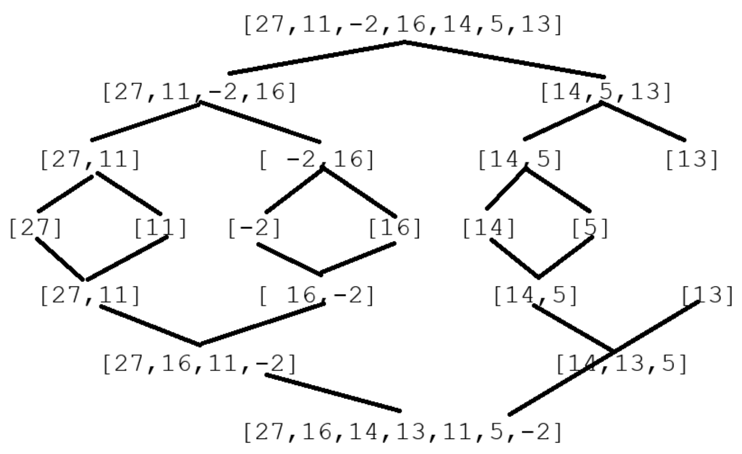
**Jurong Pioneer Junior College**

**2022 JC 2 H2 Computing Mock Practice 4 (Paper 1)**

| | |
|---|---|
| **1** | **(a)**<br>Able to call itself in its body (general case)<br>Able to terminate/complete when the base case has been reached |
| | **(b)**<br>**1.** By finding the middle position of the unsorted array and **recursively divide** the unsorted array into **2 halves** until **multiple subarrays of single element** remains.<br>**2.** Compare by sorting and merging the single element subarrays from bottoms up into pairs.<br>**3. Recursively sort and merge** the elements of the subarrays to produce new sorted subarrays until a single sorted array is formed eventually.<br>**3 marks** |
| | **(c)**<br> [27,11,-2,16,14,5,13]<br>[27,11,-2,16]<br>[14,5,13]<br>[27,11]<br>[ -2,16]<br>[14,5]<br>[13]<br>[27]<br>[11]<br>[-2]<br>[16]<br>[14]<br>[5]<br>[27,11]<br>[ 16,-2]<br>[14,5]<br>[13]<br>[27,16,11,-2]<br>[14,13,5]<br>[27,16,14,13,11,5,-2] |
| | **(di)**<br>In the following order:<br>`mergeSort(0, 3)`<br>`mergeSort(0, 1)`<br>`mergeSort(0, 0)`<br>`mergeSort(1, 1)`<br>`merge(0, 0, 1)`<br>`mergeSort(2, 3)`<br>`mergeSort([2, 2)`<br>`mergeSort([3, 3)`<br>`merge(2, 2, 3)`<br>`merge(0, 0, 3)`<br>**(dii)**<br>7 times |
| | **(e)** |

When a subroutine calls itself, its information like return address, parameters, and local variables, etc need to be stored somewhere.

When the base case subroutine returns, the program needs to retrieve the stored information of the current's subroutine caller function from the stack

**(f)** Worst case time complexity = O(nlog$_2$n)

**(g)**
Merge sort always splits the data set in two equal halves and take linear time to merge two halves regardless of the initial order of the values in the data set.

**2**

**(b)**
**Note:**

- if order >$500 is True, then there no more need to consider whether order is >$250 **<u>and</u>** if distance is within 10km.
- if order > $250 but <=$500, then consider if distance is within 10km.

| **Conditions** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1. Is order > $500? | **Y** | **N** | **N** | **N** | **Y** | **N** | **N** | **N** |
| 2. Is order > $250? | **-** | **Y** | **Y** | **N** | **-** | **Y** | **Y** | **N** |
| 3. Is distance <= 10km? | **-** | **Y** | **N** | **-** | **-** | **Y** | **N** | **-** |
| 4. Is member a VIP? | **N** | **N** | **N** | **N** | **Y** | **Y** | **Y** | **Y** |
| **Actions** | | | | | | | | |
| $0 | **X** | | | | **X** | | | |
| $20 delivery | | **X** | | | | **X** | | |
| $30 delivery | | | **X** | | | | **X** | |
| $50 delivery | | | | **X** | | | | **X** |
| Give $15 discount | | | | | **X** | **X** | **X** | **X** |

**3** **(a)**

- Perform linear search for a key specified in a sorted array of integers
- Returns the index position when found and 0 when not found.
- Iteratively checks the array item by item, without the possibility of skipping, starting with the first item in the array (ie. item with index position 1)

**(b)**

| key | index | ar[index] | Remarks |
|---|---|---|---|
| 54 | 1 | 3 | REPEAT |
| 54 | 2 | 12 | REPEAT |
| 54 | 3 | 32 | REPEAT |
| 54 | 4 | 54 | RETURN 4 |

**(c)(i)**
Line 5.
Index out of bound error.

**(c)(ii)**
Line 10: `UNTIL index - 1` **>=** `LEN(ar)` ……

```
Or UNTIL index - 1 = LEN(ar)
Or UNTIL index - 1 > LEN(ar) - 2
```

**(c)(iii)**

6 times

**(d)**

```
WHILE index <= len(ar) and key >= ar[index]:
  IF key = ar[index]
      THEN
          RETURN index
      ENDIF
      index □ index + 1
ENDWHILE
```

**(e)**

Comments

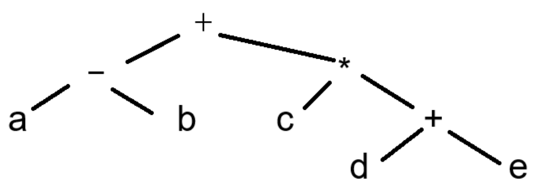
Proper code indentation

Meaningful variable names

**4** **(a)(i)**

- binary tree data structure can represent the operators and operands of an algebraic expression in its nodes.
- a leaf node will contain an operand of the algebraic expression,
- while an interior node contains an operator of the algebraic expression.

**(a)(ii)**



**(b)**

Note: Remember to denote Root.

| **Array index** | **LeftPointer** | **Item** | **RightPointer** |
|---|---|---|---|
| **1** | 0 | d | 0 |
| **2** | 0 | e | 0 |
| **3** | 4 | + | 5 |
| **4** | 6 | – | 7 |
| **5** | 8 | * | 9 |
| **6** | 0 | a | 0 |
| **7** | 0 | b | 0 |
| **8** | 0 | c | 0 |
| **9** | 1 | + | 2 |

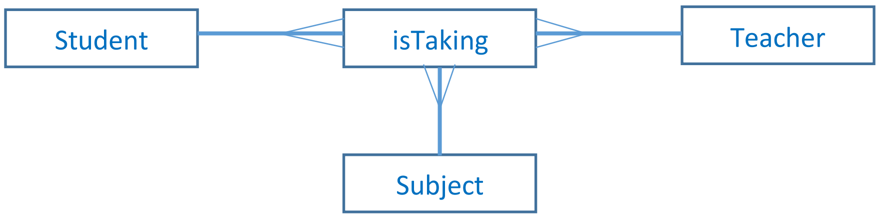
Root: $\boxed{3}$

**c(ii)** post order representation: `a b – c d e + * +`

**(e)**

|   |   |
|---|---|
|   | ***Note:** `stackPtr` defined in this question points to the topmost item in stack (ie. most recently pushed item)<br>`PROCEDURE PUSH(item : CHAR):`<br>`    IF s = 5 THEN`<br>`        RETURN NULL //stack is full`<br>`    ENDIF`<br>`    stackPtr ← stackPtr + 1`<br>`    s[stackPtr] ← item`<br>`ENDPROCEDURE`<br>`FUNCTION POP():`<br>`    IF s = 0 THEN`<br>`        RETURN NULL //stack is empty`<br>`    ENDIF`<br>`    temp ← s[stackPtr]`<br>`    stackPtr ← stackPtr - 1`<br>`    RETURN temp`<br>`ENDFUNCTION` |
| **5** | **(a)(i)**<br>A hashing algorithm refers to a formula that uses the key of an item to compute an address of a location that is used to store the item. |
|   | **(a)(ii)**<br>Hash address is the location of the data structure that stores the item. |
|   | **(b)**<br>Collisions happen when keys of different items get hashed to the same location (ie. hash address). Collisions can be handled using linear probing or chaining. |
|   | **(c)**<br>User ID and password<br>Electronic records of individuals, each can be identified by a unique key. |
| **6** | **(a)**<br>The table is not in 1NF as there are attributes that do not contain atomic values. For example, there are multiple values of SubjectID and Subject in one record. |
|   | **(b)**<br>Student (PersonID, FullName, L1R5)<br>Teacher (PersonID, FullName, Department)<br>Subject (SubjectID, SubjectName)<br>isTaking (PersonID*, SubjectID*)<br><br>*The suggested solution is not the only solution. An alternative solution for (b) will produce a solution that is different from the proposed solution for (c). |

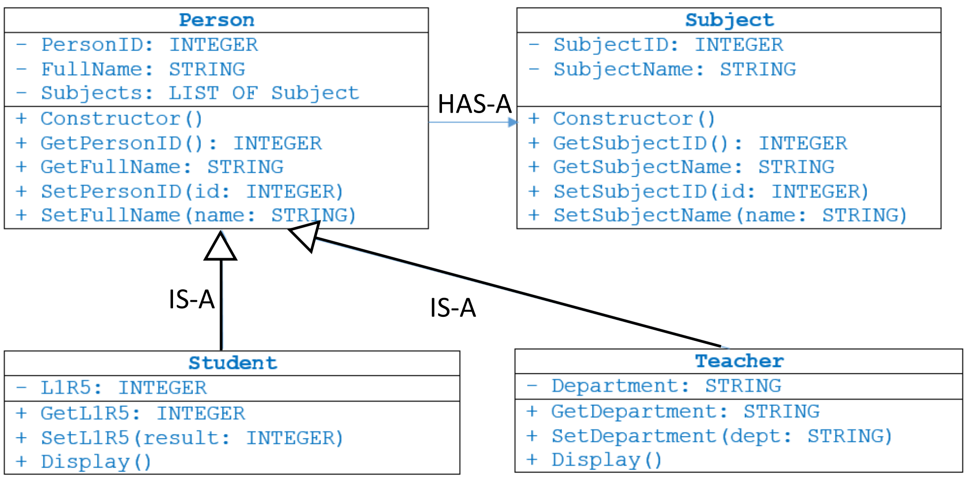
**(c)**



**(d)**

Reducing data redundancy reduces, insert, update, and delete anomalies leading to data inconsistencies compromising data integrity.
It also saves memory space potentially enhancing the efficiency of the database since it is smaller in size.

**(e)**



**(f)**

The purpose of a superclass is to allow reusable code by allowing subclasses to inherit its attributes and methods while extending them further.

**7** **(a)**

First, the sender will interact with the software in the Application Layer he/she intends to send the music file to. The sender will trigger the send where the music file will be handed to the Transport Layer.

At the Transport Layer, the music file will be broken down into data packets. These data packets will each be labelled with a sequence number and check digit and sent to the next layer, the Network Layer.

At the Network Layer, the data packets will be each wrapped with the IP address of the recipient and also the IP address of the sender. After which, they will be sent to the Data Link Layer.

At the Data Link Layer, the data packets will be sent across to the recipient over the physical network. Any data packets lost in the transmission will be asked by the recipient's end to be resent.

Finally at the receiver's end, the data packets are handed back up the layers starting from Data Link Layer to Network Layer to Transport Layer and finally to the Application Layer until eventually all the data packets will be accounted for, arranged and assembled in the correct sequence and presented the recipient through the software he/she uses to receive the music file.

**(b)**

- Packet switching is when **data is broken into** small units known as **data packets** that are **are transmitted/routed individually** through a network.
- In order to ensure **each packet takes the best route available** at any given time, **routers** can detect busy routes and **send data packets around different paths to get them to their destination**.

In a connection-oriented packet-switched network,

- the **first packet** sent from the sender to receiver will **include a request for an acknowledgement** from the receiver.
- Once the sender **receives the acknowledgement** from the receiver, it will then **continue to transmit the remaining packets**.
- If **no acknowledgement is received**, the **sender will try again** by sending the **first packet**.

**(c)**

Step 1:

$99E22484_{16}$ =

$10011001\ 11100010\ 00100100\ 10000100_2$

Step 2:

8-bit segments: $10011001_2$, $11100010_2$, $0100100_2$, $10000100_2$

Step 3:

$$\begin{array}{rl} & 1001\ 1001_2 \\ & 1110\ 0010_2 \\ & 0010\ 0100_2 \\ + & 1000\ 0100_2 \\ \hline (10) & 0010\ 0011_2 \end{array}$$

Result of addition: $0010\ 0011_2$

Step 4:

$00100011_2 + 00000010_2 = 0010\ 0101_2$

Step 5:

8-bit checksum $= (0010\ 0101_2)' = 1101\ 1010_2$