1. The Human Resource department of an organization would like to develop a system using object-oriented approach to manage the information of the employees.

One of the functions of the system is to compute the monthly pay of the full-time employees which comprise of the monthly salary and the overtime allowance.

Due to the rapid expansion of the organization, the organization starts to employ daily-rated employee. For daily-rated employee, their monthly pay are computed based on the rate per day and the number of days worked per month.

1. Draw a class diagram which exhibits the following:

* Suitable classes with appropriate properties and methods
* Inheritance
* Polymorphism [6]

1. Explain how your design in **(a)** demonstrates code reuse. [2]
2. Explain the term **polymorphism** and how it is applied in your design in **(a)**. [2]

Solution Guide:

1(a)

[4M] 3 classes – 1 superclass, 2 subclasses with private attributes and public get/set methods for the attributes. Arrow from subclass to superclass to show

[1M] – Inheritance - Arrow from subclass to superclass to show inheritance

[1M] – Polymorphism – show method for all 3 classes

(b) The sub-classes (DailyRatedEmployee and FullTimeEmployee) inherit all the attributes and methods of the Employee class. The methods **getName, SetName, getDepartment, setDepartment, getDesignation, setDesignation are** **inherited without changes to the implementation and the no coding are required, hence code reused is achieved.** [2M]

(c) Polymorphism refers to an object’s ability to take different forms. It allows subclasses to have methods with the same name as methods in their superclasses. It gives the ability for a program to call the correct method depending on the type of object that is used to call it. [1M]

The method *show* in both the subclasses **overrides** the superclass *show*. If the subclass object (DailyRatedEmployee) is used to call *show*, then the subclass’s version of the method is invoked. If the superclass object (Employee) is used to call *show*, then the superclass method will be invoked. [1M]

1. A queue data structure is implemented using an array Queue and two pointers, Head and Tail. The space in array is fully utilized to perform the queue operations.

Queue: 1-dimensional array with index 1 to 10

Head: pointing to the index of the first item in the queue

Tail: pointing to the index of the next item that is inserted

1. Describe an algorithm, using pseudocode, to insert a new item NewItem into the queue. [4]

1. Describe an algorithm, using pseudocode, to delete an item from the queue. [6]
2. Peter intends to use the pseudocode Length 🡨 Tail – Head to find the length of the queue. Give an example to explain why he fails. Write down the correct pseudocode to find the length. [3]
3. This data structure can also be implemented using linked list. Give **one** advantage and **one** disadvantage of linked list over array implementation. [2]

Solution Guide:

2(a)

01 IF Head = Tail

02 THEN

03 OUTPUT 'No more room to add item'

04 ELSE

05 Queue[Tail] <- NewItem

06 Tail <- Tail + 1

07 IF Tail = 11

08 THEN

09 Tail <- 1

10 IF Head = 0

11 THEN

12 Head <- 1

13 ENDIF

14 ENDIF

[1] line 01 for correct condition of full queue

[1] line 05&06 for insertion

[1] line 07-10 for circular array

[1] head = 0 case

(b)

Method 1

01 IF Head = 0

02 THEN

03 OUTPUT 'Empty queue'

04 ELSE

05 IF (Tail = Head + 1) OR (HEAD = 10 AND Tail = 1)

06 # queue with one element

07 THEN

08 Head <- 0

09 Tail <- 1

10 ELSEIF HEAD = 10 #new head circulates to index 1

11 THEN

12 Head <- 1

13 ELSE

14 THEN

15 Head <- Head + 1

16 ENDIF

17 ENDIF

[1] line 01 for correct condition of empty queue

[2] line 05 for condition of one-element queue

[1] line 08&09 for reset Head and Tail

[1] line 10-12 for circular array

[1] line 15 for updating Head

Method 2

01 IF Head = 0

02 THEN

03 OUTPUT 'Empty queue'

04 ELSE

05 Head <- Head + 1

06 IF (Tail = Head) OR (HEAD = 11 AND Tail = 1)

07 # queue with one element

08 THEN

09 Head <- 0

10 Tail <- 1

11 ELSEIF HEAD = 11 #new head circulates to index 1

12 THEN

13 Head <- 1

14 ENDIF

15 ENDIF

[1] line 01 for correct condition of empty queue

[1] line 05 for updating Head

[2] line 06 for condition of one-element queue

[1] line 09&10 for reset Head and Tail

[1] line 11-13 for circular array

(c)

For example, in this case, the queue contains items in index 7, 8, 9, 10, 1, 2, so the length is 6, but Tail – Head = –4

| Index |  |
| --- | --- |
| 1 |  |
| … |  |
| 3 | Tail |
| … |  |
| 7 | Head |
| … |  |
| 10 |  |

Correct Statement:

IF Head = 0

THEN

Length <- 0

ELSEIF Head < Tail

THEN

Length <- Tail – Head

ELSE

THEN

Length <- 10 + Tail – Head

ENDIF

Each case with correct condition + calculation gives 1 mark. If students only get 2 cases correct, still give 2 marks. But tutors must explain all cases when returning the paper.

(d)

Advantage: linked list does not need to handle the circular case. Linked list provides dynamic spaces and provides bigger size for queue

Disadvantage: linked list requires more memory space to store the pointer for every item

1. (a) The following is an algorithm for an insertion sort procedure.

PROCEDURE **sort** ( A, n )

{insertion sort the array A, items 1 to n}

IF n > 1 THEN

sort ( A, n - 1 )

insert ( A, n - 1, A[ n ] )

ENDIF

ENDPROCEDURE

PROCEDURE  **insert** ( A, i, X )

{the array A has items 1 to i already sorted; insert the item X into position to make items 1 to i + 1 sorted}

IF i = 0 THEN

A[ 1 ] 🡨 X

ELSE

IF X > A[ i ] THEN

A[ i + 1 ] 🡨 X

ELSE

A[ i + 1 ] 🡨 A[ i ]

insert ( A, i - 1, X )

ENDIF

ENDIF

ENDPROCEDURE

Illustrate the operation of procedure **insert** ( A, 4, X ) where

A[ 1 ] is ‘Amy’

A[ 2 ] is ‘Ben’

A[ 3 ] is ‘Ken’

A[ 4 ] is ‘Tim’ and

X is ‘Jin’

by completing the trace table given below.

|  | A[1] | A[2] | A[3] | A[4] | A[5] | i | X |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **insert**(A,4,’Jin’) | Amy | Ben | Ken | Tim |  | 4 | Jin |
| **insert**(A,3,’Jin’) |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

[4]

1. Write an algorithm, in pseudocode, for a **non-recursive** version of the insertion sort to sort items held in an array in ascending order. [6]
2. Identify **two** features of the array that would have an impact on the performance of this insertion sort algorithm in part **(b)**. [2]
3. D. State the time complexity of the sort algorithm written in **(b)** if items in the array are initially in
4. reverse order [1]
5. sorted order [1]

What is the maximum number of comparisons needed to sort an array of N items? [1]

Solution Guide:

(a) [1] 3 calls [1] per row, total 3 marks

|  | A[1] | A[2] | A[3] | A[4] | A[5] | i | X |
| --- | --- | --- | --- | --- | --- | --- | --- |
| insert(A,4,’Jin’) | Amy | Ben | Ken | Tim |  | 4 | Jin |
| insert(A,3,’Jin’) | Amy | Ben | Ken | Tim | **Tim** | **3** | Jin |
| insert(A,2,’Jin’) | Amy | Ben | Ken | **Ken** | Tim | **2** | Jin |
|  | Amy | Ben | **Jin** | Ken | Tim | 2 | Jin |

There are no more recursive calls.

‘Jin’ has been inserted into the array A.

(b)

01 FOR i 🡨 2 TO ArraySize [1]

02 Key 🡨 Array[i]

03 j 🡨 i-1 [1]

04 WHILE (j > 0) AND (Key < Array[j]) [2]

05 Array[j+1] 🡨 Array[j] [1]

06 j 🡨 j-1

07 ENDWHILE

08 Array[j+1] 🡨 Key [1]

09 ENDFOR

(c)

* + - The size of that array [1]
    - How ordered the items already are [1]

(d) Reverse order : O(n2) [1]

Sorted order: O(n) [1]

N(N-1)/2 comparisons [1]

1. A mall operator operates 4 malls in Singapore. Due to the recent outbreak, the mall operator decides to develop a centralized system to accurately limit the number of people entering in its premises to prevent overcrowding.

In each mall, there will only be one entrance and one exit. There is a sensor at the entrance to capture the timestamp when a person enters the mall. At the exit, there is also one sensor to capture the timestamp when a person exits the mall.

1. The mall operator wants to model this system using a relational database.
   * + 1. A database needs a number of tables to store the data for this system.

Draw the Entity-Relationship (E-R) diagram to show the tables in third normal form (3NF) and their relationships between them. [4]

* + - 1. A table description can be expressed as:

TableName( Attribute1, Attribute2, Attribute3, …)

The primary key is indicated by underlining one or more attributes. Foreign keys are indicated by using a dashed underline.

Using the information given, write table descriptions for the tables you identified in **part (a) (i)**. [4]

1. State **two** reasons why the mall operator may wish to choose a NoSQL database. [2]

Solution Guide:

4(a)

There are a few possible solutions.

(i) 



[3M] 3 Tables with correct relationships.

[1M] Mall 1:M Sensor 1:M EntranceExitRecord

(ii)

Mall (MallID, MallName, MallMaxCapacity)

Sensor (SensorID, MallID, UseForEntry)

EntranceExitRecord(RecordID, SensorID, TimeStamp)

[4M] 3 tables with the columns. Each table has a primary key. MallID is a Foreign Key in Sensor. SensorID is the foreign Key in EntranceExitRecord table.

(i)



[3M] 3 Tables with correct relationships.

[1M] Mall 1:M EntranceExitRecord M:1 Sensor

(ii)

Mall (MallID, MallName, MallMaxCapacity)

Sensor (SensorID, UseForEntry)

EntranceExitRecord(RecordID, SensorID, MallID~~,~~ TimeStamp)

[4M] 3 tables with the columns. Each table has a primary key. MallID and SensorID is a Foreign Key in EntranceExitRecord table.

(b) Any 2 [2M]

1.Data storage needs to be performed quickly.

2. There will be an extremely large amount of data (i.e., Big Data).

3. Dynamic schemas (possible answer)

**--- END OF PAPER ---**