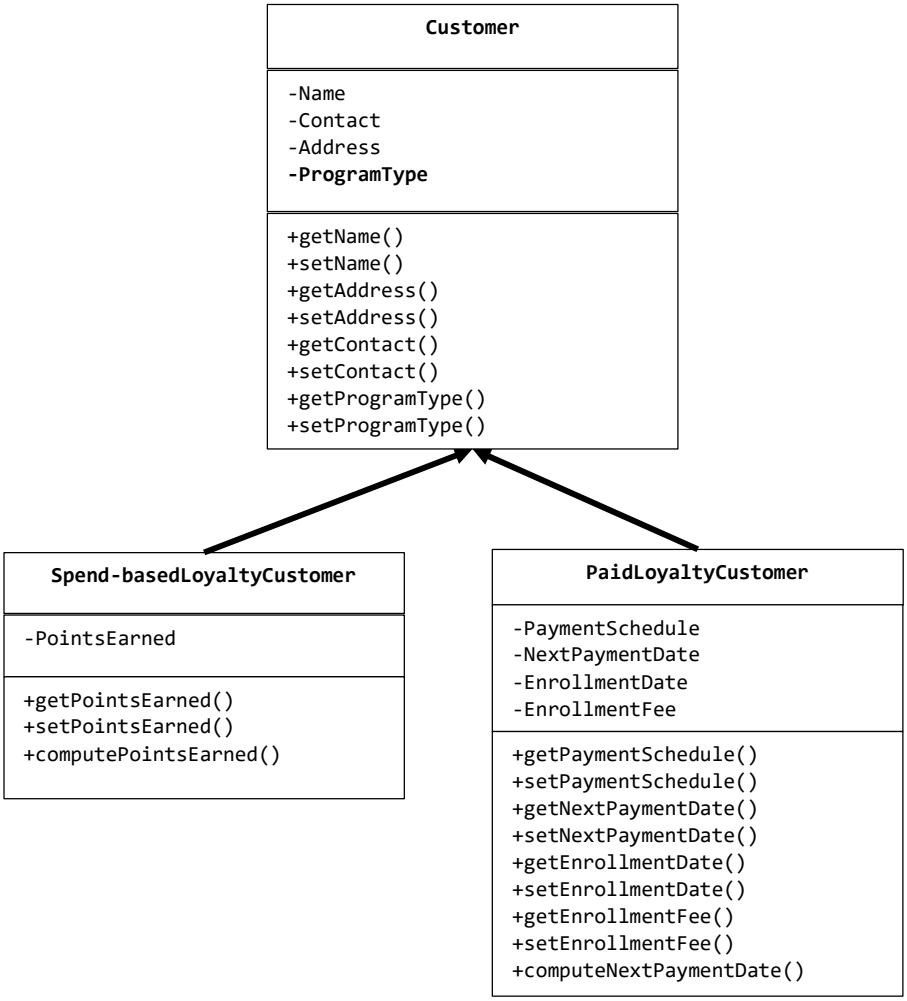
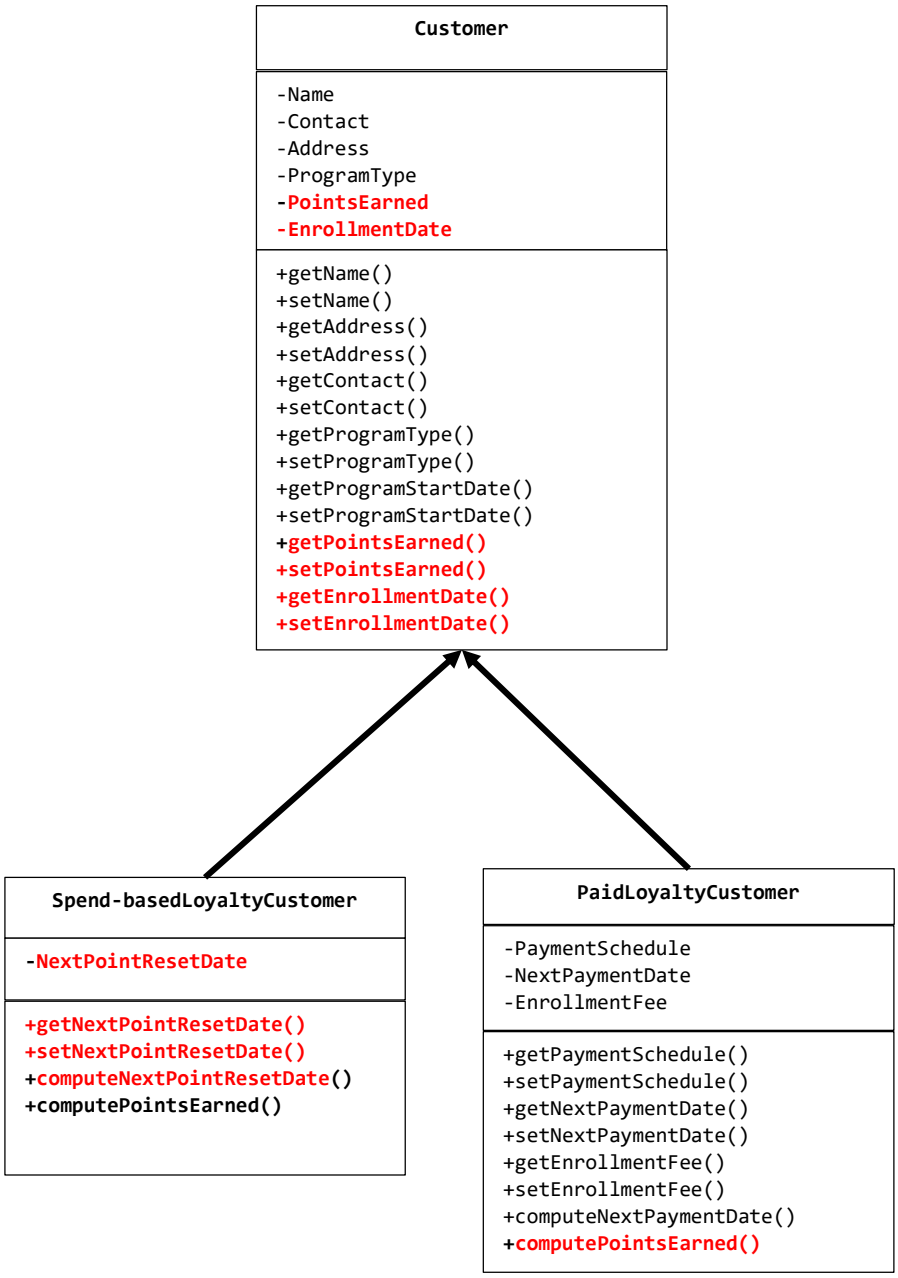
2021 JC2 Prelim Paper 1 Marking Scheme

1(a)



1(b)




1(c)Inheritance allows real world objects to be easily modeled, where some objects are specialize d versions of more general objects. Inheritance promotes code reusability as subclasses inherit all data attributes and methods of their superclasses and do not need to declare the attributes and recode the methods.

(d)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Conditions | Earned more than 2000 points in a year | Y | Y | Y | Y | N | N | N | N |
| | Enrolled for at least a year | Y | Y | N | N | Y | Y | N | N |
| | Average of at least one order per month | Y | N | Y | N | Y | N | Y | N |
| Actions | Contact by staff | | | Y | | | | | |
| | Receive invitation via email | | | | | Y | | | |
| | Contact by staff and Receive invitation via email | Y | | | | | | | |
| | No invitation | | Y | | Y | | Y | Y | Y |

(e)

Method 1

| | | | | | | |
|---|---|---|---|---|---|---|
| Conditions | Earned more than 2000 points in a year | Y | Y | N | N | - |
| | Enrolled for at least a year | Y | N | Y | N | - |
| | Average of at least one order per month | Y | Y | Y | Y | N |
| Actions | Contact by staff | | Y | | | |
| | Receive invitation via email | | | Y | | |
| | Contact by staff and Receive invitation via email | Y | | | | |
| | No invitation | | | | Y | Y |

Method 2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Conditions | Earned more than 2000 points in a year | Y | Y | N | Y | N | N |
| | Enrolled for at least a year | Y | N | Y | - | Y | N |
| | Average of at least one order per month | Y | Y | Y | N | N | - |
| Actions | Contact by staff | | Y | | | | |
| | Receive invitation via email | | | Y | | | |
| | Contact by staff and Receive invitation via email | Y | | | | | |
| | No invitation | | | | Y | Y | Y |

2 (a) Assume array index starts from 0

```
PROCEDURE Merge (A, low, mid, high)

    // create and copy data from A to temporary left subarray
    leftLen ← mid – low + 1
    FOR i = 0  TO  leftLen - 1
        left[i] ← A[low + i]
    ENDFOR

    // create and copy data from A to temporary right subarray
    rightLen ← high - mid
    FOR i = 0  TO  rightLen - 1
        right[i] ← A[(mid+1) + i]
    ENDFOR

    // merge the temp subarrays back into A[low..high]
    i ← 0        // index of the left subarray
    j ← 0        // index of the right subarray
    k ← low    // index of the merged subarray

    WHILE (i < leftLen) AND (j < rightLen)
        IF left[i] < right[j]
            A[k] ← left[i]
            i ← i + 1
        ELSE
            A[k] ← right[j]
            j ← j + 1
        ENDIF
        k ← k + 1
    ENDWHILE


    // check if any element was left
    WHILE (i < leftLen)
        A[k] ← left[i]
        i ← i + 1
        k ← k + 1
    ENDWHILE

    WHILE (j < rightLen)
        A[k] ← right[j]
        j ← j + 1
        k ← k + 1
    ENDWHILE
ENDPROCEDURE
```

(b)

- O(NlogN)
- The array of size N is divided into a max of logN parts, and the merging of all subarrays into a single array takes O(N) time, so the run time of Merge Sort algorithm is O(NLogN)
- Merge Sort always divides the array in two halves and takes linear time to merge two halves

3 (a)

```
INPUT number
hexadeimal ← "

Create(S)

WHILE number  <> 0
        remainder ← number MOD 16
        IF remainder > 9
            remainder ← ASC_CHAR(remainder + ASC_VAL('A') - 10)
        ENDIF
        Insert(S, STRING(remainder))

        number ← number DIV 16
ENDWHILE

WHILE NOT EmptyStack(S)
        hexadeimal ← hexadecimal + Retrieve(S)
ENDWHILE

OUTPUT hexadeimal
```

(b)

```
Create (R)
Insert(R,L1)
Retrieve(R)
Insert(R, L2)
Insert(R, L3)
Retrieve(R)
Retrieve(R)
```

4 (a)

- It is defined in terms of itself // it calls itself
- It has a stopping condition // base case
- It is a self-contained subroutine
- It can return data to its previous call

(b)

- Each time when the recursive call is made, an activation record is created including all parameters/local variables/return address and return value(s). This activation record is pushed onto the run-time stack.

- The top activation record in the run-time stack is always the procedure currently being executed

- When the procedure terminates/returns, its activation record is popped from the stack.

(c)

```
FUNCTION  SumDigits (N: INTEGER) RETURNS INTEGER
// returns the sum of the digits in the integer, N.

    IF N < 10
        RETURN N
    ELSE
        remainder ← N MOD 10
        N ← N DIV 10
        RETURN  remainder + SumDigits(N)
    ENDIF

 ENDFUNCTION
```

N.B. x `DIV` y gives the integral part of the quotient when x is divided by y, and x MOD y gives the remainder

5 (a)

(i) client-server network.

Disadvantage: if server is down, the whole system is down. Method: build more servers, increase the security and protection level of server

Disadvantage: server requires specialised professionals to build and maintain. Method: consistently train staff to the skills

(ii) Professional: ensure accurate and prompt updating of data

Integrity: protect the data from leaking

(b)

(i) Digital signature ensures the **security** of data and **authenticates** the sender.

**Sender Side**

- The sender uses a hash algorithm to **create a hashed version** of the message
- The sender uses its **private key** to **encrypt the hash** to the digital signature
- Both the **message** (encrypted or not) and the **digital signature** are sent to the receiver

**Receiver Side**

- The receiver uses the sender's **public key** to **decrypt** the digital signature back to the sender's version of hash
- The receiver uses the **same** hash algorithm to create a **new hash** from the received message If the two hashes **match**, it means the data is not altered and is sent by the known sender

(ii) Network protocol at the transport layer of TCP/IP model which ensures the **security and validity of data**. TCP uses a three-way-handshake to **establish a connection before data transmission**. Data is broken into segments with sequential numbers for **reassembly** at the receiver.

(iii) DNS **translates domain name** given by the user **to the IP address** that the computer can read and process. A **hierarchy** of DNS servers searches the domain name database to find the corresponding IP address.

6(i) length check/range check/format check/presence check

(ii)

data verification example: enter password twice, proof read before submitting forms

Data validation ensures the input data is sensible and reasonable; data verification ensures the input data matches the original resource

(iii) ID 598709 gives correct check digit, but the order of digits is wrong

ID 58799 gives correct check digit, but 0 is omitted

(iv) 3*6 + 0*5 + 5*4 + 2*3 + 6*2 = 56

56 mod 11 = 1

Check digit is 11 – 1 = 10, and replaced by 'X'

(v) *checking if calculated check digit matches is WRONG method

The 5 digits carry weights 6, 5, 4, 3, 2 respectively.
The check digit carries weight 1.

If the check digit is `'X'`, its numerical value is 10.

Calculate the sum of each digit multiplied by its weight

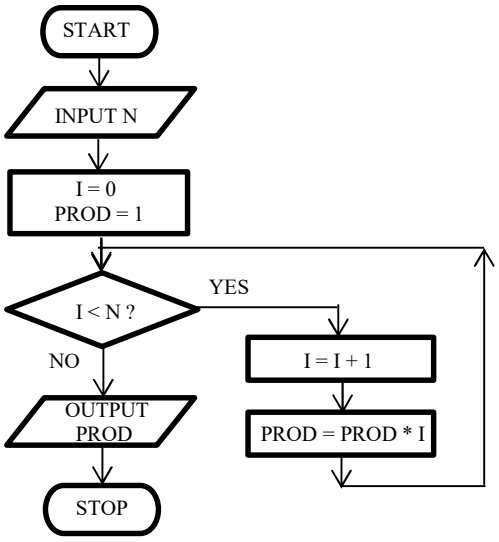
If the sum is divisible by 11, then the student ID is valid. Otherwise invalid.

(vi) STRING. Reason: check digit can be a character; 5 digits may start with 0; easy to use a for loop to read each character

7 (a)

(i) -- A flowchart is a step by step diagrammatic representation of the logic paths to solve a given problem. Or
-- A flowchart is a visual or graphical representation of an algorithm.

(ii) correct symbols logic / workflow



(b)

Assume array index starts at 1

```
noOfMoves ← 0
start ← 1
end ← 2*n

FOR i ← 1 TO n     // n passes
        FOR j ← start TO end STEP 2
                // interchange A[j] & A[j+1]
                temp ← A[j]
                A[j] ← A[j+1]
                A[j+1] ← temp

                noOfMoves ← noOFMoves + 1
        ENDFOR

        // for the next pass, consider A[start+1] to A[end-1]
        //  as A[start] and A[end] are in position
        start ← start + 1
        end ← end – 1

ENDFOR

OUTPUT A,  noOfMoves
```

8

(a) The table has a repeated group of attributes - There are several orders for the same student/StudentID

(b)

a. `StudentID` for `Student` Table
b. `MealID` for `Meal` Table
c. `StudentID` + `OrderDate` for `Order` Table

(c) Data redundancy leads to data anomalies and data corruption. Same StudentName and ClassName are being stored more than once, which will cause data anomalies during inserting, updating and deleting of data from the database.

(d)

(e) Student table.
There are transitive dependencies. `ClassName` is transitive dependent on `StudentID`.

(f)

STUDENT (StudentID, StudentName, ClassID)

Class (ClassID, ClassName)

MEAL (MealID, MealDescription, Price)

ORDER (StudentID, OrderDate, MealID)

(g)

```
SELECT STUDENT.StudentName, ORDER.OrderDate FROM STUDENT INNER
JOIN ORDER ON STUDENT.StudentID =ORDER.StudentID INNER JOIN MEAL
ON MEAL.MealID = ORDER.MealID WHERE MEAL.MealDescription =
'Japanses Bento with green tea'
```