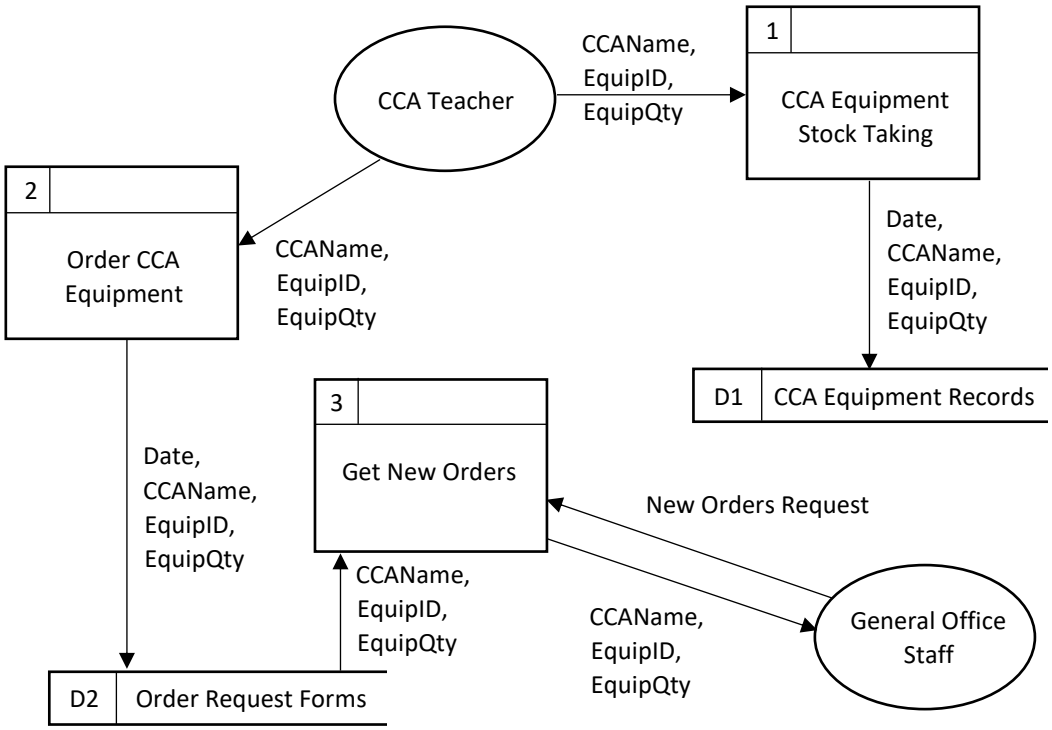
2019 SH2 Preliminary Examination – Theory – Suggested Solutions and Marking Scheme

1a	Relevant constraints: 1. Cost and Budget – development cost of the new system and if the school has the necessary funds 2. Training – What are the training implications once the system is developed? 3. Technical feasibility – technically possible to create system based on requirements? • **1 mark for each point to a maximum of 2 marks [2]**

1b	Alternative solutions: 1. Off-the-shelf solution - The school may consider looking for an existing inventory control system that meets all the needs of the school in terms of tracking CCA equipment inventories. 2. Partial off-the-shelf solution with some customisation – The school may consider looking for an existing inventory control system (including both hardware and software) that closely meets the needs of the school. Once purchased, this system will then be customised to meet the needs of the school in terms of tracking CCA equipment inventories 3. Service rental solution – The school may also seek to contract the stock taking process to a third party that would service the system at a cost based on usage (this is similar to the usage of a cloud service, except, with additional hardware and physical requirements). • **1 mark for each point to a maximum of 2 marks [2]** • **1 additional mark for each appropriate example (corresponding to the points made) [2]**

1c	Systems Analyst requirements analysis mechanisms: 1. Face-to-face Interviews – The analyst interviews school staff and management, in particular the CCA teachers directly utilising the system (e.g., data entry, inventory-taking process, etc.). In particular, the analysts should try to determine what the main problems are and whether users have any suggestions on how to improve the way things work. 2. Observation – The analyst will observe several CCA teachers with more automated (i.e., less manual) systems actually using their current systems. They will probably follow a complete process from start to finish and note down every interaction that happens 3. Questionnaires – Questionnaires enable the analyst to obtain the views of a large number of staff/ users. Questionnaires are also easier to analyse than face-to-face interviews but the trade-off is that they don't give as much detail. The analysts should use these to question CCA teachers about the features they would like to see in the system developed. 4. Examination of business documents – Examining all documentation pertaining to processes currently adopted, as well as the record-keeping and report generation requirements. • **1 Mark for each point to a maximum of 2 marks (including a simple description and mention of relevant parties) [2]**

1d



- **At least 1 (relevant) Data Entity, 1 Process and 1 Dataflow [1]**
- **Stock Take process with relevant dataflows [1]**
- **Order Equipment process with relevant dataflows [1]**
- **Process Orders process or Get Order Information process with relevant dataflows [1]**

1e

DFDs are typically used during:

1. **Analysis Phase** as part of the **Requirements documentation** – DFDs are used to capture the scope of the system, indicating which elements are inside and outside the system, as well as to specify which people and technologies are used in which processes to move and transform data, accepting inputs and producing outputs. These may allow one to highlight issues with the current system.
2. **Design Phase** as part of the **Design documentation** – DFDs may also be employed as a top-down design tool for the proposed system, describing how different parts of the system (e.g., users, applications) will interact.

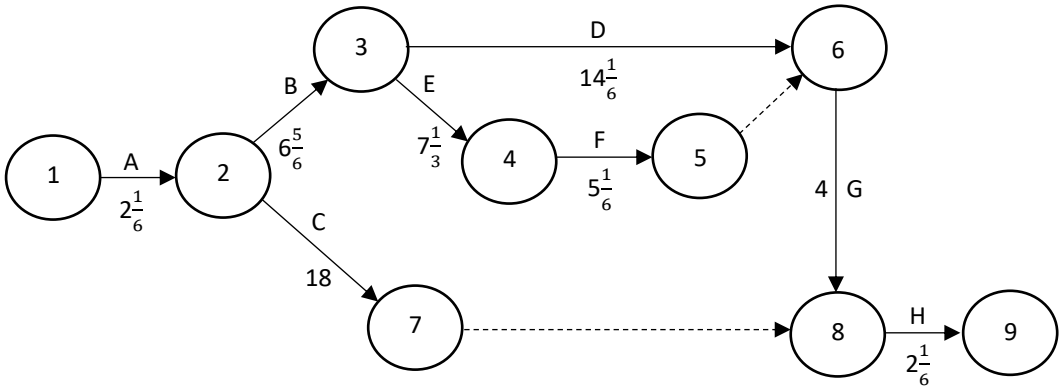
- **1 mark for each appropriate phase and document mentioned, to a maximum of 2 marks [2]**
- **1 mark for each description of the DFD used in the specified phase, to a maximum of 2 marks (this description must correspond to the specified phase/document) [2]**

1f

A	B	C	D	E	F	G	H
$2\frac{1}{6}$	$6\frac{5}{6}$	18	$14\frac{1}{6}$	$7\frac{1}{3}$	$5\frac{1}{6}$	4	$2\frac{1}{6}$

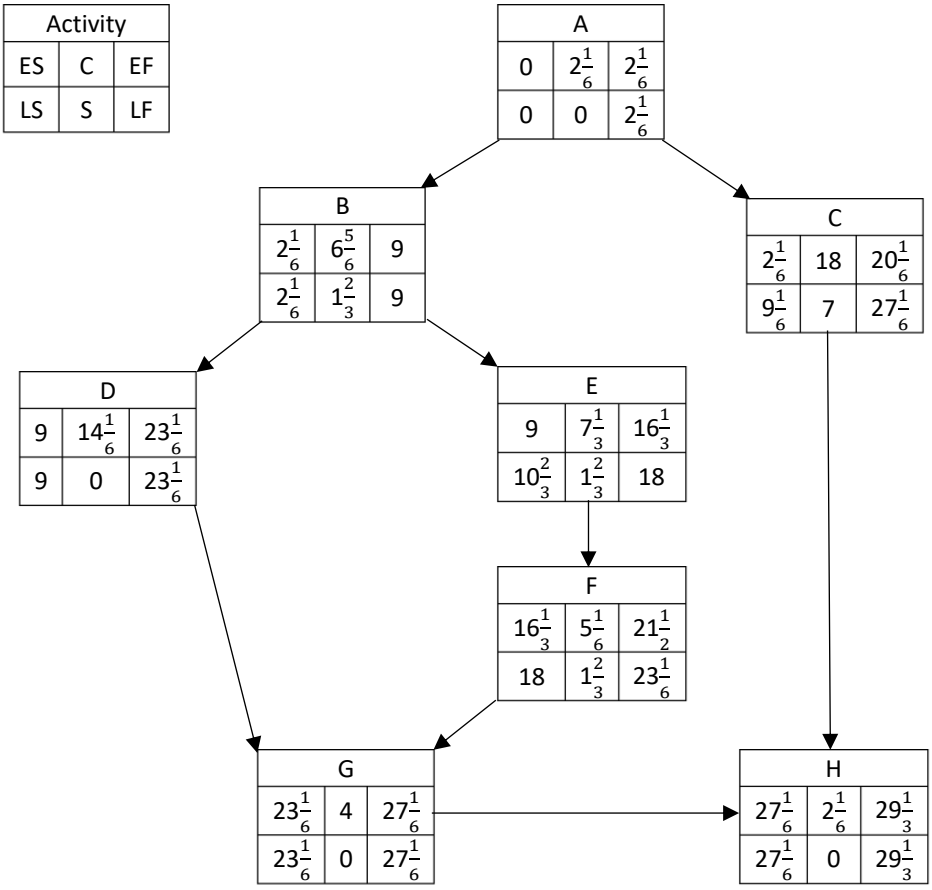
- **1 Mark for all correct values [1]**

1g



- All activities present with appropriate times (ECF) [1]
- Dummy activity after F [1]
- Dummy activity after C [1]

1h



- **Early Start and Early End times on all activities [1]**
- **Late Start and Late Finish times on all activities [1]**
- **Slack times [1]**

1i	Prior to the Analysis Phase. Once the initial feasibility study is conducted and the client decides to go ahead with the project, the Project Manager must then plan for resource requirements and the project timeline. • **Prior to Analysis Phase [1]** • **In order for the Analysis Phase (and beyond) to begin, the planning of resources and the project schedule must have been completed [1]**

1j	The test plan is typically drafted in the implementation/development phase, and may go through several iterations during this phase. The test plan is not drafted during the analysis or requirements phase as specifics of the implementation may not be ready, and as such, the test cases cannot be adequately defined. If testing is done only after the implementation phase, then failures may cascade and cause substantial loss in project resources. • **Implementation/Development Phase [1]**

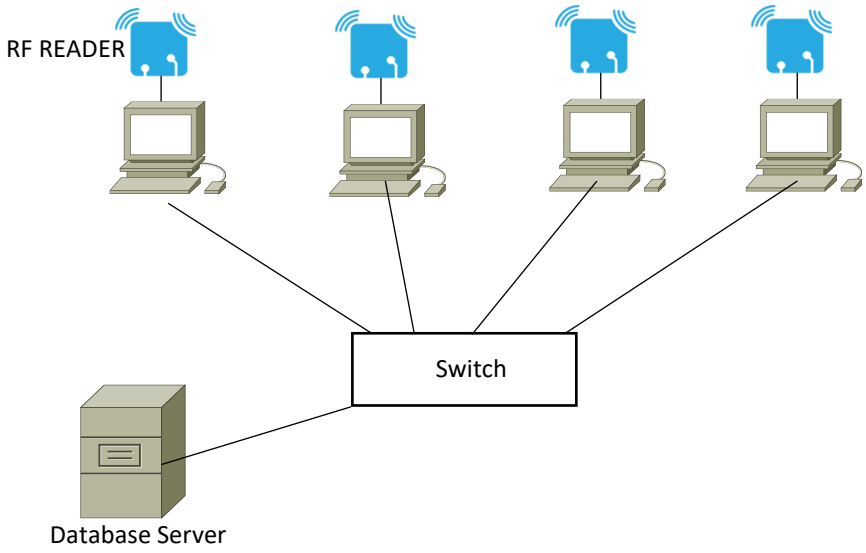
1k	SDLC methodologies include: 1. Agile development 2. Waterfall development 3. Spiral development • **1 mark for each method listed (including any other valid methods not listed above), up to a maximum of 3 marks [3]**

1l	Agile development. Since this is a small project, it would benefit from the Agile development methodology. That is, instead of having monolithic phases based on a waterfall development methodology, which is meant for much larger-scale projects. It would be more beneficial to repeat the: analyse > develop and test > get feedback cycle to more quickly develop the desired application. • **Agile development [1]** • **Context methodology → Waterfall development [1]** • **Agile suits smaller-scaled project [1]**

1m	Installation Phase. • **Installation Phase [1]**

1n	Perfective maintenance – To find tweaks or minor improvements that could be made to improve the way the system works. For example, changes made to hardware used in the system, or the layout of hardware, and changes to interface design. Adaptive maintenance – This type of maintenance often occurs as a result of external influences or strategic changes within the company. For example, changes to the requirements in terms of stock taking and equipment orders. • **Perfective maintenance with description and example based on context [1]** • **Adaptive maintenance with description and example based on context [1]**

1o



[2]

1 m for switch
1m for all wired connections
If wireless is used, there must a WAP connected to switch

1p

- Simplex communication is needed for the reader to retrieve data from the RF reader

[1]

1q

- User's permission are not sought. This violates the Personal Data Protection Act in Singapore even when the tag is used for tracking purposes only.
- Usage of the data for trackin purposes outside of school violates the Personal Data Protection Act in Singapore.
- User's are not aware that data are being accessed. This can be considered a data theft, where data associated with the tag can be used to retrieved more sensitive data frm another source.
- Even if the user is informed of the usage of the tag, in this case for taking attendance. The original intent can be changed without the knowledge of the user.

Any 2, 1m each

[2]

2a	A command line interface (CLI) allows the user to interact directly with the computer system by typing in commands (i.e., instructions) into a terminal (window). In contrast, a graphical user interface (GUI) is a typically more user-friendly interface that allows users to interact with the computer via the use of a point-and-click mechanism and buttons/icons. CLIs are used for users with higher expertise that require more flexibility within their interaction with the computer. Whereas GUIs are for more basic users who require more static functionality. CLIs are less resource intensive as compared to GUIs, and are thus more suitable for less powerful devices. • **Distinction between the 2 interfaces [1]** • **Applicability either in terms of user requirements, or technical requirements [1]**

2b	**1. Strive for consistency** - Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent colour, layout, capitalization, fonts, and so on, should be employed throughout. Exceptions, such as required confirmation of the delete command or no echoing of passwords, should be comprehensible and limited in number **2. Seek universal usability** - Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Novice to expert differences, age ranges, disabilities, international variations, and technological diversity each enrich the spectrum of requirements that guides design. Adding features for novices, such as explanations, and features for experts, such as shortcuts and faster pacing, enriches the interface design and improves perceived quality. **3. Offer informative feedback** - For every user action, there should be an interface feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. **4. Design dialogs to yield closure** - Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives users the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions. **5. Prevent errors** - As far as possible, design the interface so that users cannot make serious errors. If users make an error, the interface should offer simple, constructive, and specific instructions for recovery. Erroneous actions should leave the interface state unchanged, or the interface should give instructions about restoring the state. **6. Permit easy reversal of actions** - As far as possible, actions should be reversible. This feature relieves anxiety, since users know that errors can be undone, and encourages exploration of unfamiliar options. **7. Keep users in control** - Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don't want surprises or changes in familiar behaviour, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result. **8. Reduce short-term memory load** - Humans' limited capacity for information processing in short-term memory requires that designers avoid interfaces in which users must remember information from one display and then use that information on another display. 1. **1 mark for each of the above, up to a maximum of 4 [4]**

3a

```
PROCEDURE pascals(n: INTEGER)
    DECLARE prevRow, currRow: LIST OF INTEGER
    DECLARE currStr: STRING
    DECLARE currVal: INTEGER
    IF n < 1 THEN
        OUTPUT "Invalid input; requires n > 0."
    ELSE
        OUTPUT "1"
        IF n ≥ 2 THEN
            prevRow.INSERTBACK(1)
            prevRow.INSERTBACK(1)
            OUTPUT "1 1"
        ENDIF
        IF n ≥ 3 THEN
            WHILE prevRow.LENGTH() < n DO
                currRow.EMPTY()
                currRow.INSERTBACK(1)
                currStr ← "1 "
                FOR i = 1 TO prevRow.LENGTH() - 1 // all but last element
                    currVal ← prevRow.GET(i) + prevRow.GETIDX(i + 1)
                    currRow.INSERTBACK(currVal)
                    currStr ← CONCATENATE(currStr, STR(currVal), " ")
                ENDFOR
                currRow.INSERTBACK(1)
                currStr ← CONCATENATE(currStr, "1")
                OUTPUT currStr
                prevRow ← currRow
            ENDWHILE
        ENDIF
    ENDIF
ENDPROCEDURE
```

- **Base cases; accurate when n = 1 and 2 [1]**
- **Main (outer) loop condition and iteration (for required rows) [1]**
- **Nested (inner) loop condition and iteration (for each row) [1]**
- **All non-base cases (i.e., n > 2) accurate [1]**

3b

Valid boundary test case: n = 1

Expected Output: `"1"`

- **An appropriate value of n and its corresponding expected output [1]**

3c

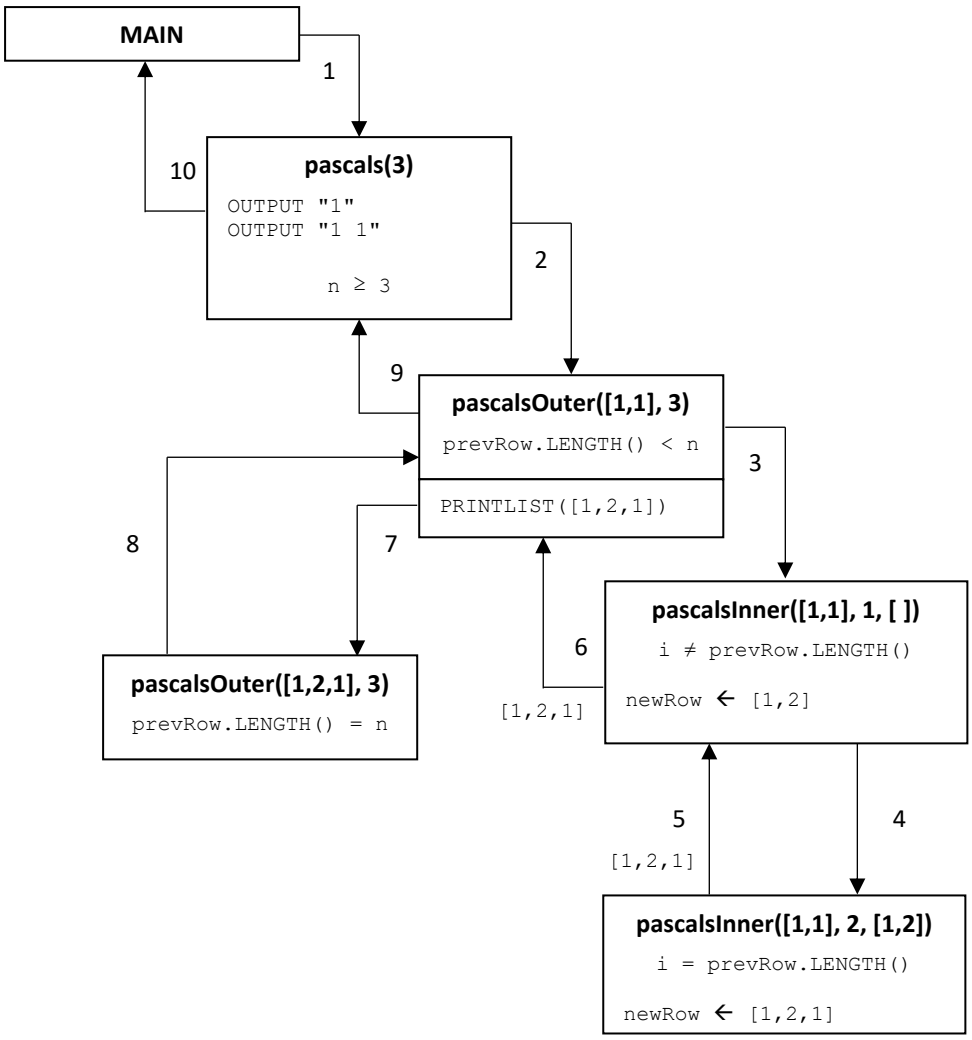
```
FUNCTION pascalsInner(prevRow: LINKEDLIST, i: INTEGER,
                      newRow: LINKEDLIST) RETURNS LINKEDLIST
    IF i = prevRow.LENGTH() THEN // at last index
        newRow.INSERTBACK(1)
        RETURN newRow
    ELSE
        IF i = 1 THEN
            newRow.INSERTBACK(1)
        ENDIF
        newRow.INSERTBACK(prevRow.GETINDEX(i) + prevRow.GETINDEX(i + 1))
        RETURN pascalsInner(prevRow, i + 1, newRow)
    ENDIF
ENDPROCEDURE

PROCEDURE pascalsOuter(prevRow: LINKEDLIST, n: INTEGER)
    DECLARE newRow: LINKEDLIST
    IF prevRow.LENGTH() < n THEN
        prevRow <- pascalsInner(prevRow, 1, newRow)
        PRINTLIST(prevRow) // prints contents of prevRow as specified
        pascalsOuter(prevRow, n)
    ENDIF
ENDFUNCTION

PROCEDURE pascals(n: INTEGER)
    DECLARE prevRow: LINKEDLIST
    IF n < 1 THEN
        OUTPUT "Invalid input; requires n > 0."
    ELSE
        IF n ≥ 1 THEN
            OUTPUT "1"
        ENDIF
        IF n ≥ 2 THEN
            OUTPUT "1 1"
            prevRow.INSERTBACK(1)
            prevRow.INSERTBACK(1)
        ENDIF
        IF n ≥ 3 THEN
            pascalsOuter(prevRow, n)
        ENDIF
    ENDIF
ENDPROCEDURE
```

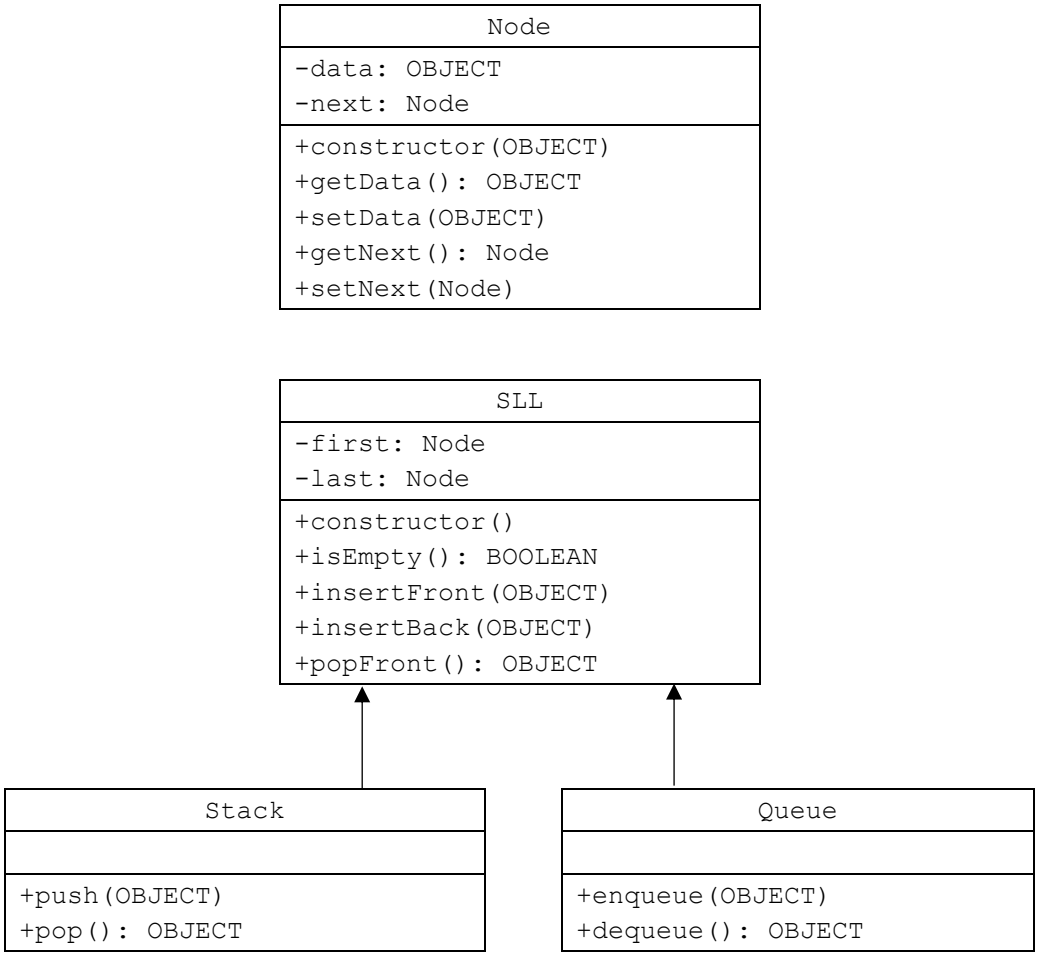
- **Base cases; accurate when n = 1 and 2 [1]**
- **Main (outer loop) recursion (for required rows) [1]**
- **Nested (inner loop) recursion (for each row) [1]**
- **All non-base cases (i.e., n > 2) accurate [1]**

3d



- **All calls to Outer Loop functions/procedures present (with correct arguments) [1]**
- **All calls to Inner Loop functions/procedures present (with correct arguments) [1]**
- **Accurate sequence of all calls and returns [1]**
- **Accurate return values for resolved function calls, and appropriate output shown [1]**

4ai



All 4 methods in the Stack and Queue classes are wrapper methods:

- `push(OBJECT)` calls:
  - `insertFront(OBJECT)`
- `pop(): OBJECT` calls:
  - `isEmpty(): BOOLEAN`
  - `popFront(): OBJECT`
- `enqueue(OBJECT)` calls:
  - `insertBack(OBJECT)`
- `dequeue(): OBJECT` calls:
  - `isEmpty(): BOOLEAN`
  - `popFront(): OBJECT`

- **SLL class attributes and methods [1]**
- **Stack and Queue class attributes and methods [1]**
- **Stack and Queue classes inherit Singly-linked List (SLL) symbolised [1]**
- **Fully Modular [1]**

4aii

A doubly or circular doubly-linked linked list is unnecessary for dynamically (i.e., node-based) Stacks and Queues as only require 3 methods. InsertFront and PopFront only require access to the head/first attribute within the list, and InsertBack, which only requires access to the tail/last attribute within the list.

- **Identification of the 3 necessary methods and that none of them require a doubly-linked or a circular doubly-linked linked-list to function as expected [1]**

4bi

```
FUNCTION quadProbe(ht: ARRAY OF OBJECT, data: OBJECT) RETURNS INTEGER
    DECLARE currIndex, step: INTEGER
    IF NOT ht.ISFULL() THEN
        currIndex ← HASH(data) % ht.LENGTH()
        step ← 1
        WHILE ht[currIndex] <> NULL DO
            // Quadratic Probing
            currIndex ← currIndex + step * step
            IF currIndex > ht.LENGTH() THEN
                currIndex ← 1
            ENDIF
        ENDWHILE
        Ht[currIndex] ← data
    ELSE
        OUTPUT "Unable to insert into Hash Table. It is full."
    ENDIF
ENDFUNCTION
```

- **Quadratic probing:**
  - **Step size is exponential (e.g., 1, 4, 9, …) [1]**
  - **Increment of step size in each iteration [1]**
- **Wrapping when current index exceeds the length of the Hash Table [1]**

4bii

Linear probing utilises a step size that is static, whereas quadratic probing utilises a step size that is growing (exponentially).

The problem with linear probing is that there is a tendency for elements to be clustered together, which increases the chances that new elements being inserted into the Hash Table will also require linear probing. Quadratic probing attempts to alleviate this problem by better spreading elements that have encountered collisions.

- **Linear and quadratic probing difference [1]**
- **Issue with linear probing ("causes clustering" is sufficient) [1]**

4biii	It is invalid. Consider the following counter example. Given a Hash Table [1, NULL, 3], and new element 1, which has hashValue 1. Iteration 1: HT[1] <> NULL, so hashValue ← 1 + (-1), but since hashValue < 1, hashValue ← 3, and step ← (-1) * (-2) Iteration 2: HT[3] <> NULL, so hashValue ← 3 + 2, but since hashValue > 3, hashValue ← 1, and step ← 2 * (-2) Iteration 3: HT[1] <> NULL, so hashValue ← 1 + (-4), but since hashValue < 1, hashValue ← 3, and step ← (-4) * (-2) Iteration 4: HT[3] <> NULL, so hashValue ← 3 + 8, but since hashValue > 3, hashValue ← 1, and step ← 8 * (-2) … From the above, we notice that the INSERT function enters an infinite loop is formed as the hashValue simply alternates between 1 and 3. It is thus invalid. • **Invalid answer given [1]** • **Counter example provided [1]**

5a	In synchronous communication, the signal or control information is not embedded in the data stream, instead it utilises a separate channel for signalling. Hence a large amount of data can be transferred compare to asynchronous mode. In asynchronous mode, the signalling or control information is embedded in the data stream, e.g., start/stop bits. Hence the actual bandwidth for data is reduced. • **Separate channel for signalling in synchronous communication, but embedded for asynchronous communication [1]** • **Higher bandwidth with synchronous communication [1]**

5b	Layer 1: Datalink. Examples are Ethernet, Bluetooth, 802.11(WIFI). Uses MAC addressing to handle point-to-point communication. Layer 2: Network. Handles routing data from one network to another. Uses logical addressing via IP address. Layer 3: Transport. Handles end-to-end communication. Responsible for re-assembly of packets and re-transmission of loss packets. Layer 4: Application. Business logic of the application. Protocols like HTTP, SMTP, FTP, Telnet. • **1 mark for each of the above points (requires mode of addressing and description of functionality) [4]**

5ci	Each signal needs to carry 4 bits. Therefore, level requires is 2^4 = 16 levels. • **16 levels [1]**

5cii	A circuit switching network. • **Circuit switching [1]**

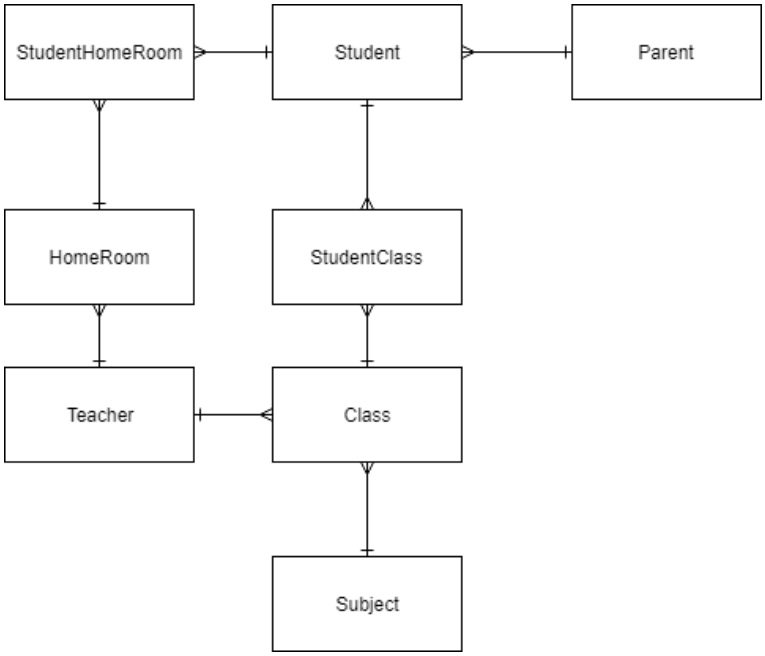
5ciii	Point 1: In a circuit switching network, the bandwidth required for the connection is reserved or pre-allocated in the entire communication path/route before the data is sent. Point 2: Only after the resource pre-allocation (i.e., call setup) phase is the actual data sent. OR (for ECF) Point 1: In a packet switching network, the data to be sent is broken into chunks. Point 2: These chunks are send independent of each other and can take different routes to the destination. • **Either Point 1 [1]** • **Either associated Point 2 [1]**

5di	A socket is an end point in data communication; it is represented by an IP address and port number. • **Socket definition and composition [1]**

5dii	UDP. It is faster than TCP as no buffering is required, which affords best effort delivery. • **UDP [1]** • **No buffering thus faster [1]**

5e	Symmetric key, where both the sender and the receiver use the same secret key for encryption and decryption. Asymmetric keys, where the sender uses the public key of the receiver to encrypt, and the receiver uses its private key to decrypt or vice versa. • **Symmetric key [1]** • **Asymmetric key [1]**

6a



- **Student, Class, Subject, Teacher, Parent entities present [1]**
- **StudentClass entity present [1]**
- **Homeroom and StudentHomeRoom entities present [1]**
- **All relationship cardinalities [1]**

6b

Parent(ParentID, Name, Contact, Email)
Student(StudentID, Name, Contact, Address, ParentID*, ParentRelationship)
Class (ClassID, Year, SubjectID*, TeacherID*)
StudentClass(StudentID*, ClassID*,Year*)
Subject(SubjectID, Level, SubjectDescription)
Teacher(TeacherID, Name, Contact)
HomeRoom(HomeRoomID, Year, TeacherID*)
StudentHomeRoom(StudentID*, HomeRoomD*, Year*)

- **All relations (as presented in 6a) present in 1NF [1]**
- **All primary keys clearly indicated [1]**
- **All foreign keys clearly indicated [1]**
- **All relations in 2NF [1]**
- **All relations in 3NF [1]**
- **Year attribute present in Class and Homeroom [1]**

6c

DDL or data definition language is used to define/create the tables of the database, whereas DML, or data manipulation language is used to manipulate the data in the tables (e.g., insert, update, delete).

- **Different categories of instructions for each [1]**

6d	A data dictionary contains: 1. The tables in the database – presents the contents of the database so that the scope of data is known. 2. Constraints over each table – describes the expectations on the data stored so that these expectations may be adopted as required. 3. Relationships between attributes – describes the relationships so that the data to facilitate query crafting. 4. Authorisation information – to review or augment ownership and responsibility. 5. Timestamps on access and updates – to help audit access and changes made. • **1 mark for each point above, up to a maximum of 4 marks [4]**