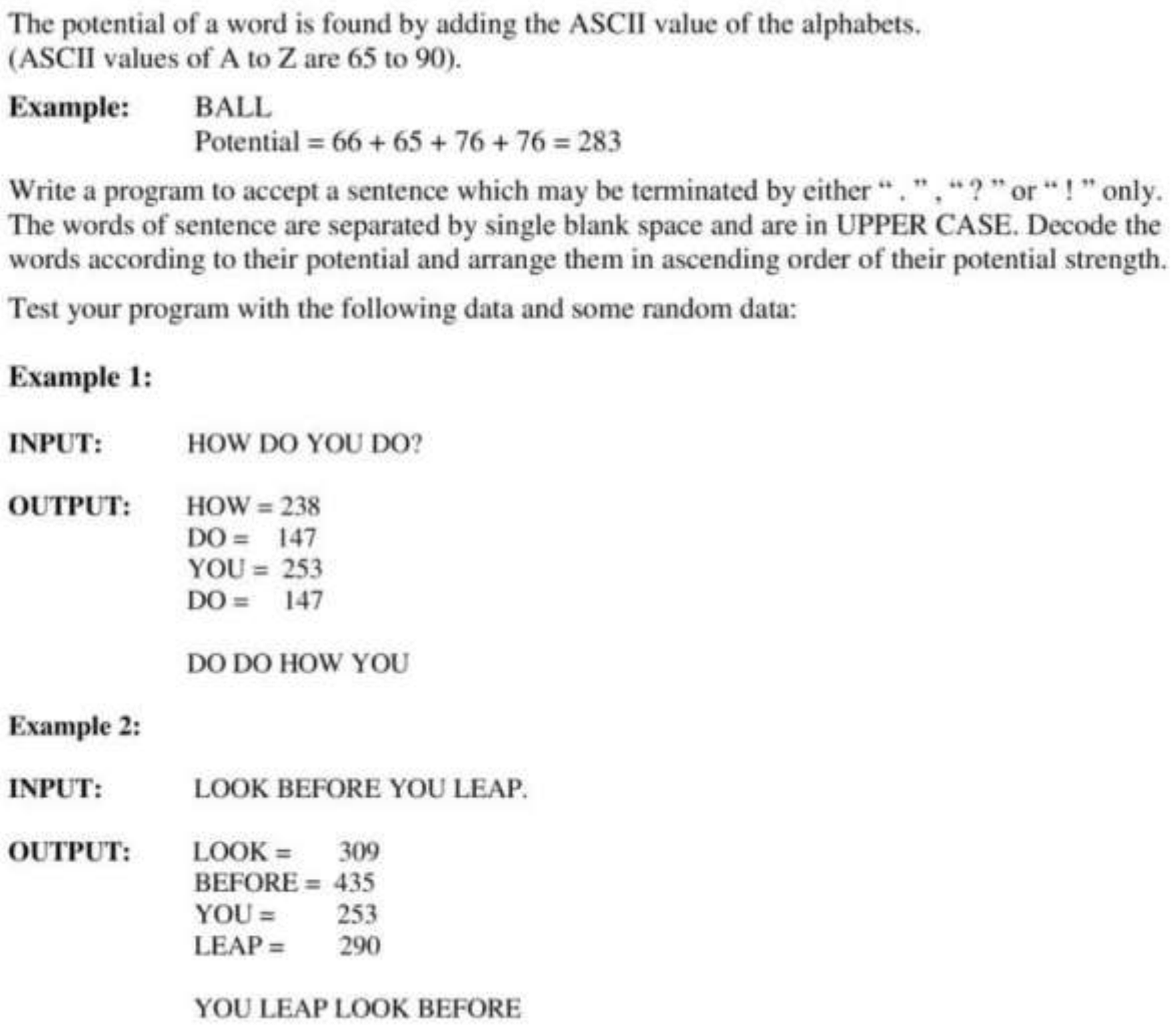
PIP ASSESSMENTS QUESTIONS IOT/MC

NOTE:-

1. INDIVIDUAL STUDENTS (NO GROUP)
2. HANDWRITTEN ONLY
3. ONLINE SUBMISSION ONLY CUIMS.
4. LAST DATE 4/4/2020 BEFORE 4:20 PM

Question 1.



The potential of a word is found by adding the ASCII value of the alphabets.
(ASCII values of A to Z are 65 to 90).

Example: BALL
Potential = 66 + 65 + 76 + 76 = 283

Write a program to accept a sentence which may be terminated by either " . " , " ? " or " ! " only. The words of sentence are separated by single blank space and are in UPPER CASE. Decode the words according to their potential and arrange them in ascending order of their potential strength.

Test your program with the following data and some random data:

Example 1:

INPUT: HOW DO YOU DO?

OUTPUT:

```
HOW = 238
DO =    147
YOU =  253
DO =    147

DO DO HOW YOU
```

Example 2:

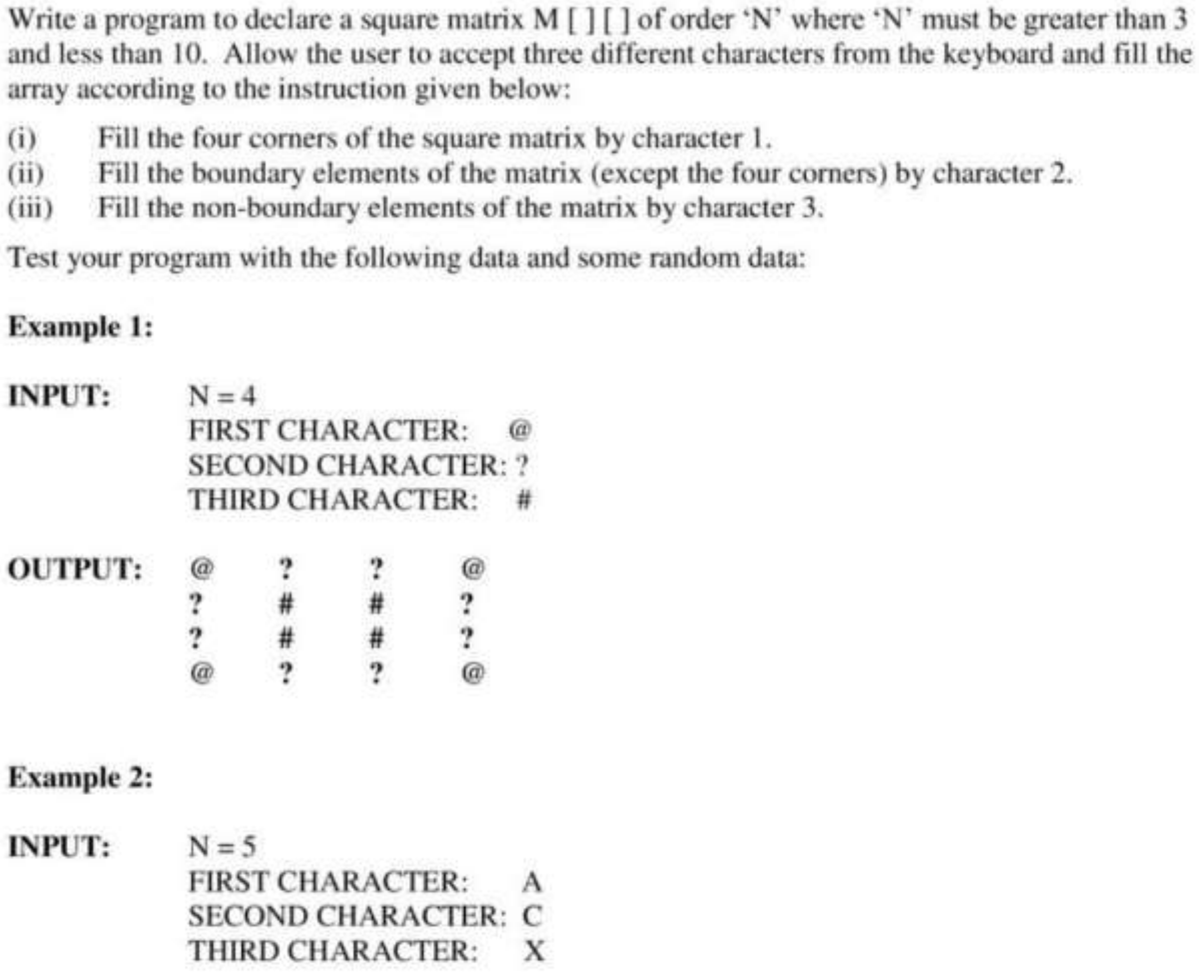
INPUT: LOOK BEFORE YOU LEAP.

OUTPUT:

```
LOOK =      309
BEFORE =  435
YOU =        253
LEAP =       290

YOU LEAP LOOK BEFORE
```

Question 2.



Write a program to declare a square matrix M [ ] [ ] of order 'N' where 'N' must be greater than 3 and less than 10. Allow the user to accept three different characters from the keyboard and fill the array according to the instruction given below:

(i) Fill the four corners of the square matrix by character 1.
(ii) Fill the boundary elements of the matrix (except the four corners) by character 2.
(iii) Fill the non-boundary elements of the matrix by character 3.

Test your program with the following data and some random data:

Example 1:

INPUT:

```
N = 4
FIRST CHARACTER:    @
SECOND CHARACTER: ?
THIRD CHARACTER:    #
```

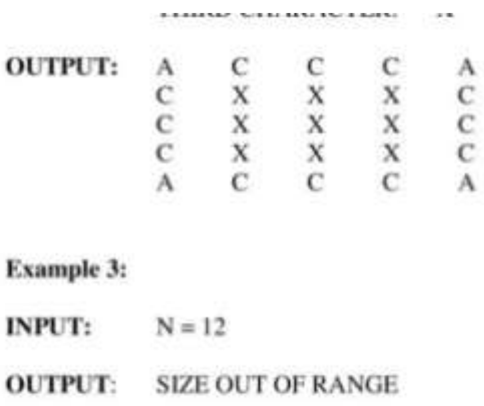
OUTPUT:

```
@    ?    ?    @
?    #    #    ?
?    #    #    ?
@    ?    ?    @
```

Example 2:

INPUT:

```
N = 5
FIRST CHARACTER:     A
SECOND CHARACTER:  C
THIRD CHARACTER:     X
```



OUTPUT:

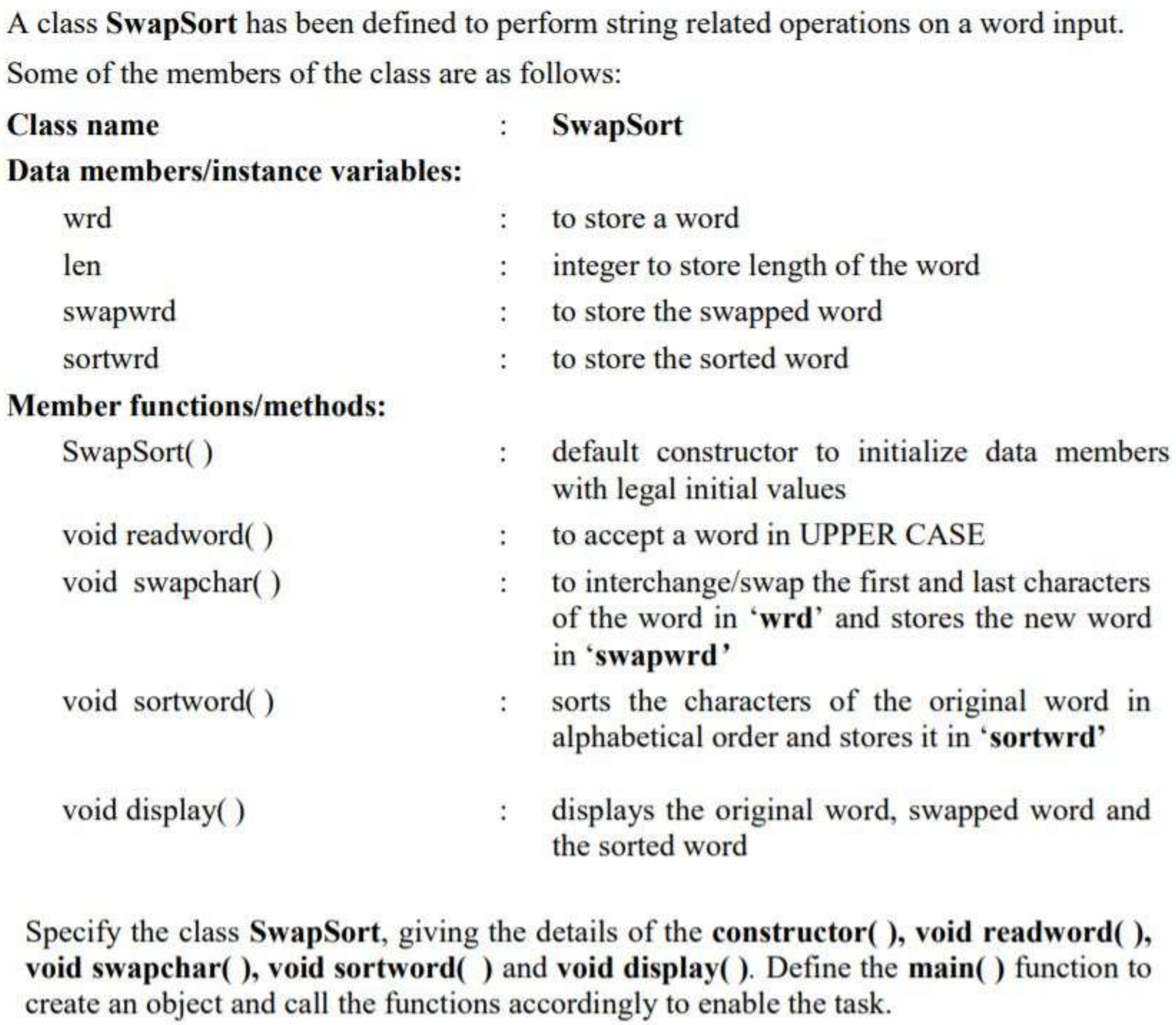
```
A    C    C    C    A
C    X    X    X    C
C    X    X    X    C
C    X    X    X    C
A    C    C    C    A
```

Example 3:

INPUT: N = 12

OUTPUT: SIZE OUT OF RANGE

Question 3.



A class **SwapSort** has been defined to perform string related operations on a word input.

Some of the members of the class are as follows:

Class name	:	**SwapSort**
Data members/instance variables:		
wrd	:	to store a word
len	:	integer to store length of the word
swapwrd	:	to store the swapped word
sortwrd	:	to store the sorted word
Member functions/methods:		
SwapSort( )	:	default constructor to initialize data members with legal initial values
void readword( )	:	to accept a word in UPPER CASE
void swapchar( )	:	to interchange/swap the first and last characters of the word in '**wrd**' and stores the new word in '**swapwrd**'
void sortword( )	:	sorts the characters of the original word in alphabetical order and stores it in '**sortwrd**'
void display( )	:	displays the original word, swapped word and the sorted word

Specify the class **SwapSort**, giving the details of the **constructor( ), void readword( ), void swapchar( ), void sortword( )** and **void display( )**. Define the **main( )** function to create an object and call the functions accordingly to enable the task.

Question4.

A class **Adder** has been defined to add any two accepted time.

Example: Time A - 6 hours 35 minutes
Time B - 7 hours 45 minutes
Their sum is - 14 hours 20 minutes (where 60 minutes = 1 hour)

The details of the members of the class are given below:

Class name : **Adder**

Data member/instance variable:

a[ ] : integer array to hold two elements (hours and minutes)

Member functions/methods:

Adder( ) : constructor to assign 0 to the array elements

void readtime( ) : to enter the elements of the array

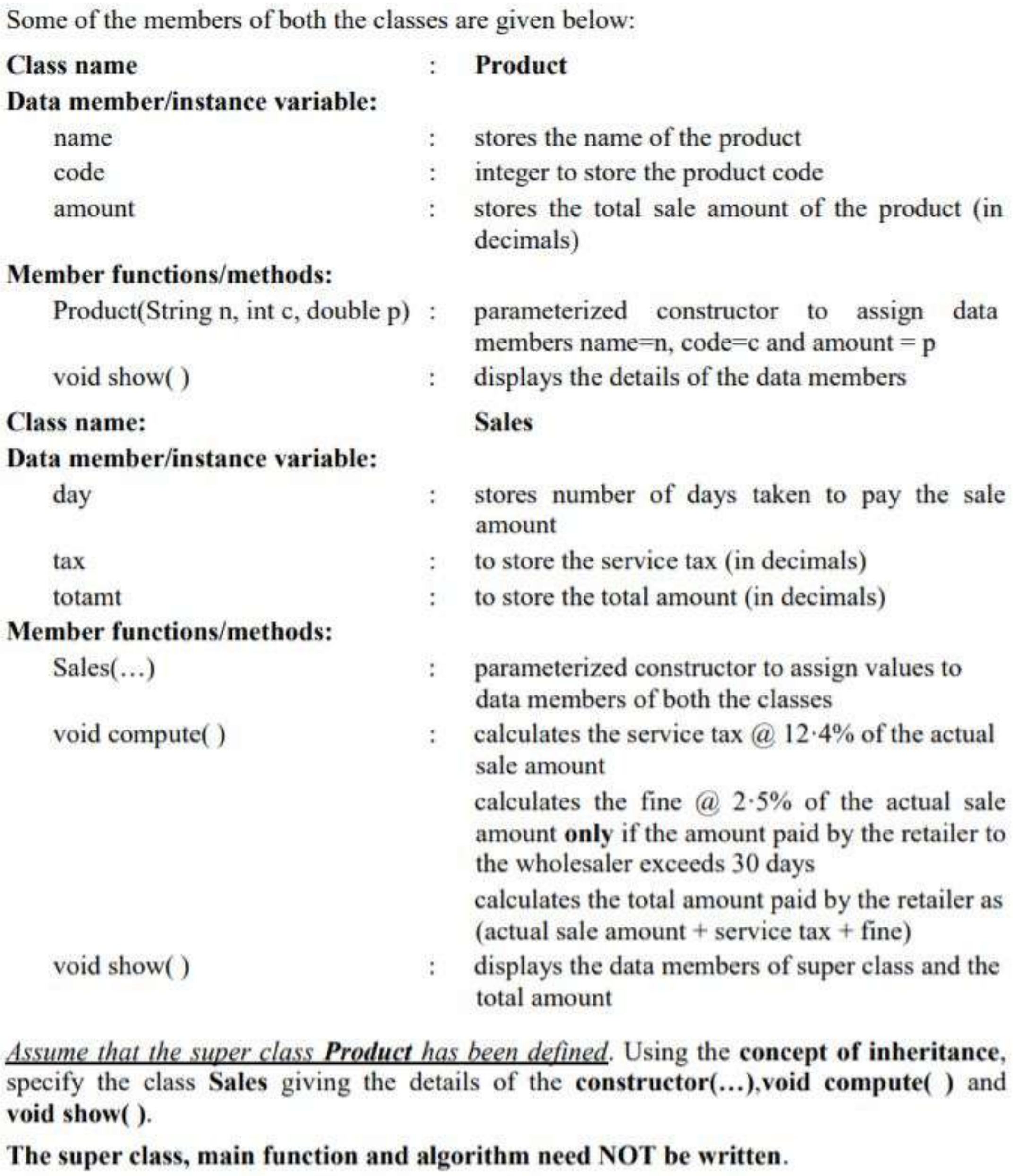
void addtime(Adder X, Adder Y) : adds the time of the two parameterized objects X and Y and stores the sum in the current calling object

void disptime( ) : displays the array elements with an appropriate message (i.e. hours = and minutes =)

Specify the class **Adder** giving details of the **constructor( ), void readtime( ), void addtime(Adder, Adder)** and **void disptime( ).** Define the **main( )** function to create objects and call the functions accordingly to enable the task.

Question 5.

A *super class* **Product** has been defined to store the details of a product sold by a wholesaler to a retailer. Define a *sub class* **Sales** to compute the total amount paid by the retailer *with* or *without fine* along with *service tax*.



Some of the members of both the classes are given below:

Class name : **Product**

Data member/instance variable:

name : stores the name of the product

code : integer to store the product code

amount : stores the total sale amount of the product (in decimals)

Member functions/methods:

Product(String n, int c, double p) : parameterized constructor to assign data members name=n, code=c and amount = p

void show( ) : displays the details of the data members

Class name: **Sales**

Data member/instance variable:

day : stores number of days taken to pay the sale amount

tax : to store the service tax (in decimals)

totamt : to store the total amount (in decimals)

Member functions/methods:

Sales(…) : parameterized constructor to assign values to data members of both the classes

void compute( ) : calculates the service tax @ 12·4% of the actual sale amount

calculates the fine @ 2·5% of the actual sale amount **only** if the amount paid by the retailer to the wholesaler exceeds 30 days

calculates the total amount paid by the retailer as (actual sale amount + service tax + fine)

void show( ) : displays the data members of super class and the total amount

Assume that the super class ***Product*** *has been defined*. Using the **concept of inheritance**, specify the class **Sales** giving the details of the **constructor(…),void compute( )** and **void show( )**.

The super class, main function and algorithm need NOT be written.

Question 6.

Queue is an entity which can hold a maximum of 100 integers. The queue enables the user to add integers from the rear and remove integers from the front.

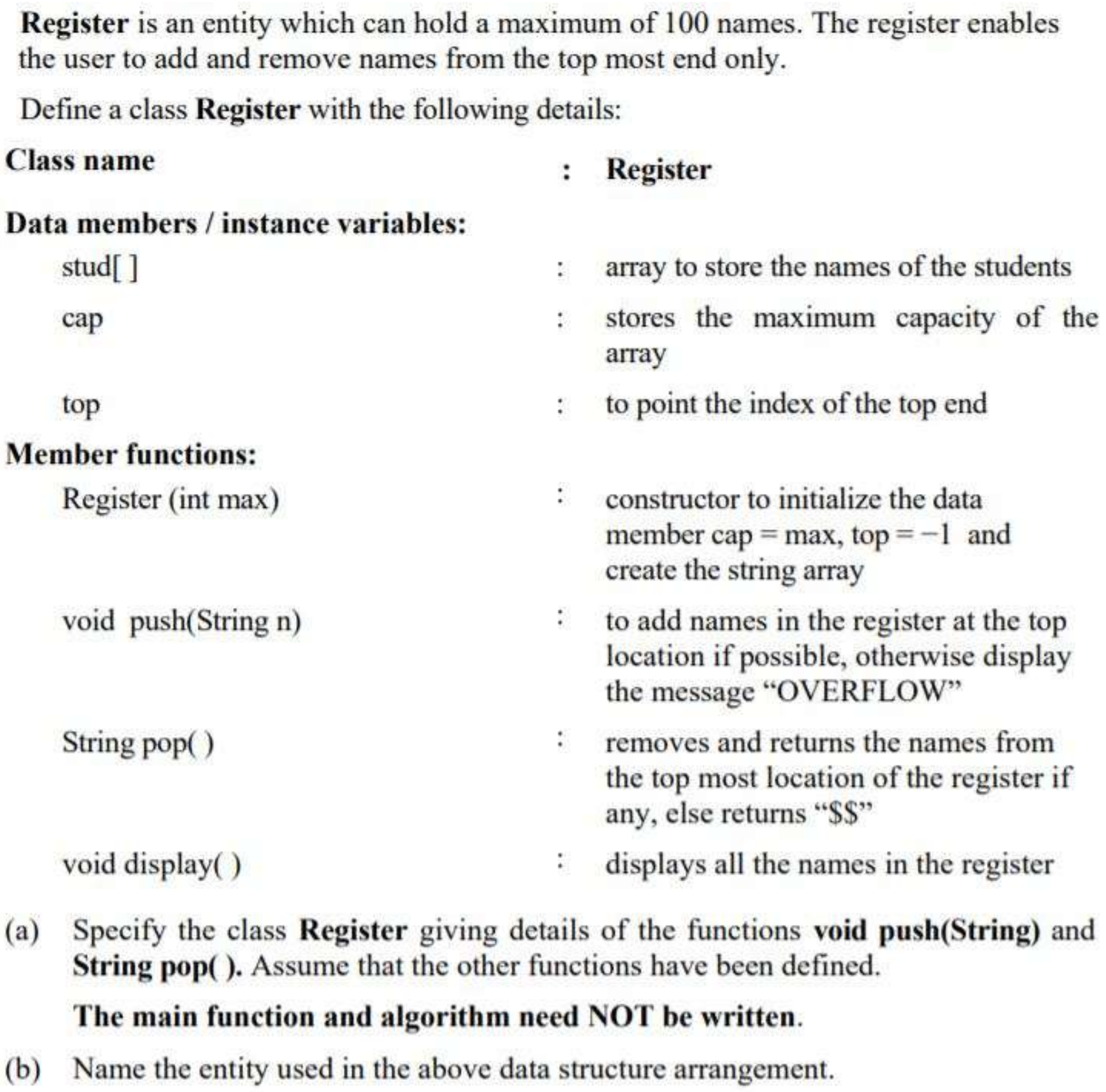
Define a class **Queue** with the following details:

Class name	**:**	**Queue**
Data Members / instance variables:		
Que[ ]	:	array to hold the integer elements
size	:	stores the size of the array
front	:	to point the index of the front
rear	:	to point the index of the rear
Member functions:		
Queue (int mm)		constructor to initialize the data size = mm, front = 0, rear = 0
void addele(int v)	:	to add integer from the rear if possible else display the message "**Overflow**"
int delele( )	:	returns elements from front if present, otherwise displays the message "**Underflow**" and return -9999
void display ( )	:	displays the array elements

Specify the class **Queue** giving details of **ONLY** the functions **void addele(int)** and **int delele( )**. Assume that the other functions have been defined.

The main function and algorithm need NOT be written.

Question 7.



Register is an entity which can hold a maximum of 100 names. The register enables the user to add and remove names from the top most end only.

Define a class **Register** with the following details:

Class name : **Register**

Data members / instance variables:

stud[ ] : array to store the names of the students

cap : stores the maximum capacity of the array

top : to point the index of the top end

Member functions:

Register (int max) : constructor to initialize the data member cap = max, top = −1 and create the string array

void push(String n) : to add names in the register at the top location if possible, otherwise display the message "OVERFLOW"

String pop( ) : removes and returns the names from the top most location of the register if any, else returns "$$"

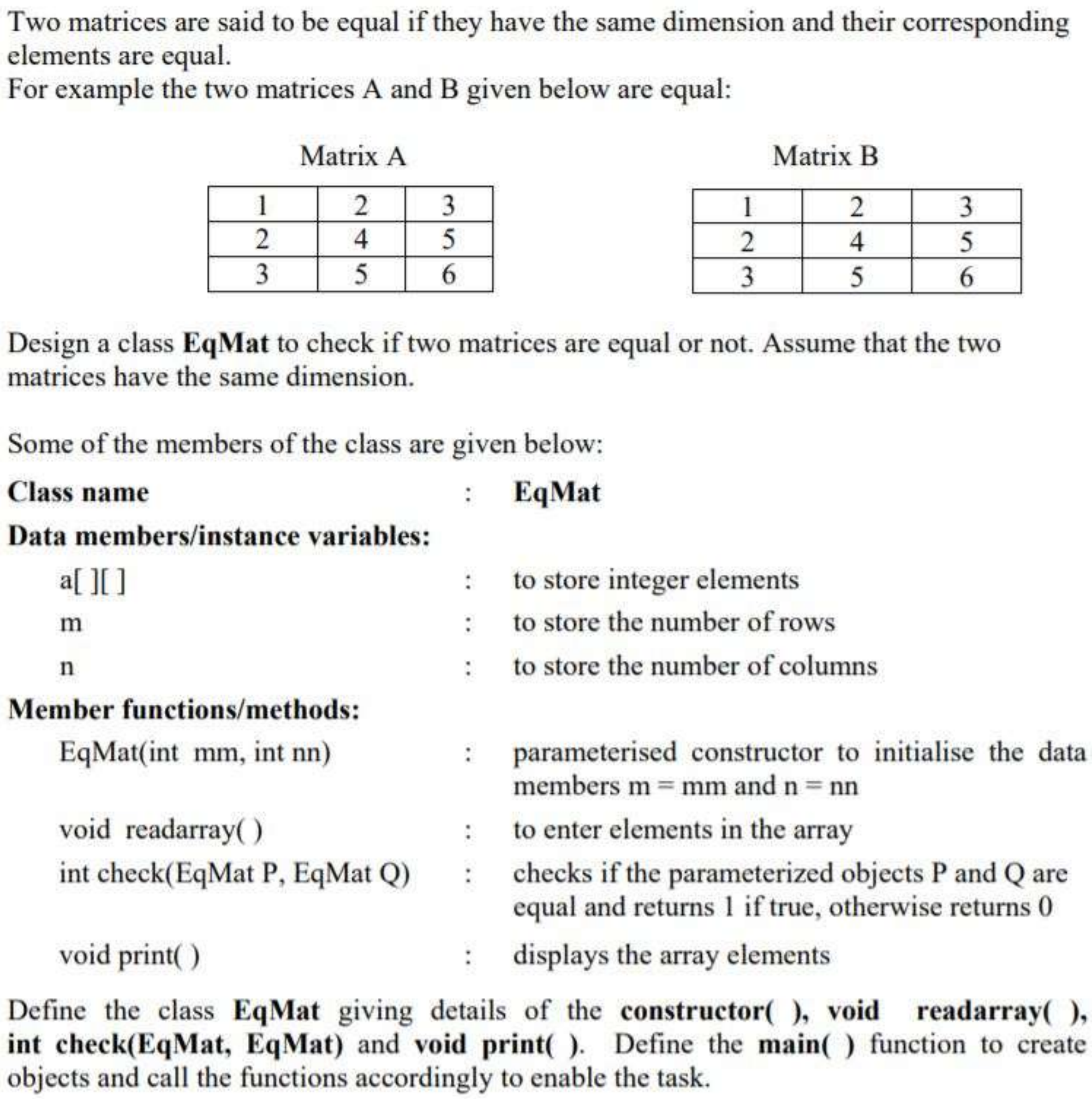
void display( ) : displays all the names in the register

(a) Specify the class **Register** giving details of the functions **void push(String)** and **String pop( ).** Assume that the other functions have been defined.

The main function and algorithm need NOT be written.

(b) Name the entity used in the above data structure arrangement.

Question 8.



Two matrices are said to be equal if they have the same dimension and their corresponding elements are equal.
For example the two matrices A and B given below are equal:

Matrix A

1	2	3
2	4	5
3	5	6

Matrix B

1	2	3
2	4	5
3	5	6

Design a class **EqMat** to check if two matrices are equal or not. Assume that the two matrices have the same dimension.

Some of the members of the class are given below:

Class name : **EqMat**

Data members/instance variables:

a[ ][ ] : to store integer elements

m : to store the number of rows

n : to store the number of columns

Member functions/methods:

EqMat(int mm, int nn) : parameterised constructor to initialise the data members m = mm and n = nn

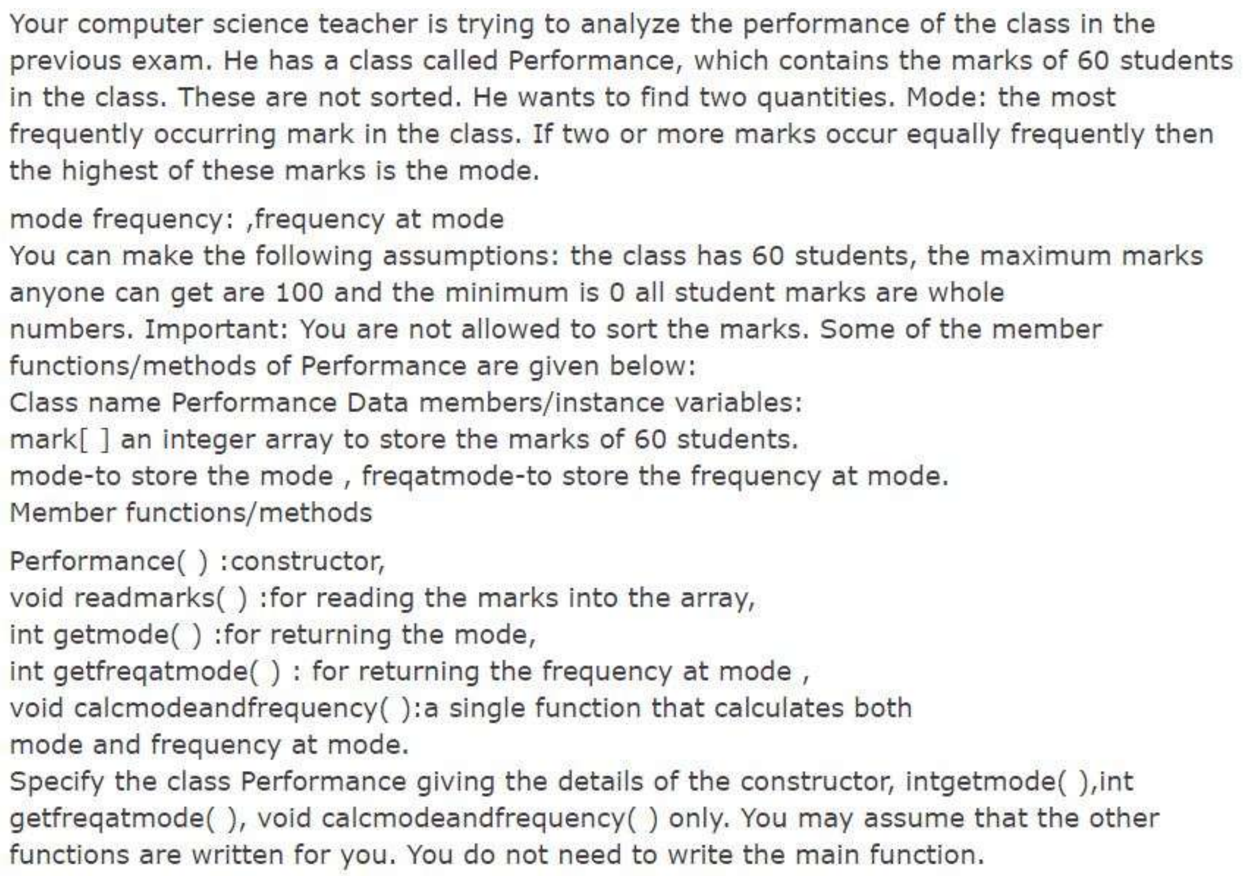
void readarray( ) : to enter elements in the array

int check(EqMat P, EqMat Q) : checks if the parameterized objects P and Q are equal and returns 1 if true, otherwise returns 0

void print( ) : displays the array elements

Define the class **EqMat** giving details of the **constructor( ), void readarray( ), int check(EqMat, EqMat)** and **void print( )**. Define the **main( )** function to create objects and call the functions accordingly to enable the task.

Question 9.

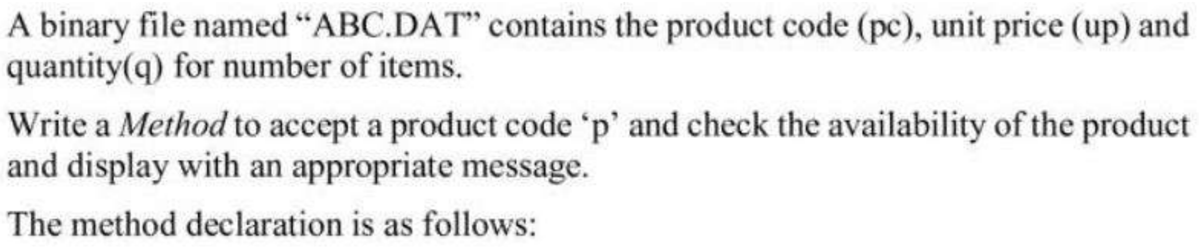


Your computer science teacher is trying to analyze the performance of the class in the previous exam. He has a class called Performance, which contains the marks of 60 students in the class. These are not sorted. He wants to find two quantities. Mode: the most frequently occurring mark in the class. If two or more marks occur equally frequently then the highest of these marks is the mode.

mode frequency: ,frequency at mode
You can make the following assumptions: the class has 60 students, the maximum marks anyone can get are 100 and the minimum is 0 all student marks are whole numbers. Important: You are not allowed to sort the marks. Some of the member functions/methods of Performance are given below:
Class name Performance Data members/instance variables:
mark[ ] an integer array to store the marks of 60 students.
mode-to store the mode , freqatmode-to store the frequency at mode.
Member functions/methods

Performance( ) :constructor,
void readmarks( ) :for reading the marks into the array,
int getmode( ) :for returning the mode,
int getfreqatmode( ) : for returning the frequency at mode ,
void calcmodeandfrequency( ):a single function that calculates both mode and frequency at mode.
Specify the class Performance giving the details of the constructor, intgetmode( ),int getfreqatmode( ), void calcmodeandfrequency( ) only. You may assume that the other functions are written for you. You do not need to write the main function.

Question 10



A binary file named "ABC.DAT" contains the product code (pc), unit price (up) and quantity(q) for number of items.

Write a *Method* to accept a product code 'p' and check the availability of the product and display with an appropriate message.

The method declaration is as follows: