from the tree. Some candidates might include them because they realise that they may be needed so the expression works correctly.

[2]