Diploma Programme
Programme du diplôme
Programa del Diploma

Markscheme

November 2023

Computer science

Standard level

Paper 2

26 pages

Subject details: Computer science SL paper 2 markscheme

Mark allocation

Candidates are required to answer **all** questions in **one** Option. Total 45 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*.

Option A — Databases

1. (a) *Award **[4 max]***
Data can be structured/normalised/referential integrity;
So that the quality/consistency/integrity of data is maintained;

Databases allow data to be integrated from multiple sources;
Data can be exported in different formats/be used in other applications;

Concurrency can be ensured;
So that multiple users can make changes to the database at the same time

A database can load from storage only the data needed;
A spreadsheet will load all the data into RAM;

Data can be searched/queried by multiple fields;
Using advanced search features/SQL can be used;

Databases allow the creation of views;
To improve data privacy;

*Mark as **[2]** and **[2]***

(b) *Award **[2 max**]*
Dates will be sorted by year first, then month, then day;
Dates sorted by day or month first will be out of sequence/*accept an example 28/01/20, 28/04/19, 28/09/20;*
This is the internationally accepted way to represent dates/ISO standard;
Appropriate, if integrating data with other systems/databases;

(c) *Award **[3 max]***
Using a composite key eliminates the need for an extra field;
OrderlineID primary key creates additional storage;
Unlikely to search by OrderlineID/OrderlineID is a meaningless field;
OrderID and ItemID are natural entities since they relate to one order;
If the client orders the same meal another time the OrderID will be different;

(d) *Award **[2 max]***
The program may not know which UnitPrice is being referenced;
This would cause an error message to be display/the query would not execute;
It will be necessary to state the table name with the UnitPrice;

(e) *Award **[4 max]***
*Award **[1]** for selecting FirstName, Surname and OrderDate*
*Award **[1]** for identifying the tables Customer and Order*
*Award **[1]** for correctly JOIN condition (PK – FK comparison)*
*Award **[1]** for correct Firstname and Surname filtering condition*

SELECT Customer.Firstname, Customer.Surname, Order.OrdDate
FROM Customer INNER JOIN Order ON Customer.CustomerID = Order.CustomerID)
WHERE Customer.Firstname = 'Akemi' AND Customer.Surname = 'Watene';

SELECT Customer.Firstname, Customer.Surname, Order.OrdDate
FROM Customer, Order
WHERE Customer.CustomerID = Order.CustomerID) AND
Customer.Firstname = 'Akemi' AND Customer.Surname = 'Watene';

Accept equivalent in Structured English

2. (a) *Award **[3 max]***
The user may type in the wrong amount;
LinePrice should be a derived field/calculated field//=[Quantiy] * [UnitPrice];
It is not necessary to store this data/wastes storage space;
The relation/table does not conform to 3NF;

(b) *Award **[1 max]***
A logical unit of work executed in full or not at all;
A change in the database state;

(c) (i) *Award **[1 max]***
Range check/accept example, >0 and <=12;
Presence check/a blank field may cause a calculation error;
Type check/must enter a number/ float/integer;

(c) (ii) *Award **[2 max]***
Proofreading/Comparing the payrate to hardcopy version;
Lookup payrate in another table;
Based on the selected job/jobID;
double data entry/Two-pass verification;

(d) *Award **[4 max]***
Only allow waiters to view work hours not approve them during a shift;
Writing to the table could only be open at the end of the shift/set times;
The database could be set to lock the record/table while it is being edited;
So the waiter could not approve the hours while they are edited;
The database could be set to block viewing while the table is being edited;
A message could be conveyed to the waiter that the data is being updated;
Two-phase locking can be applied to guarantee full isolation of the update;
Multi-version concurrency control (MVCC) could be used to provide the waiter the prior, unmodified version of data that is being modified;

(e) *Award **[4 max]***
Instant recovery/Recovery in place;
A snapshot is created so the backup remains in a correct state;
All user write operations are redirected to that snapshot;
Users then work off the backup virtual machine (VM);
The recovery process begins in the background/Users are unaware of the recovery;
When the recovery is complete the workload is directed back to the original VM;

Crash recovery;
If a crash occurs mid-transaction;
The system follows a protocol/algorithm to recover lost data;
A check is made of the states of the transactions that were being executed;
If the transaction can be completed it should do so;
If not the transactions that were interrupted during the crash are rolled back;
No transaction can be allowed to leave in an inconsistent state;

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

Accept variations of this diagram. Owner, Pet, and Passport correctly positioned.

*Award **[1]** for 1 to m;*
*Award **[1]** for 1 to 1;*

(b) *Award **[1 max]***
A secondary key is a non-identifying field used to find row in a table;

(c) *Award **[8 max]***
*Award **[3]** for all 4 entities/**[2]** if there are only 3 entities/**[1]** if there are 2 correct entities.*
*Award **[1]** for correct primary keys*
*Award **[1]** for correct foreign keys*
*Award **[1]** for all correct fields in* VACCINE
*Award **[1]** for all correct fields in* PETVACCINATION
*Award **[1]** for all correct fields in* PET
*Award **[1]** for all correct fields in* PASSPORT

PASSPORT(TransponderCode, VetID(fk), DateOfIssue)
PET (TransponderCode, PetName, Species, Breed, Sex, DOB)
PETVACCINATION (VaccinationID, vaccineID, VetID(fk), TransponderCode(fk), VaccineDate, ValidFrom, ValidTo)
VACCINE (vaccineID, VaccineName, Manufacturer, BatchNumber)

Accept PassportID instead of TransponderCode in the Passport table.
Accept OwnerID in PASSPORT.

(d) *Award **[4 max]***
Data should only be used for the reason that it was collected;
Data should not be transferred to another country unless it has equal or better data protection law;
Data should not be shared without permission;
Data should be registered with the data protection governing body;
Data should be used in accordance with the law;
Data should be adequate and relevant;
Data should not be kept longer than necessary;
Data should be accurate and up-to-date;
The pet owner has the right to see the data stored about them;

Option B – Modelling and simulation

4. (a) *Award **[1 max]***
0.5 or 50% or 50:50;

Accept any form that indicates and even chance.

(b) *Award **[1 max]***

	A	B	C
1	1	2	3
2	1	3	2
3	2	1	3
4	2	3	1
5	3	1	2
6	3	2	1

Award the mark if the sequence differs but the content is correct.

(c) *Award **[2 max]***
*Award **[1]** for if condition i.e. B1>A1, B1*
*Award **[1]** for else condition i.e. C1*

```
=IF(B1>A1,B1,C1)
```

(d) (i) *Award **[2 max]***
*Award **[1]** for using a COUNTIF*
*Award **[1]** for using referential integrity for the range*

```
=COUNTIF(D$1:D$6,E1)
```

Accept other values to E1 e.g. F1 or 1, 2, 3

Alternative solution:
```
=SUM(IF($D$1=1,1,0), IF($D$2=1,1,0), ... IF($D$6=1,1,0))
```

(ii) *Award **[3 max]***
*Award **[1]** each for 16.7%, 33.3%, and 50% or equivalent*

	A	B	C	D	E	F	G
1	1	2	3	2	1	1	16.7%
2	1	3	2	3	2	2	33.3%
3	2	1	3	3	3	3	50%
4	2	3	1	3			
5	3	1	2	2			
6	3	2	1	1			

Note that the spreadsheet is not necessary. Only the values in G are needed.

(e) *Award **[1 max]***
*Award **[1]** for second if statement*

```
=IF(B1>A1,B1,IF(C1>A1,C1,D1));
```

(f) *Award **[6 max]***
*Award **[3 max]** for advantages*
*Award **[3 max]** for disadvantages*
*Award **[1 max]** for conclusion*

Advantages
Spreadsheet reduces the likelihood of calculation mistakes;
You can clearly see which cat is chosen;
You could create a graph to visualise the results;
Formulas can be changed easily e.g. if you change your decision making;
(e.g. always reject the first two cat, accept the third cat if higher than the first two, else choose the fourth cat;)

Disadvantages
There is no easy way to calculate ordered permutations/permutations need to be entered manually;
May lead to errors if the formula needs to be changed e.g. If statements changed to reflect decision-making;
(e.g. always reject the first two cat, accept the third cat if higher than the first two, else choose the fourth cat;)

Conclusion
Works well when there are only 3 or 4 cats;
For more than 4 cats it is unrealistic to model using a spreadsheet;

5. (a) *Award **[2 max]***
*Award **[1]** for initialising TOTAL and returning TOTAL*
*Award **[1]** for looping from 1 to N times*
*Award **[1]** for TOTAL = TOTAL * X*

Example 1:
```
function FACTORIAL(N):
    TOTAL = 1
    loop X from 1 to N
        TOTAL = TOTAL * X
    end loop
    return TOTAL
end function
```

*Award **[1]** for a terminating condition* `N == 1`
*Award **[1]** for calling* `FACTORIAL(N-1)` *to loop recursively*
*Award **[1]** for multiplying by* `N` *to return the total value*

Example 2:
```
function FACTORIAL(N):
    if N == 0 or N == 1 then       // does not need to include N==0
        return(1)
    else
        return FACTORIAL(N-1) * N
    end if
end function
```

Accept alternative solutions that calculate factorial.

(b) *Award **[2 max]***
*Award **[1]** for using a variable* `HIGHEST` *that is suitably initialised*
*Award **[1]** for correctly finding the highest value*

Example 1:
```
function FINDHIGH(ARR):
    HIGHEST = -1
    loop COUNT from 0 to 2
        if ARR[COUNT] > HIGHEST then
            HIGHEST = ARR[COUNT]
        end if
    end loop
    return HIGHEST
end function
```

Example 2:

```
function FINDHIGH(ARR):
    HIGHEST = ARR[0]
    if ARR[1] > HIGHEST then
       HIGHEST = ARR[1]
    end if
    if ARR[2] > HIGHEST then
       HIGHEST = ARR[2]
    end if
    return HIGHEST
end function
```

(c) *Award **[6 max]***

*Award **[1]** for looping from 0 to (*`FACTORIAL(8) – 1`*)/looping through all permutations*
*Award **[1]** for correctly using* `FINDHIGH` *and* `NEXTPERMUTATION`
*Award **[1]** for loop from index 3 to 7*
*Award **[1]** using a boolean variable (*`FOUND`*) to exit the loop*
*Award **[1]** testing if* `ARR[POS] > HIGHEST`
*Award **[1]** assigning* `CHOICE = ARR[POS]`
*Award **[1]** choosing the last cat if no better cat was found (*`CHOICE = ARR[7]`*)*
*Award **[1]** for incrementing* `total` *and correctly calculating the percentage*

Example 1:

```
START = 3
ARRLEN = 8
TOTAL = 0
loop ROW from 0 to (FACTORIAL(ARRLEN)-1)
    HIGHEST = FINDHIGH(ARR)
    FOUND = False
    POS = START
    loop while FOUND = False and POS < ARRLEN
        if ARR[POS] > HIGHEST then
            CHOICE = ARR[POS]
            FOUND = True
        end if
        POS = POS + 1
    end loop
    if FOUND = False then
        CHOICE = ARR[ARRLEN-1]
    end if
    if CHOICE = X then
        TOTAL = TOTAL + 1
    end if
    if ROW < FACTORIAL(ARRLEN)-1 then
        ARR = NEXTPERMUTATION(ARR)          // accept without test
    end if
end loop
PERCENT = TOTAL/FACTORIAL(ARRLEN)*100
output(PERCENT)
```

Example 2 (simplified):

```
TOTAL = 0
loop ROW from 0 to (FACTORIAL(8)-1)
    HIGHEST = FINDHIGH(ARR)
    FOUND = False
    POS = 3
    loop while FOUND = False and POS < 8
        if ARR[POS] > HIGHEST then
            CHOICE = ARR[POS]
            FOUND = True
        end if
        POS = POS + 1
    end loop
    if FOUND = False then
        CHOICE = ARR[7]
    end if
    if CHOICE = X  then
        TOTAL = TOTAL + 1
    end if
    ARR = NEXTPERMUTATION(ARR)
end loop
PERCENT = TOTAL/FACTORIAL(ARRLEN)*100
output(PERCENT)
```

(d) *Award **[2 max]***
Change the number of cats to reject at the beginning;
From 3 cats to 2 cats and/or from 3 cats to 4 cats;
Compare the probabilities with the first solution;

(e) *Award **[4 max]***
The program with 20 values will not run well on a standard laptop/This is a NP-type problem;
Fast CPU required;
Maths co-processing/ASIC may be needed;
Program could be adapted to run with parallel processors;
Program adapted to run with GPU;

Allow 1 mark for a generic statement such as:
Needs a lot of RAM;

Do not award marks for 'using a GPU' since this won't be useful on its own.

6. (a) *Award **[1 max]***
A technique for generating a flat (non-3D) image/animation representation of an object/process/set of results;

(b) (i) *Award **[4 max]***
Maps use 2D coordinates/ latitude and longitude;
The GPS location of the emergency can be accurately added to a map;

To find the emergency location it is not necessary to have altitude;
Because in a city, emergencies are not likely to occur underground or at altitude;

A 2D map is easier to read than a 3D map;
Because it reduces the complexity/the 3D map has additional unnecessary data;

*Award **[2]** and **[2]***

(b) (ii) *Award **[2 max]***
2D visualisation is less complicated than a 3D visualisation;
So there is not a great deal of memory needed;

Accept suitable answers that do not compare to 3D visualisation.

(c) *Award **[6 max]***
Highly realistic rendering is not required/only functional 3D visualization is needed;
So photorealistic rendering (e.g. ray tracing) should not be used;

Emergency rescue only need a 3D walkthrough to know where to go;
They don't need high levels of detail just the route;

3D rendering requires a great deal of processing power;
Emergency services usually have access to fast computers;

3D photorealistic rendered animations are large files;
Transmitting them will take longer/may buffer;

3D wireframe or scanline will be smaller in size;
So will be received without delay/won't buffer;

There is little available time;
So wireframe or scanline processing would be a better choice;

Photorealistic rendering (e.g. ray tracing) can take days;
The people will be burned before the rescue team can get there;

Accept answers that only address components of the visualization, not the process of.

*Award **[2]** and **[2]** and **[2]***

Option C – Web science

7. (a) *Award **[2 max]***
Banner;
Header;
Meta-tags;
Body of the page with content;
Page footer containing contact details, copyright statement, site administrator;
Embedded object;
Buttons;
Menu;
Sound;
Applet;
Flash content;
Script;
Link;
Image;
Form;
CSS;

Accept any other reasonable response.

(b) *Award **[2 max]***
Stateless Protocol;
Connectionless Protocol;
Text-Based Messaging;
Unencrypted;
Client-Server Architecture;
Request Methods (e.g., GET, POST);
Uses Status Codes;
Uses TCP/IP Protocol;
Uses Port 80 (default);
Support for MIME types;

(c) *Award **[4 max]***
Receives $dogname parameter in HTTP POST;
Attempts to establish a database connection to dogDB;
Constructs an SQL query based on the received $dogname;
Executes the SQL query;
Checks if there are rows in the query result;
Iterates through the query results;
Displays information about each dog;
Closes the database connection;
Displays "not found" if no results were found in the query;

(d) *Award **[2 max]***
Data Security: Safeguards sensitive data on the server like database username and password;
Data Validation: Ensures data is clean and sanitised;
Database Interaction: Efficiently manages data through direct database access;
Controlled Access: Enforces user authentication and access control;
Server Resources: Relieves client devices by using server power;
Scalability: Handles heavy workloads and scales seamlessly;
Bandwidth: does not transmit large query results across the internet, saving bandwidth;

(e) *Award **[3 max]***
Lossy compression will reduce the size of the image file being downloaded;
This means that the time required to download the file will be reduced/faster response;
the decompressed file may lose some quality but not significant enough to make the file unusable;
Save disk space on the server and memory for the client;
It helps in reducing the cost (associated with data transfer over networks and storage on web servers);

(f) *Award **[5 max]***

Advantages of using a template *[3 max]*
Provides an easy to understand/consistent user interface;
Can be developed in modules such as headers, footers, sidebars etc;
Quicker/cheaper to develop as the code doesn't need to be rewritten;
Code quality is likely to be higher due to reuse in many places;
Perceived quality might be superior as it has been developed with more care/time;

Disadvantages of using a template *[3 max]*
May be constrictive and inhibit creativity;
Lack of flexibility as the template may not be editable;
The code may not be developed according to best practices;
Potential for code bloat if the template includes more features than needed;
Initial cost of template;
Subsequent cost for customization of template;
Site may not be distinguishable from other sites;

8. (a) *Award **[2 max]***
Ask for tour Ratings;
Incorporate Quizzes and Stories;
Encourage Customer Feedback;
Post Customer-Generated Content/ images;
Incentivize with Gamification and Rewards;
Add Booking Forms: Allow users to book tours directly;
Incorporate Live Chat: Provide real-time customer support;
Include Interactive Maps: Show tour locations and details;
Implement Social Media Links: Connect to social profiles;
Offer Virtual Tours: Provide 360-degree views;
Create User Accounts: Enable personalized experiences;
Add Newsletter Signup: Keep users informed;

Accept other reasonable responses.

(b) *Award **[2 max]***
Use content/code/images etc. generated themselves rather than being taken from external sources;
If materials are taken from external sources, seek permission from the owner to use their content;
If content used is under Creative Commons License, attention must be paid to specific permissions like use for non-commercial purpose, credit original creator, quid pro quo;

(c) *Award **[2 max]***
Time taken to load the site;
Time spent on the site by a visitor;
number of visitors;
Quality and quantity of returns/incoming links/back links;
Trustworthiness of linking domain;
quality of the content;
friendliness to different devices;
Relevancy of content on the pages that are linked;
Click through rat;
Conversion rate (ie buying tours);
Proprietary algorithm (eg PageRank, or HITS);

(d) *Award **[2 max]***
Use of relevant keywords;
Increasing the dwell time of users on the website/build navigation and functionality that enhances user experience;
Publish relevant and quality content;
High-Quality Backlinks;
Mobile Optimization;
Correct use of metatags;
User-Generated Content;
Secure and Well-Hosted Website;
Site Structure;
Guest blogging;
link baiting;

Accept other reasonable responses.

(e) *Award **[4 max]***
Sustainability, white hat SEO ensures long-term ranking success by adhering to search engine guidelines;
Credibility and Trust, ethical practices build trust with users and search engines, leading to higher rankings;
Avoidance of Penalties, white hat SEO avoids search engine penalties associated with manipulative tactics;
Better User Experience, focusing on user experience improves retention and search rankings;
Positive Reputation, ethical SEO enhances online reputation and can lead to organic promotion through social shares;
Adherence to Web Standards, white hat methods comply with web standards, aiding accessibility and indexation;
Future-proofing, white hat SEO is less likely to be negatively impacted by future search engine algorithm updates;

(f) *Award **[3 max]***
Technical Assumptions:
Efficiency and cost targets;
Including goals and constraints;
Which may affect indexing depth and query responsiveness;

Assumptions about webpage significance;
And the algorithm's reliance on backlinks as indicators of page importance
Which can influence content surfaced and overlook less-linked yet relevant content;

Static assumptions about web content;
That can limit adaptability to new information;
And user behavior changes;

User-Centric Assumptions:
Search algorithms may have inherent biases;
Based on developer assumptions
That affect result relevance and diversity;

Varying design goals among search engines;
That lead to different effectiveness in result accuracy;
And speed;

Developer-defined metrics for desirable result types and user experience;
Design assumptions that might prioritize certain content types;
Potentially skewing result relevance for specific user needs;

9. (a) *Award **[2 max]***
A server shares/hosts services/resources with a number of clients;
A client is a single computer that is part of a network and requests functions/services from a server;

(b) *Award **[2 max]***
Identification is claiming/presenting an identity /evidence such as entering your username to confirm who or what something is;
Authentication is a process of establishing/validating someone's identity using the given credentials such as username and password to gain access to a service/resource;

(c) *Award **[3 max]***
Cost saving;
Cloud storage results in reduction of storage and operative costs;
As organisations can cut on expensive hardware requirement needed in a client-server network architecture;
Also internal power and maintenance costs are reduced;

Accessibility;
Cloud storage allows an organisation to access the stored files from anywhere increasing the accessibility and usability;
Users can easily drop files to save in the cloud storage;
Files can be shared with team members and clients with multiple different devices in an easy way;

Scalability**;**
The organization can pay for as much they use;
They can easily scale up/down the storage when they need to do so;
Depending on changing demands for storage;

Disaster recovery/automatic backup;
The organization can rely on the automated backup process offered by the cloud storage provider;
This will lead to better recovery in case of natural disasters;
This doesn’t require too much involvement of the organization;

(d) *Award **[6 max]***

Answers may include:

Ownership Concentration:

Cloud storage has expanded geographically;
But is predominantly controlled by major providers, which challenges the extent of decentralization;
The perceived spread of data centres is overshadowed by the consolidation of ownership and control,
Which can lead to increased market share for digital oligarchs;
The trend of businesses transitioning from local or distributed storage to centralized cloud services may decrease the diversity of storage solutions;

Challenges to Decentralization:

Centralized data in the cloud presents significant security risks, which can impede decentralization efforts due to the attractive target it presents for cyber threats;
A lack of adopted standards for interoperability across different platforms hinders the possibility of a decentralized web ecosystem;
The economic advantages that large-scale providers have due to their size can dissuade the shift towards a more decentralized infrastructure;

Impact on Innovation:

The control exercised by a few large cloud providers can create high barriers to entry, potentially stifling innovation in the sector;

Data Sovereignty:

Data storage by centralized cloud services in various international locations raises issues of data sovereignty;
Affected by differing local laws and practices;

Network Effects:

The strong network effects associated with the services of large providers enhance their value as more users join;
Reinforcing the centralization of the web;

Conclusion:

Decentralization may remain limited until peer-to-peer and blockchain technologies become more mainstream and offer a viable alternative to centralized cloud storage;
Environmental concerns over the distributed computing model may challenge its adoption, with its benefits needing to outweigh its carbon footprint;
Government regulations could significantly impact the trajectory towards decentralization, either by imposing barriers or by encouraging diversified storage practices;
A cultural shift towards prioritizing data privacy and control may catalyze movements towards a more decentralized web structure.

Option D – Object-oriented programming

10. (a) *Award **[1 max]***
Variable belongs to the class not to the instantiation;
All instances share the same variable;
A change to one object state of that variable affects all of them;
Variable can be accessed without instantiating an object of the WaterMonster class;

Note: do not award: It cannot be changed/is a constant.

(b) (i) *Award **[2 max]***
Data hiding / increased security;
This prevents accidental changes to these variables from outside the class instance;
Since modifications require getter and setter methods;

(ii) *Award **[2 max]***
accessor;
mutator;

Note: allow get(ter) and set(ter) methods, or a correct java example.

(c) *Award **[2 max]***
A reference to the superclass; (but do not accept reference to inheritance / `extends`)
Calls the constructor (of Monster);

Note: both marks can be gained in one statement, eg.
"Super()" calls the constructor of the superclass (Monster).

(d) *Award **[2 max]***
Security;
Affordability;
Transparency;
Perpetuity;
Interoperability;
Flexibility;
Localization;

Note: Accept other reasonable answers.

(e) *Award **[2 max]***
Award [1] for an appropriately typed and named variable (left of "=")
Award [1] for `new FireMonster` and correct parameters for constructor

```
eg. FireMonster flamey = new FireMonster("Flamey", 40);
```

11. (a) *Award **[4 max]***
*Award **[1]** for correctly using both methods* `rollDice()` *and* `output()`
*Award **[1]** for correctly re-calculating the attacked monster's health (ie subtracting the damage)*
*Award **[1]** for outputting any correct message if health > 0*
*Award **[1]** for outputting the correct message if health <= 0 ("must leave the game")*
*Award **[1]** for setting health to 0, if it is <0*

Example:

```
public void attack(Monster defender)
{
     int damage =  rollDice();
     defender.health = defender.health - damage;
     if (defender.health > 0)
     {
         output(defender.name+" has sustained "+damage+" damage
                and now has "+defender.health+" health left");
      }
      else
      {
          output(defender.name+" must leave the game");
         defender.health = 0;
       }
}
```

Example without int damage:

```
public void attack(Monster defender)
{
    int oldHealth = defender.health;
    defender.health = defender.health - rollDice();
    if (defender.health > 0)
    {
        output(defender.name+" damage: "+(oldHealth-defender.health)+
               " health "+defender.health);
    }
    else
    {
        defender.health = 0;
        output(defender.name+" must leave the game");
    }
}
```

(b) *Award **[4 max]***
Class is a code definition / template / blueprint;
Whereas Object is the actual / customizable instance variable;

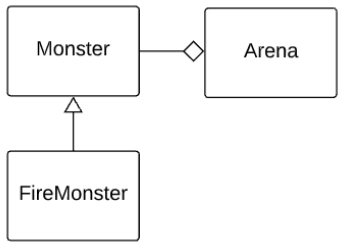
Memory use of a class is only for static elements;
whereas an object uses memory for non-static parts;

There is only 1 class;
however there could (potentially) be multiple objects / instances of that class;

Mark as **[2]** and **[2]**

(c) *Award **[2 max]***
*Award **[1]** for connector between Arena and Monster indicating aggregation (has a)*
*Award **[1]** for connector between FireMonster and Monster indicating inheritance (is a)*

Example:



(d) *Award **[5 max]***
*Award **[1]** for checking if* `monsterCount<10 / monsterCount < monsters.length`*;*
*Award **[1]** for checking if that instance already exists (ie. using* `monsters[i]==M` *or* `.equals(M)` *or* `monsters[i].name==M.name)` *or* `.equals(M.name)`*;*
*Award **[1]** for adding monster in correct position and increasing monsterCount;*
*Award **[1]** for correctly located error message if* `Arena` *is full;*
*Award **[1]** for correctly located error message if monster already exists;*

Note: Candidates are not expected to cater for a situation in which the Arena is full AND the monster is already there. They may give either or both error messages if this is the case.

Example:

```
public void addMonster(Monster M)
{
     if (monsterCount < monsters.length)
     {
           boolean alreadyEntered = false;
           for (int i = 0; i < monsterCount; i++)
           {
                 if (monsters[i]==M)
                 {
                     alreadyEntered = true;
                  }
           }
           if (!alreadyEntered)
           {
               monsters[monsterCount] = M;
               monsterCount++;
           }
           else
           {
                output(M.name + " has already entered");
           }
     }
     else
     {
            output("Sorry the Arena is full");
     }
}
```

(e) *Award **[4 max].***
Unnecessarily complex for small problems;
Some applications do not require OOP features and could be solved in a simpler way;

Larger program size;
More code is produced as classes, methods etc. need to be defined;

Might run slower;
This might be due to the extra time involved in creating instances of objects (ie. many monsters may be “created” and “destroyed” during the game);

Not good for modelling situations in which only single objects/instances are needed;
A small application may not require all those features;

OOP training is required by developers;
The programmers would be trained beforehand / this could be expensive;

*Mark as **[2]** and **[2]***

12. (a) *Award **[2 max]***
A method with the same name but different parameter types is created;
Depending on the arguments/parameters, one of the methods is executed;

A method with the same name and parameters is created / inherited in the subclass;
and used instead of the method in the superclass;

(b) *Award **[6 max]***
*Award **[1]** mark for attempt to find the index of* `Monster M`
*Award **[1]** mark for correct loop that finds the index of* `Monster M` *(using* == *or* .equals()*)*
*Award **[1]** mark for attempting a loop to shift monsters*
*Award **[1]** mark for correctly shifting monsters from the index onwards (either until the end or until the first* `null` *value)*
*Award **[1]** mark for correctly adding the extra* `null` *value to the array (at the end)*
*Award **[1]** for decrementing* `monsterCount`

Example 1:

```
public void removeMonster(Monster M)
{
     for (int i = 0; i < monsters.length; i++)
     {
         if (monsters[i]==M)
         {
             for (int move = i; move < monsterCount – 1; move++)
             {
                 monsters[move] = monsters[move + 1];
             }
             monsters[monsterCount – 1] = null;
             monsterCount = monsterCount – 1;
             break;
         }
     }
}
```

Example 2:

```
public void removeMonster(Monster M)
 {
     // find the monster
     boolean found = false;
     int i = 0;
     while (!found && i<monsterCount)
     {
         found = (monsters[i] == M);
         i = i + 1;
     }
     // close up the rest of the array
     for (int j=i; j<10; j++)
         monsters[j-1] = monsters[j];
     monsterCount = monsterCount - 1;
 }
```

Example 3:

```
public void removeMonster(Monster M)
{
   int i=0;
   while (monsters[i]!=M)
   {
       i++;
   }
   for (int j = i; j < monsters.length-1; j++)
   {
       monsters[j] = monsters[j+1];
   }
   monsters[monsters.length-1] = null;
   monsterCount = monsterCount-1;
}
```

Example 4:

```
public void removeMonster(Monster M)
{
   int i=0;
   while (monsters[i]!=M)
   {
       i++;
   }
   for (int j = i; j < monsterCount-1; j++)
   {
       monsters[j] = monsters[j+1];
   }
   monsters[monster.length-1] = null;
   monsterCount = monsterCount-1;
}
```

(c) (i) *Award **[2 max]***
It is the mechanism/process in which a new class(derived/child) is based on the features of / is allowed to copy of/ extends the existing class (base/parent);
A parent object holds common data and actions for a group of related child objects;
Variables and methods defined in superclass are available in subclass;
Private variables and methods will not be accessible/inherited by the subclass;
Protected variables and methods also available to subclass if in the same package;

(ii) *Award **[5 max]***

(Inheritance)
*Award **[1]** mark - Each Specific card extends its monster type class (eg. Splashy extends WaterMonster)*
*Award **[1]** mark - Therefore specific cards will inherit the public behaviour (abilities) from the Monster class and their MonsterType class*
*Award **[1]** mark - Ability1 must be implemented in Monster class and will be inherited by all*
*Award **[1]** mark - Abillity2 and Ability3 should be defined in the Monster class (but not implemented*

(Polymorphism)
*Award **[1]** mark - Ability2 should be overridden in Monster type class (eg. WaterType)*
*Award **[1]** mark - Ability3 should be overridden in class of specific monster (eg. Squishy)*

(both)
*Award **[1]** mark – A specific behaviour is defined/overridden in the subclasses*
*Award **[1]** mark – The default behaviour is defined in the superclass*