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.

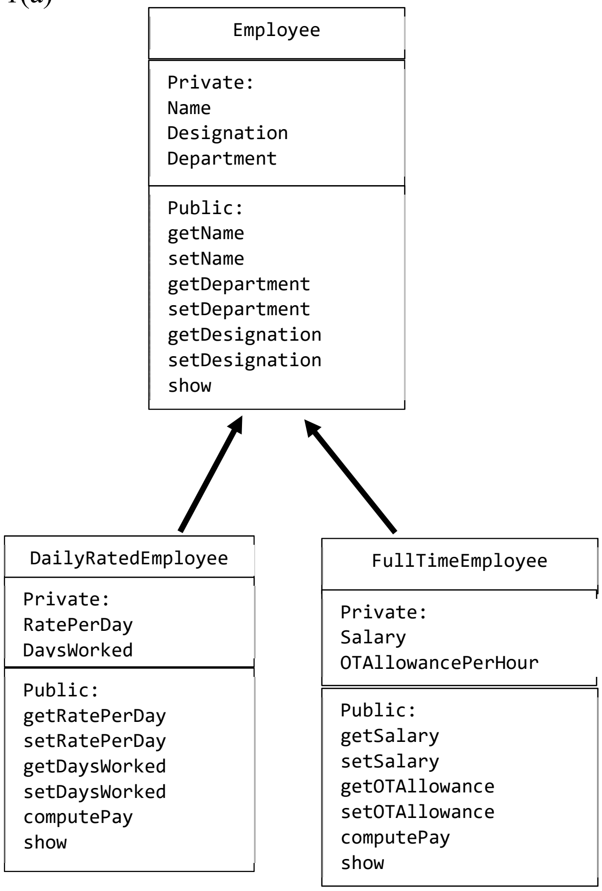
   (a) Draw a class diagram which exhibits the following:
   - Suitable classes with appropriate properties and methods
   - Inheritance
   - Polymorphism [6]

   (b) Explain how your design in **(a)** demonstrates code reuse. [2]

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

Solution Guide:

1(a)



(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.**

(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.

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.

2. 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

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

   (b) Describe an algorithm, using pseudocode, to delete an item from the queue. [6]

   (c) 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]

   (d) 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)

```
IF Head = Tail
  THEN
        OUTPUT 'No more room to add item'
  ELSE
        Queue[Tail] <- NewItem
        Tail <- Tail + 1
        IF Tail = 11
             THEN
                   Tail <- 1
        IF Head = 0
             THEN
                   Head <- 1
        ENDIF
ENDIF
```

(b)

Method 1

```
IF Head = 0
  THEN
        OUTPUT 'Empty queue'
  ELSE
        IF (Tail = Head + 1) OR (HEAD = 10 AND Tail = 1)
        # queue with one element
             THEN
                   Head <- 0
                   Tail <- 1
        ELSEIF HEAD = 10 #new head circulates to index 1
             THEN
                   Head <- 1
        ELSE
             THEN
                   Head <- Head + 1
        ENDIF
ENDIF
```

Method 2

```
IF Head = 0
  THEN
        OUTPUT 'Empty queue'
  ELSE
        Head <- Head + 1
        IF (Tail = Head) OR (HEAD = 11 AND Tail = 1)
        # queue with one element
             THEN
                   Head <- 0
                   Tail <- 1
        ELSEIF HEAD = 11 #new head circulates to index 1
             THEN
                   Head <- 1
        ENDIF
ENDIF
```

(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
```

(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

3. (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]

(b) 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]

(c) Identify **two** features of the array that would have an impact on the performance of this insertion sort algorithm in part **(b)**. [2]

(d) D. State the time complexity of the sort algorithm written in **(b)** if items in the array are initially in

(i) reverse order [1]

(ii) sorted order [1]

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

Solution Guide:

(a)

	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)

```
FOR i ← 2 TO ArraySize
   Key ← Array[i]
   j ← i-1
   WHILE (j > 0) A ND (Key < Array[j])
        Array[j+1] ← Array[j]
        j ← j-1
   ENDWHILE
   Array[j+1] ← Key
ENDFOR
```

(c)

- The size of that array
- How ordered the items already are

(d) Reverse order : O(n^2)

Sorted order: O(n)

N(N-1)/2 comparisons

4. 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.

   (a) The mall operator wants to model this system using a relational database.

   (i) 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]

   (ii) 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]

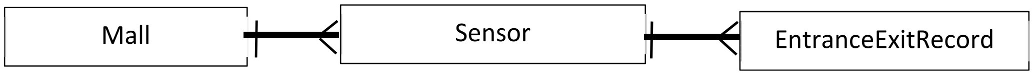
   (b) State **two** reasons why the mall operator may wish to choose a NoSQL database. [2]

Solution Guide:

4(a)

There are a few possible solutions.

(i)



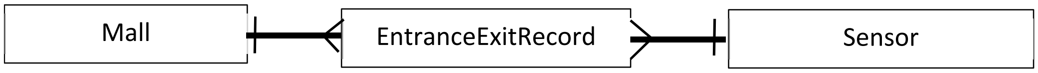
(ii)

Mall (MallID, MallName, MallMaxCapacity)

Sensor (SensorID, MallID, UseForEntry)

EntranceExitRecord(RecordID, SensorID, TimeStamp)

(i)



(ii)

Mall (MallID, MallName, MallMaxCapacity)

Sensor (SensorID, UseForEntry)

EntranceExitRecord(RecordID, SensorID, MallID,TimeStamp)

(b)

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 ---