M23/4/COMSC/SP2/ENG/TZ0/XX/M

Markscheme

May 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) (i) Award ***[1 max]***
One – many;
One – one;

(ii) Award ***[1 max]***
CustID;
RentalNum;
CarID;

(iii) Award ***[1 max]***
Rental;

(b) Award ***[1 max]***
EF475;

(c) Award ***[4 max]***
Length Check;
Character count is 7;

Input mask/Format check;
LLL0000/3 STRING/TEXT following by 4 INTEGER/NUMBER;

Data Type Check;
STRING data type/only alphanumeric and no special characters;

Uniqueness Check
Data is not duplicated

Presence check;
Certain attributes must be present/ Data exists;

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

(d) *Award **[4 max]***
*Award **[1]** for selecting the SName from Customer;*
*Award **[1]** for correct tables(both) identified;*
*Award **[1]** for making the CustID comparison;*
*Award **[1]** for writing the correct DateOut and DateRet;*

```
SELECT SName FROM CUSTOMER
INNER JOIN RENTAL ON “CUSTOMER.CustID” =
“RENTAL.CustID”
WHERE DateOut = “22/01/2020” AND
DateRet = “26/01/2020”;
```

Note: candidates who have used 2 separate queries can be awarded up to 3 marks
Note: the mark for using both tables cannot be awarded

(e) Award ***[3 max]***
Used to save frequently used, complex queries;
Though a view doesn't store data, we refer to a views as "virtual tables,"
you can query a view like you can a table;
A view can combine data from two or more tables, using joins, and also just contain a subset of information;
Views don't store data directly/draw data from tables;
This makes them convenient to abstract, or hide, complicated queries;
Views automatically update (dynamic) as the data in tables change;
(Static/materialised) views capture the query at that moment in time/to allow for faster retrieval with large datasets;
Views improve security because you can give access to a view without access to the underlying database;
Some types of views allow you to insert / update / delete multiple tables within the one view;

Note: must be related to views of queries not queries themselves

(f) Award ***[3 max]***
A DDL is used to specify the schema of a database / defines parameters and data
A DDL allows you to define the tables, fields, datatypes e.g CREATE TABLE;
A DDL allows you to specify keys/create relationships;
A DDL allows you to modify schema (e.g. DROP, ALTER);

(g) Award ***[3 max]***
Data about the rents given out will be accurate at any point in time;
data written to the database must be valid according to all defined rules, including constraints, cascades, triggers;
Consistency allows user to trust the data for decision making;
Consistency is need for transactional operations to be carried out correctly/ACID;
Consistency allows data to be transferred (to other systems) with confidence;
Consistency allows data to be analysed using algorithmic tools/multiple formats cannot be analysed accurately;
Consistency is important or it leads to integrity issues (e.g. foreign keys unlinked to primary keys);
Without consistency it leads to confusion and misinterpretation of the data/ need of accuracy;
Compliance with the car industry may be a legal requirement;

2. (a) Award ***[1 max]***

Amount of acceptable data loss;
Time to full recovery/Recovery Time Objective(RTO);
Cost (Instant recovery requires failover systems);
Resources/Architecture; (volume of data) // size of database not accepted;
Legal regulations/sensitivity of data;

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

Backup and Restore/The DBA restores the database from backup;
Using BACKUP and RESTORE SQL commands;
After making periodic backups of the database (may be full or incremental);

Log Recovery/ Point in time recovery/rollback;
Backup is recorded in the log files and restored at that point;
Transaction after that point are added back in;
If the failure was due to an error in a transaction, the DBA can also "undo" the transaction to revert the database to a consistent

Instant Recovery/database mirroring/recovery in place;
Redirects user workloads to the backup server/backup database is brought online;
When recovery is completd the original system is used with the interim transactions added;

Deferred update (NO-UNDO/REDO or write-ahead logging);
The system consults the transaction log and any transactions that were not fully written to the database (deferred) are redone;

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

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

Access control levels;
same as in a school, admistrators can see everything, teachers can see some, students can see very little;
Any example to explain the access control can be accepted;

Maintaining data anonymity;
for example the personally identifiable information (PII) that could potentially identify a person in the database is located in a separate table;
and the person is given a unique identifier like PER0001 so the anonymity of the person can be ensured;

Note: password alone not accepted, encryption not accepted

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

3. (a) Award ***[2 max]***
1NF requires atomic values;
CourseNum, CourseName all have multiple values in the cells in their columns;

Repeating data;
For example, Student 2907 has three courses so is repeated each time;

(b) Award ***[8 max]***
Award ***[1]*** *for each* correct table with the correct attributes (x4)
Award ***[1]*** *for each correct primary key (x4)*

3NF
STUDENT (StudentID, FName, SName, MajorID)
MAJOR (MajorID, Major)
COURSE (CourseNum, CourseName)
ENROLLMENT (StudentID, CourseNum) or ENROLLMENT (EnrollmentID, StudentID, CourseNum)

3NF
STUDENT (StudentID, FName, SName, MajorID)
MAJOR (MajorID, Major)
REGISTER (StudentID, CourseNum)
COURSE (CourseNum, CourseName)

3NF
STUDENT (StudentID, FName, SName)
ENROLL (StudentID, CourseNum)
MAJOR (MajorID, Major)
COURSE (CourseNum, CourseName, MajorID)

Example using 2NF
STUDENT (StudentID, FName, SName, MajorID, Major)
COURSE (CourseNum, CourseName)
ENROLLMENT (StudentID, CourseNum) or ENROLLMENT (EnrollmentID, StudentID, CourseNum)

Note: this normalisation to 2NF would be awarded 5 marks as follows:
*Award **[1]** for PK in STUDENT*
*Award **[1]** for PK in COURSE*
*Award **[1]** for correct attributes in COURSE*
*Award **[1]** for PK in ENROLLMENT*
*Award **[1]** for attributes in ENROLLMENT.*

(c) Award ***[5 max]***
Data must be modelled knowing the purpose otherwise the data will not do what is required;
Identify the entities/tables;
If incorrect they will not support the purpose of the DB;
The attributes of the tables should be necessary and sufficient for purpose;
Otherwise may get inefficiencies;
The keys for accessing the tables need to be identified;
Otherwise the user cannot access the data;
Identify the relationships among tables;
Otherwise the user cannot perform complex queries across several tables;
Identify relationships and normalization that reduces data duplication;

Option B – Modelling and simulation

4. (a) *Award* ***[1 max]***

A computer program designed to simulate a real-life scenario / what might or what did happen in a situation;

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

Units – Integer (allow number or real, etc);
MONTH – String / text / alphanumeric
UnitCost – Real;
DailyCharge – Real;
Tax – Real;

Award ***[1 max]*** *for 3 variables without datatype*
Allow other suitable responses.

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

Inputs – units used and days in month;
  // do not allow a reference to a 2D array for the first marking point
Calculation of bill;
Calculation of units used per day;
Calculation of tax;
  // allow references to a 2D array for calculations
Outputs – bill and units used per day;
Appropriate messages in inputs and outputs;

Example algorithm

```
UNITCOST = 0.2
DAILYCHARGE = 0.75
TAX = 0.1
input "units used this month? ", UNITMONTH
input "Days in month? ", DAYSINMONTH
BILL = (UNITCOST * UNITMONTH + DAILYCHARGE * DAYSINMONTH)
TOTALBILL = BILL + BILL * TAX
UNITDAY = UNITMONTH / DAYSINMONTH
output "The bill for this month is ", TOTALBILL
output "The number of units used per day were ", UNITDAY
```

(d) *Award* ***[2 max]***

The model only shows how much power has been used in a month/per day;
Insufficient information on where money/electricity could be saved;

(e) *Award* ***[2 max]***

Spreadsheets;
Code;

// accept for ***[1]*** *any form of code, eg. linear search*

(f) *Award [8 max]*
*Award **[1]** for initialization of MIN/MAX or arrays MONTHS/USAGE*
*Award **[1]** for declaring HIGHMONTH/LOWMONTH or TEMP/TEMPMONTH*
*Award **[1]** for appropriate loop / double loop*
*Award **[1]** for use of parallel arrays // Do not accept a 2D array*
*Award **[1]** for finding and recording data for highest month / sorting the usage;*
*Award **[1]** for finding and recording data for lowest month / sorting the month;*
*Award **[1]** for Outputting *month* for highest usage;*
*Award **[1]** for outputting *month* for lowest usage;*
// allow for finding months with highest and lowest bill

Example algorithm

```
MAX = 0
MIN = 9999999    (or similar)
COUNTER = 0
HIGHMONTH = ""
LOWMONTH = ""
loop while COUNTER < 12
    input "units used this month? ", ELECMONTH[COUNTER]
    input "name of month? ", MONTH[COUNTER]
      if ELECMONTH[COUNTER] > MAX then
        MAX = ELECMONTH[COUNTER]
        HIGHMONTH = MONTH[COUNTER]
      end if
      if ELECMONTH[COUNTER] < MIN then
        MIN = ELECMONTH[COUNTER]
        LOWMONTH = MONTH[COUNTER]
      end if
    COUNTER = COUNTER + 1
end loop
output "The highest usage was in ", HIGHMONTH
output "The lowest usage was in ", LOWMONTH
```

Alternative solution which sorts the data in two parallel arrays and outputs the lowest and the highest that way.

```
MONTHS = ["Jan", … , "Dec"]
USAGE = [5800, … , 6280]
TEMP = 0
TEMPMONTH = ""
I = 0
loop while I < 11                  // allow I < 12
    J = 0
    loop while J < 11
        if USAGE[J]>USAGE[J+1] then
            TEMP = USAGE[J]
            USAGE[J] = USAGE[J+1]
            USAGE[J+1] = TEMP
            TEMPMONTH = MONTH[J]
            MONTH[J] = MONTH[J+1]
            MONTH[J+1] = TEMPMONTH
        end if
        J = J+1
    end loop
    I = I+1
end loop
output "highest usage was in ", MONTH[11]
output "lowest usage was in ", MONTH[0]
```

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

A simulation is the application of a model (to a specific situation)
A simulation uses a variety of inputs in the model to determine the overall balance of the budget;

// award for **[1]** answers that do not refer to the context.

(b) (i) *Award* ***[2 max]****:*

The simulation could demonstrate that the money spent (on healthcare) will be inadequate in five years;
This could help prevent shortages in the budget;

(ii) *Award* ***[2 max]****:*

The variables used to run the simulation may be flawed/incomplete/inaccurate;
This could lead to the wrong budgets being set leaving areas incorrectly funded;

(c) *Award* ***[4 max]****:*

Data could be collected more frequently;
To detect more subtle changes in trends that would allow the predictions to be more accurate;

Data could be collected to cover more/different variables;
to give a smaller chance for outliers/unusual data from skewing the overall model;

Mark as 2 and 2

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

*Award [****2 max]*** *for concerns*
*Award [****2 max]*** *for benefits*
*Award [****1 max]*** *for conclusion*

An ethical issue is invasion of privacy;
It is possible that the data could be shared with third party individuals;
Citizens may not trust the government to use the data for purposes other than those for which it was collected;
Citizens may be concerned as to how securely their data is stored;

personal data is essential to get the best quality model ;
so that accurate tax, health, social security, earnings information may be used;

Suggest any trade-off between the invasion of privacy and the need for accurate information for future resource allocation;

6. (a) (i) *Award **[1 max]***
a flat image/a diagram, or a simple photographic type image;

(ii) *Award **[1 max]***
CGI graphics in animations/movies; a 3D model

(b) *Award **[2 max]***
The keyframe is a drawing that defines the starting or ending points;
of a smooth transition (in the animation);

(c) *Award **[5 max]***
High degree of processing between the CPU and the GPU for rendering the graphics;
Particularly when working with multiple layers;
The rendering that combines the different layers into a final 3D image;
Transitions can be processor heavy;
May require access to secondary memory that has high processing requirements;
May have issues with different memory speeds;
Rendering times can be very high if the image required is highly detailed;
Therefore, sufficient computer memory is required;

Option C – Web science

7. (a) Award ***[1 max]***

The address of a resource on the Internet / the address of a World Wide Web page;
The address of a resource on localhost / LAN server;

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

Resolve the URL or web address from a web browser and matches them to a corresponding internet protocol address;
attempts to match them locally (on primary DNS server) and if unable to send them to secondary;
and eventually top level domain name server to resolve;
Resolved URL is returned to the browser to be inserted into the TCP-IP packets;
Web addresses are resolved in the TCP component of the TCP-IP process;
If unable to resolve then an error is returned;

(c) Award ***[2 max]***

Is a secure form of HTTP;
Encrypts the data being transferred;
Uses TLS Transport layer security to provide communications security;
uses Symmetrical encryption;
Encryption keys are generated at the start of each session;
Authenticates the server which usually has a digital certificate (SSL);

(d) Award ***[2 max]***

Standards A set of guidelines or rules for a platform or language that should be adhered to;
it includes specifications of hardware, software, interfaces, interoperability and safety to ensure functionality;

Protocol – A set of formal rules describing how to transmit data, especially across a network;
Rules that must be adhered to ensure effective communication;

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

Packets/datagrams are sent without prior arrangement;
no handshake prior to transmission;
Packets/datagrams are individually addressed;
Routes for the packets/datagrams are developed from the information in the packet/datagram;
No guarantee of delivery – best effort;

Note to examiners: Connectionless is sometimes described as stateless

8. (a) (i) Award ***[3 max]***

Insert the Values into the named fields;
In the GuestsTable;
If able to connect to the server reply with success message;
if unable to connect reply with error message;

Note: accept insertion of the record for 3rd/4th marking point.

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

Web Client sends a request to server;
WebServer executes the script in a CGI software;
Webserver returns the results of the executed script to the client;
Executes the script at time of request and creates the HTML rendered out by webserver;
Used to create dynamic web pages;

(b) (i) Award ***[2 max]***

Use of appropriate metatags; Examples could include – keywords and description;
Develop search friendly URLS;
Use ALT attributes to provide keywords for images;
Refresh and update the content of the pages regularly;
Outlink to authority websites
Have clean HTML;
Keyword stuffing;
Link farms;
Cloaking;
Link-baiting;
Guest blogging -
Ability of the page to adapt to mobile devices – use of techniques like boot strap;

Note: Do not accept white hat / black hat optimization

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

Both page rank and HITS use links to calculate ranking in a search;
In page rank the number of incoming and outgoing links is used as a measure of the page;
When one page links to another it effectively votes for the page its linking to;
The Authors rank is used as a measure of importance of the link;
In HITS the quality and number of the links called the Authority and Hub Values are key metrics;
The importance of links leads to black hat techniques like link farming, blog comment links;

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

Start from Seeds (or known URLs) – initial pages to be visited;
Index/make copy of the page, download some of the content (may use robots.txt and sitemap) (amount is search engine dependent);
Find links on the page; and then visit links (could be breadth first or depth first);
Return result using criteria to provide ranking value/weight;

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

Scenario 1: Technical aspects related to the search engine having difficulty in accessing deep web.

A limitation of many search engines is that they are unable to access the deep web;
The Information in the deep web is held in databases which is not easily accessible or visible;
or searchable using popular search engines since the pages are dynamic and the content of the pages will vary depending on the query;
Information is only accessible via a query of the database;
More sites being developed using dynamic pages rather than static; less information will be accessible on a superficial level;
Rate of growth of information is 'exponential'; ability to search and index increasing volumes of data is a challenge;
number of deep-web databases is increasing and this limit indexing as it is not available;
Structure of data is not consistent;
lack of data standards – making interpretation/indexing difficult;
Further Data may be behind a paywall; or be password protected;

Scenario 2: Consequential aspects related to the search engine having difficulty in accessing deep web.

It would impact the whole purpose of having a search engine as it would miss out on a lot of information;
content that gets added to the deep web servers missing valuable and unique information such as academic journals, scientific research, which may restrict the users from accessing comprehensive and relevant search results;
leading to the inefficiency of search engines;
Inability for the search engines to access data from deep web may lead to unrealistic or unreliable data predictions/trends which may lead to lack of trust among the users for search engines;

Note to the examiners: Accept the points from one cluster only, ignore the points from second cluster if there is a mix of both.

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

Decentralized network architecture; lack of dependency on a single point of failure or central infrastructure.
Peers contribute and consume resources in a balanced manner; peers directly interact and share resources without central authority;
Enables direct communication and collaboration among peers without intermediaries;
Facilitates efficient content distribution and load balancing across the network;
Can foster community-based interactions and user-driven content discovery;

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

Award ***[2]*** *for audio compression*
Award ***[2]*** *for book/image compression.*
Award ***[2]*** *for the discussion of appropriateness*

Appropriateness of Lossy Compression for music album & e-books:
Reduced file size;
Which results in faster downloads / less storage capacity required to store the media / reduced bandwidth requirement;

Inappropriateness of Lossy Compression:
Loss of Quality; Artifacts and Distortion / loss of resolution / text;
Irreversible Data Loss;

On the album/audio:
At normal settings the impact will be minimal. The file compression will reduce the accuracy of the inaudiable frequencies;
but this will do little to reduce audiable sound quality;
Further compression below that of CD quality or reducing the bit rate may result in reduce quality of reproduction;

Inappropriateness of lossy compression for the ebook/images
The reduction in the quality of the images may result in reduced range of colours being used for each pixel;
this becomes more marked with more compression;
It also may result in the pixelation of the image;

Evaluation - candidates need to make a judgment regarding the impact of compression underpinned by a balanced analysis.

(c) Award ***[2 max]***

A service, infrastructure or platform (networks, servers, storage, applications, and services); hosted on the internet; that is often location independent;
SaaS, PaaS etc can be used to replace local storage and processing with relevant implications for the organization;
Scalability and flexibility;
Pay as per the use;

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

Log in option
Requires extra effort from the viewer to establish an account;
It will protect artist's work from unauthentic distribution to public in general.
Enables tracking of the viewers (system dependent);
Make the images harder to find in a search and therefore reduces the accessibility to the artists works;
Would need to have some form of authentication to prevent bogus accounts being established – some users may not be keen on providing suitable verification of identity;
May encure additional cost to the artist;
Does protect access but reduces visibility and accessibility to the artists work;
The login is server side and likely to be more secure;
Those having access may further distribute the downloadable images/copyrighted material without the artist's consent;

Script option
Script may be disabled or not function across all browsers; Script is client side and may be disabled;
Images are downloaded and stored in cache and may be accessible/retrievable by an experienced user (screenshot);
Allows the images to be more easily access and found using a search engine:
Less hassle for the viewer to access the images;

Note to the examiners: The response should have balance between Log in option and Script option ***[3]*** *+* ***[3]***

Option D – Object oriented programming (OOP)

10. (a) *Award **[1 max].***
These are data types that are **pre**-defined / fundamental / basic in the programming language;
A data type that is always assigned a value (in the memory);
They are the building blocks of the composite data types / classes / objects;
The types that are implemented directly as bit patterns (by a Java compiler);

(b) (i) *Award **[1 max]** for any suitable example. Allow a description.*
For example
`hasParking;`
`hasPool;`
`hasGarden;`
Whether the property has been sold or not;

(ii) *Award **[1 max]** for any suitable example. Allow a description.*

For example
`numberRooms;`
`area;`
The number of bathrooms;

(c) (i) *Award **[1 max].***
Aggregation (allow Property 'has a' Owner);

(ii) *Award **[1 max].***
Inheritance (allow House 'is a' Property);
House is a subclass of Property;
House inherits Property;

(d) *Award **[2 max].***
*Award **[1]** for distinguishing between a class and an instantiation at the definition level*;
*Award **[1]** for including an example*;

A class (e.g. House class) is a blueprint / definition / specification (that defines all variables and methods that are needed) and an instantiation of a class is creates a (new) object of that class ***[1]*** (e.g. *a (new) object of an actual house* giving its actual address etc ***[1]***;

(e) *Award **[2 max].***
A static variable is used when it is to be a class variable (belongs to the class not to the instantiation);
It is used when all objects of that class are to have the same value for the static variable;
It will not be instantiated when a new object is created;
If the value of a static variable is changed, it will be changed in all instances of that class;
It does not require an object of that class to be accessed;

(f) *Award **[2 max].***

Both classes House and Apartment need a static (integer) variable `count`;
that is incremented whenever a new object of that class is created;

A static variable (e.g `total`) could be defined in the Property class;
Which is incremented every time a House or Apartment object is instantiated;

A static method in the Property class retrieves the total number of house and apartments;
From static total variables that are in each of the 2 sub-classes;

(g) *Award **[3 max]**.*

*Award **[1]** for the identifier (left-hand side);*
*Award **[1]** for use of `new`;*
*Award **[1]** for `House[10]`;*

```
House[] wishList = new House[10];
```

11. (a) *Award* ***[1 max].***

The grouping of the data and the methods that operate on the data into one single unit;
The practice of hiding the structure and representation of data within a class with the use of private variables / and making it only accessible outside that class via accessor functions;
The practice of hiding the structure and representation of data within a class and making it only accessible directly from within that class;

(b) *Award* ***[3 max]****.*

Improved security;
Using data-hiding / Use of private variables;
So that other classes cannot (accidentally) access/change the data (directly) / So other classes have to use accessor methods to access the data / So programs can be made read-only;

Ease of maintenance / reusability;
Programs will be easier to update / reuse;
Because changes to the encapsulated class should not impact on other classes / Because all data and methods are in the same place;

Ease of understanding;
Programs will be easier to follow;
Because all data and methods are in the same place;

Ease of testing;
Because the class can be fully tested in isolation;
And limited code will contain fewer mistakes than the full program;

Faster development (of whole program);
Different classes can be worked on by teams;
Each class is effectively independent of others;

(c) *Award **[3 max]**.*
Inheritance reduces the amount of coding in the sub-class / reduces repetition of code;
By allowing sub-classes to inherit the methods / attributes of their superclass;
Therefore, leading to time being saved in the development of code;

Inheritance reduces maintenance overhead;
Because modifications to the super class are automatically inherited by the subclasses;
Reduces the errors when writing code;

Inheritance allows for further expansion / simplifies future development;
As other subclasses can inherit from the classes already present;
Or add additional functionality specific to the sub-class;

Note: only award marks from one cluster (do not mix and match the marking points from different clusters).

(d) *Award **[3 max].***
Modern programming languages use Unicode;
Which can encode 2^{16} (about 64,000) characters;
Compared to ASCII which only uses 8 bits;
Which includes special characters from different alphabets/languages/character sets;
Allowing these programming languages to be used by people from different parts of the world (only award this marking point if a correct reason has already been given);

12. (a) *Award* ***[2 max].***
The signature defines the parameters (and their types);
The method name / identifier;
Allow either the return type or access modifier for a mark;

(b) *Award* ***[2 max].***
String is an immutable object / Java uses pass-by-value / does not use pass-by-reference;
It cannot simply have its value reassigned inside the method;
The **method** type must be `String`;
To **return** the **new value** for that String variable;

(c) *Award* ***[1 max].***
To output the addresses of **all** houses in a given city;

Note to examiners. Must imply all the houses that match, not just one.

(d) (i) *Award* ***[2 max]***
The length of the array may not correspond to the number of houses stored;
Causing a null pointer error/exception;
Since the loop may try to access the `getCity` method for a null entry;

(ii) *Award* ***[2 max.]***
Declare a variable `count` that stores the number of objects in `allHouses`;
Change loop condition to `j<count`;

Add a test `if(allHouses[i]!=null);`
<u>Before</u> testing `allHouses[i].getCity().equals(x) ;`

Use a while loop with a double condition;
`(i<allHouses.length)&&(allHouses[i]!=null) ;`

Initially fill the array with dummy objects;
So that no null entries will be encountered;

Change the array to an arrayList;
So that no null entries will be encountered / so that the array has the exact number of valid objects in it;

(e) *Award **[5 max]**.*
*Award **[1]** for correctly declaring a swap variable;*
*Award **[1]** for correct outer loop;*
*Award **[1]** for correct inner loop;*
*Award **[1]** for correctly testing prices (allow* `.price`*);*
*Award **[1]** for correct swap statements (ignore here any incorrect declaration of swap variable);*

Notes to examiners.

- *As there are many variations on sort routines you will need to carefully read through each response. You can ignore minor syntax errors in every algorithm answer.*
- *Note that is the House objects that are being swapped, not the prices. However, if prices are being swapped, allow FT for ta swap variable declared as a numeric type.*

Example answer 1

```
public void houseSort()            // basic bubblesort 1
{
  House swapHouse;
  for (int i=0; i< allHouses.length-1; i++ //accept
                                           //allHouses.length
  {
    for (int j=0; j< allHouses.length-1; j++)
    {
      if (allHouses[j].getPrice()>allHouses[j+1].getPrice())
      {
        swapHouse = allHouses[j];
        allHouses[j] = allHouses[j+1];
        allHouses[j+1] = swapHouse;
      }
    }
  }
}
```

Example answer 2

```
public void houseSort()              // basic bubblesort 2
{
  boolean isSorted = false;
  House houseTemp;
  while(!isSorted)
  {
    isSorted = true;
    for(int j=0;j<allHouses.length-1;j++)
    {
      if(allHouses[j].getPrice()>allHouses[j+1].getPrice())
      {
         houseTemp = allHouses[j];
         allHouses[j] = allHouses[j+1];
         allHouses[j+1] = houseTemp;
         isSorted=false;
       }
     }
   }
 }
```

Example answer 3

```
public void houseSort()           // basic selection sort 1
{
  House swapHouse;
  int smallest;
  for (int i=0; i< allHouses.length-1; i++)
  {
    smallest = i;
    for (int j=i+1; j< allHouses.length; j++)      // allow j =i
    {
      if (allHouses[j].getPrice()<allHouses[smallest].getPrice())
      {
        smallest = j;
      }
    }
      swapHouse = allHouses[smallest];
      allHouses[smallest] = allHouses[i];
      allHouses[i] = swapHouse;
  }
}
```

Example answer 4

```
public void houseSort()           // basic selection sort 2
{
   House swapHouse;
   for (int i=0; i< allHouses.length-1; i++)
     {
    for (int j=i+1; j< allHouses.length; j++)      // allow j = i
    {
      if (allHouses[i].getPrice()>allHouses[j].getPrice())
      {
        swapHouse = allHouses[i];
        allHouses[i] = allHouses[j];
        allHouses[j] = swapHouse;
      }
    }
  }
}
```

(f) *Award **[7 max]**.*
*Award **[1]** for correct return type and return;*
*Award **[1]** for correct parameter, budget (allow* `allHouses` *to be passed as well);*
*Award **[1]** for correctly instantiating a result array;*
Award [1] for correctly using `houseSort`*;*
*Award **[1]** for while loop with one correct condition; // other loops could be used*
*Award **[1]** for while loop with two correct conditions in the right order; // note that some of these conditions may be implemented as IF statements;*
*Award **[2]** for assigning all 3* `House` *objects in the correct order (award **[1]** for good attempt at identifying the 3 objects);*

Example answer 1

```
public House[] selectThree(int budget)
{
  House[] result = new House[3];
  houseSort();                    // allow allHouses.houseSort()
  int i=0;
  while ((i<allHouses.length)&&(allHouses[i]!=null)
                             &&(allHouses[i].getPrice()<=budget))
  {
    i++;
  }
  i--;
  for (int a=0; a<=2; a++)
  {
    result[a] = allHouses[i-2+a];
  }
  return result;
}
```

Example answer 2

```
public House[] selectThree(int budget)
{
  House[] result = new House[3];
  houseSort();
  int i=0;
  while ((i<allHouses.length)&&(allHouses[i]!=null)
                             &&(allHouses[i].getPrice()<=budget))
  {
    i++;
  }
  i--;
  result[0] = allHouses[i-2];
  result[1] = allHouses[i-1];
  result[2] = allHouses[i];
  return result;
}
```

Example answer 3

```
public House[] selectThree(int budget)
{
  House[] result = new House[3];
  houseSort();
  int index = 2;
  for(int i=allHouses.length-1; i>=0 && index>=0; i--)
  {
    if(allHouses[i]!=null && allHouses[i].getPrice()<=budget)
         result[index--] = allHouses[i];
  }
  return result;
}
```