**COMPUTER APPLICATION PROJECT- CLASS XII**

**SESSION 2018-19**

**CLASS BASED**:

1. A class ***Collection*** contains an array of 100 integers. Using the following class description, create an array with common elements from two integer arrays. Some of the members of the class are given below:

**Class name : Collection**

**Data members :**

ar[] : integer array of 100 elements

len : length of the array

**Member methods :**

Collection() : default constructor

Collection(int ) : parameterized constructor to assign the length of

the array

void input() : reads array elements

Collection common(Collection): return the **Collection** containing the common

Elements of current **Collection** and the collection

object passed as parameter.

void display() : displays the array **Collection** elements

Write the **main()** method to generate the necessary output.

2. A Transpose of an array is obtained by interchanging the elements of the rows and columns. A class Transarray contains a two dimensional integer array of order [m x n]. The maximum value possible for both `m’ and `n’ is 20. Design a class Transarray to find the transpose of a given matrix. The details of the members of the class are given below

|  |  |  |
| --- | --- | --- |
| Class name | : | Transarray |
| **Data members/instance variables** | **:** |  |
| arr[] | : | stores the matrix elements |
| M | : | integer to store the number of rows |
| N | : | integer to store the number of columns |
| **Member functions** | **:** |  |
| Transarray() | : | default constructor |
| Transarray(int mm, int nn) | : | to initialize the size of the matrix, m=mm, n=nn |
| void fillarray() | : | to enter the elements of the matrix |
| void transpose(Transarray A) | : | to find the transpose of a given matrix |
| void disparrary() | : | displays the array in a matrix form |

Specify the class Transarray giving the details of the constructors, void fillarray(), void transpose(Transarray) and void disparray(). You need not write the main function.

3. A class ***Array*** is declared as follows:

**Class name : Array**

**Data members :**

ar[] : integer array of size 100

n : size of the array

num : integer variable

**Member methods :**

Array() : a constructor to store 0 at each location of ***ar[]*** and to ***n***.

void getArray() : to input n integers in array

int process(Array B, int k): to assign ***k*** to ***num*** and return the frequency of num from

the array ***ar[]*** from the array object ***B***.

Array merge(Array a) : merge the current array with the parameterised array.

void display() : to print the array elements in such a way that only **4** elements

of array should be printed in one line with two spaces in

between the numbers.

Write **main()** method and call the above methods to generate the output.

4. A class Time is declared as follows:

**Class name : Time**

**Data members :**

hrs, min : integers to store hour and minute

**Methods :**

Time() : default constructor to assign 0 to data members

void getTime(int h, int m): assigns hours and minutes

void printTime() : prints time in hh:mm format

Time sumTime(Time t1, Time t2): adds two time t1 with t2 and return them.

Implement the above class and write **main()** method and call the above methods to generate the output.

5. Class **Matrix** contains a two dimensional integer array of order [ m x n ]. The maximum value possible for both m and n is 25. Design a class Matrix to find the difference of the two matrices. The details of the members of the class are given below:

**Class name : Matrix**

**Data members**

arr[][] : stores the matrix element

m : integer to store the number of rows

n : integer to store the number of columns

**Member functions:**

Matrix(int mm, int nn) : to initialize the size of the matrix m=mm and n=nn

void fillarray() : to enter the elements of the matrix

Matrix SubMat(Matrix A) : subtract the current object from the matrix of 6 parameterized object and return the resulting object.

void display() : display the matrix elements.

Specify the class Matrix giving details of the constructor(int,int), void fillarray(),Matrix

SubMat(Matrix) and void display(). Define the main() function to create an object and call the functions accordingly to enable the task.

**DIMENSIONAL ARRAYS:**

#### 6.

**Write a program to declare a square matrix A[ ] [ ] of order (M x M) where ‘M’ is the number of rows and the number of columns such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into this matrix. Perform the following tasks:**

**(a)        Display the original matrix.  
(b)        Check if the given matrix is Symmetric or not.  
           A square matrix is said to be Symmetric, if the element of the ith row and jth column is equal to the element of the jth row and ith column.  
(c)        Find the sum of the elements of left diagonal and the sum of the elements of right diagonal of the matrix and display them.**

**Test your program with the sample data and some random data:**

**Example 1**

**INPUT           :           M = 3**

**1       2      3  
2       4      5  
3       5      6**

**OUTPUT       :**

**ORIGINAL MATRIX**

**1       2      3  
2       4      5  
3       5      6**

**THE GIVEN MATRIX IS SYMMETRIC  
The sum of the left diagonal = 11  
The sum of the right diagonal = 10**

**Example 2**

**INPUT           :           M = 4**

**7       8      9      2  
4       5      6      3  
8       5      3      1  
7       6      4      2**

**OUTPUT       :**

**ORIGINAL MATRIX**

**7       8      9      2  
4       5      6      3  
8       5      3      1  
7       6      4      2**

**THE GIVEN MATRIX IS NOT SYMMETRIC  
The sum of the left diagonal = 17  
The sum of the right diagonal = 20**

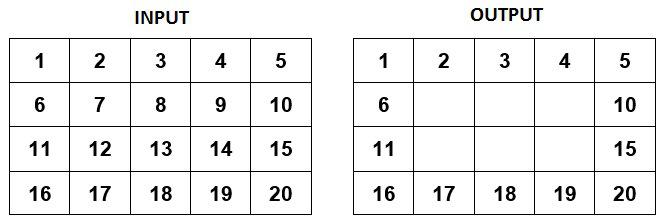
**Example 3**

**INPUT           :           M = 22**

**OUTPUT       :           THE MATRIX SIZE IS OUT OF RANGE**

#### 7. Write a Program in Java to input a 2-D array of size ‘m\*n’ and print its boundary (border) elements.

**For example:**

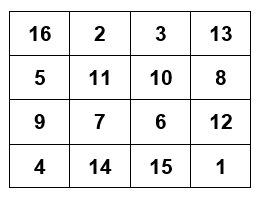
****

8. **Write a program to check if the given matrix is a Magic Square Matrix or not.**

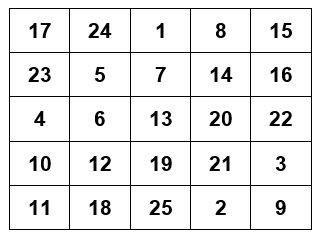
**A square matrix is said to be a Magic Square, if the sum of each row, each column and each diagonal is same. Write a program to enter an integer number ‘n’. Create a magic square of size ‘n\*n’. Finally, print the elements of the matrix as Magic Square.**

**Note: n <= 5**

**Sample Input: Enter the size of the matrix : 4  
Sample Output: The Magic Matrix of size 4×4 is:**

****

**Sample Input: Enter the size of the matrix : 5  
Sample Output: The Magic Matrix of size 5×5 is:**

****

9. **Write a program to declare a square matrix A[ ][ ] of order ‘n’. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:**

**(i) Output the original matrix.  
(ii) Find the SADDLE POINT for the matrix. If the matrix has no saddle point, output the message “NO SADDLE POINT”.**

**[Note: A saddle point is an element of the matrix such that it is the minimum element for the row to which it belongs and the maximum element for the column to which it belongs. Saddle point for a given matrix is always unique.]**

**Example: In the Matrix  
4 5 6  
7 8 9  
5 1 3**

**Saddle point = 7 because it is the minimum element of row 2 and maximum element of column 1**

10. **Write a Program in Java to input a 2-D square matrix and check whether it is an Upper Triangular Matrix or not.**

**Upper Triangular Matrix : An Upper Triangular matrix is a square matrix in which all the entries below the main diagonal () are zero. The entries above or on the main diagonal themselves may or may not be zero.**

**Example:**

****

**STRING BASED:**

**11. A sequence of Fibonacci Strings is generated as follows:**

**S0 = “a”, S1 = “b”, Sn = S(n-1) + S(n-2) where ‘+’ denotes concatenation. Thus the sequence is:**

**a, b, ba, bab, babba, babbabab, ………. n terms.**

**Design a class FiboString to generate Fibonacci strings. Some of the members of the class are given below:**

**Class name         :               FiboString**

**Data members/instance variables:**

**x                           :               to store the first string  
y                           :               to store the second string  
z                           :               to store the concatenation of the previous two strings  
n                          :               to store the number of terms**

**Member functions/methods:**

**FiboString()        :               constructor to assign x=”a”, y=”b”, z=”ba”  
void accept()      :               to accept the number of terms ‘n’  
void generate()  :               to generate and print the fibonacci strings. The sum of (‘+’ i.e. concatenation) first two strings is the third string. Eg. “a” is first string, “b” is second string then the third string will be “ba” and fourth will be “bab” and so on.**

**Specify the class FiboString, giving details of the constructor(), void accept() and void generate(). Define the main() function to create an object and call the functions accordingly to enable the task.**

**12. Write a program to accept a sentence which may be terminated by either ‘.’ ‘?’ or ‘!’ only. Any other character may be ignored. The words may be separated by more than one blank space and are in UPPER CASE.**

**Perform the following tasks:**

**(a)        Accept the sentence and reduce all the extra blank space between two words to a single blank space.  
(b)        Accept a word from the user which is part of the sentence along with its  
position number and delete the word and display the sentence.**

**Test your program with the sample data and some random data:**

**Example 1**

**INPUT:          A    MORNING WALK IS A IS BLESSING FOR   THE  WHOLE DAY.**

**WORD TO BE DELETED: IS  
WORD POSITION IN THE SENTENCE: 6**

**OUTPUT:      A MORNING WALK IS A BLESSING FOR THE WHOLE DAY.**

**Example 2**

**INPUT:          AS YOU    SOW, SO   SO YOU REAP.**

**WORD TO BE DELETED: SO  
WORD POSITION IN THE SENTENCE: 4**

**OUTPUT:      AS YOU SOW, SO YOU REAP.**

**Example 3**

**INPUT:          STUDY WELL ##.**

**OUTPUT:      INVALID INPUT.**

**13. A simple encryption system uses a shifting process to hide a message. The value of the shift can be in the range 1 to 26. For example a shift of 7 means that A = U, B =V,C = W, etc.i e.**

**Text : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
Code: U V W X Y Z A B C D E F G H I J K L M N O P Q R S T**

**Fist an extra space is added to the end of the string. To make things little more difficult, spaces within the original text are replaced with QQ before the text is encrypted. Double Q (QQ) was selected because no English word ends in Q or contains QQ.**

**Additionally the coded message is printed in blocks of six characters separated by spaces. The last block might not contain six characters. Write a program that takes the coded text (less than 100 characters), the shift value and prints the decoded original text. Your program must reject any non-valid value for shift and display an error message “INVALID SHIFT VALUE)”. Assume all characters are upper case. Test your program for the following data and some data that you have coded, using the rules given above:**

**SAMPLE DATA:**

**1. INPUT:  
CODED TEXT : “UHINBY LKKQCH HYLKK”  
SHIFT : 7  
OUTPUT:  
DECODED TEXT : ANOTHER VALUE**

**2. INPUT:  
CODED TEXT : “RUIJGG EVGGBK SAGG”  
SHIFT : 11  
OUTPUT:  
DECODED TEST : BEST OF LUCK**

**3. INPUT:  
CODED TEXT : “DKSMMW NAMMUK QMM”  
SHIFT : 29  
OUTPUT:  
INVALID SHIFT VAULE**

**14. Write a program to input a word from the user and remove the consecutive repeated characters by replacing the sequence of repeated characters by its single occurrence.**

**Example:**

**INPUT – Jaaavvvvvvvvaaaaaaaaaaa  
OUTPUT – Java**

**INPUT – Heeeiiiissggoiinggg  
OUTPUT – Heisgoing**

**15. Write a program to input a string (word). Convert it into lowercase letters. Count and print the frequency of each alphabet present in the string. The output should be given as:  
Sample Input: Alphabets  
Sample Output:  
==========================  
Alphabet             Frequency  
==========================  
a                              2  
b                              1  
e                              1  
h                              1  
l                               1  
p                             1  
s                              1  
t                               1**

**DATA STRUCTURE:**

**16. Write a program to implement Stack with pop(), push() and print() methods.**

**17. Write a program to implement LinearQ with insert(), delete() and print() methods.**

**18. Write a program to implement Dequeue with insertfront(), insertRear(), deleteFront(), deleteRear() and print() methods.**

**19. Write a program to implement CircularQ with insert(), delete() and print() methods.**

**20. Write a program to implement LinkedList using insertBeg(),insertMid(int), insertLast(), deleteBeg(), deleteMid(int), deleteLast() and print() methods.**

**INHERITANCE:**

**21.** A class **Employee** contains employee details and another class Salary calculates the employee’s netsalary. The details of the two classes are given below:

**Class name** : Employee

**Data members :**

empNo : stores the employee number.

empName : stores the employee name

empDesig : stores the employee’s designation.

**Member functions:**

Employee() : default constructor.

Employee(…) : parameterized constructor to assign values to data members.

void display() : display the employee details.

**Class name : Salary**

Data members :

basic : float variable to store the basic pay.

Member functions :

Salary(…) : parameterized constructor to assign values to data members.

void calculate() : calculates the employee’s net salary according to the

following rules:

DA = 10% of basic

HRA = 15% of basic

Salary = basic + DA +H RA

PF= 8 % of Salary

Net Salary = Salary –PF

Display the employee details and the Net salary.

Specify the class Employee giving details of the constructors and member function void display(). Using the concept of inheritance specify the class Salary giving details of the constructor and the member function void calculate(). The main function need not be written.

**22.** A super class Worker has been defined to store the details of a worker. Define a sub class Wages to compute the monthly wages for the worker. The details of both the classes are given below:

**Class name : Worker**

Data members :

name : to store the name of the worker

basic : to store the basic pay in decimal

Member functions :

Worker(….) : parameterized constructor to assign values to the instance

variables

void display() : display worker details

**Class name : Wages**

Data members :

hrs : stores the hours worked

rate : stores rate per hour

wage : stores the overall wage of the worker

**Member functions**

Wages(….) : parameterized constructor to assign values to the instance

variables of both classes

double overtime( ) : calculates and returns the overtime amount as (hours \* rate )

void display() : calculates the wage using the formula

wage=overtime amount + basic pay and displays it along with

other details

Specify the class Worker giving details of the constructor() and void display(). Using the concept of inheritance, specify the class Wages giving details of the constructor(), double overtime() and void display(). The main function need not be written.

**23.** A super class **Record** has been defined to store the names and ranks of 50 students. Define a sub class **Rank** to find the highest rank along with the name. The details of both classes are given below:

**Class name : Record**

**Data members**

name[] : to store the names of students

rnk[] : to store the ranks of students

**Member functions :**

Record() : constructor to initialize data members

void readvalues() : to store names and ranks

void display() : displays the names and the corresponding ranks

**Class name : Rank**

**Data members**

index : integer to store the index of the topmost rank

**Member functions**

Rank() : constructor to invoke the base class constructor and to

initialize index to 0.

void highest() : finds the index location of the topmost rank and stores it in

index without sorting the array

void display() : displays the name and ranks along with the name having the

topmost rank.

Specify the class Record giving details of the constructor(), void readvalues(), void display(). Using the concept of inheritance, specify the class Rank giving details of constructor(), void highest() and void display(). The main function and algorithm need not be written.

24. A super class Detail has been defined to store the details of a customer. Define a sub class Bill to compute the monthly telephone charge of the customer as per the chart given below:

**NUMBER OF CALLS RATE**

1- 100 Only rental charge

101-200 60 paisa per call + rental charge

201-300 80 paisa per call + rental charge

Above 300 1 rupee per call + rental charge

The details of both the classes are given below:

**Class Name : Detail**

**Data members**

name : to store the name of the customer.

address : to store the address of the customer.

telno : to store the phone number of the customer.

rent : to store the monthly rental charge

**Member functions :**

Detail(..) : parameterized constructor to assign values to data members.

void show() : to display the detail of the customer.

**Class Name : Bill**

**Data members**

n : to store the number of calls.

amt : to store the amount to be paid by the customer.

**Member functions :**

Bill(..) : parameterized constructor to assign values to data members of

both classes and to initialize amt = 0.0.

void cal() : calculates the monthly telephone charge as per the charge

given above.

void show() : to display the detail of the customer and amount to be paid.

Specify the class Detail giving details of the constructor( ) and void show(). Uisng the concept of inheritance, specify the class Bill giving details of the constructor( ), void cal() and void show().

**25.** A super class Perimeter has been defined to calculate the perimeter of a parallelogram. Define a sub class Area to compute the area of the parallelogram by using the required data members of the super class. The details are given below:

**Class name : Perimeter**

**Data Members**

a : to store the length in decimal

b : to store the breadth in decimal

**Member functions :**

Perimeter(…) : parameterized constructor to assign values to data members

double Calculate() : calculate and return the perimeter of a parallelogram as

2 \*(length +breadth)

void show() : to display the data members along with the perimeter of the

parallelogram.

**Class name : Area**

**Data Members**

h : to store the height in decimal

b : to store the area of the parallologram

**Member functions :**

Area(…) : parameterized constructor to assign values to data members of

both classes.

void doArea() : computes the area( breadth \* height).

void show() : to display the data members of both the classes along with the

area and perimeter of the parallelogram.

Specify the class Perimeter giving details of the constructor(…), double Calculate() and void show(). Using the concept of inheritance, specify the class Area giving details of the constructor(…), void doArea() and void show(). The main function and algorithm need not be written.

**RECURSION:**

26. A class **SeriesSum** is designed to calculate the sum of the following series:-

Sum= x2/1! + x4/3!+x6/5!+….xn/(n-1)!

Some of the members of the class are given below:

Class name : SeriesSum

**Data members**

x : to store an integer number

n : to store number of terms

sum : double variable to store the sum of the series

**Member functions :**

SeriesSum(intxx,intnn) : constructor to assign x=xx, n=nn

doublefindfact(int m) : to return the factorial of m using recursive technique

void calculate() : to calculate the sum of the series by invoking the recursive

functions repeatedly

void display() : to display the sum of the series

Specify the class SeriesSum, giving details of the constructor(int,int), double findfact(int), double findpower(int,int), void calculate() and void display(). Define the main() function to create an object and call the functions accordingly to enable the task.

27. A class **RecFact** defines a recursive function to find the factorial of a number. The details of the class are given below:

**Class Name : RecFact**

**Data Members :**

n : stores the number whose factorial is required.

r : stores an integer

**Member functions :**

RecFact() : default constructor

voidreadnum() : to enter values for n and r.

int factorial(int) : returns the factorial of the number using the recursive

technique.

voidfactseries() : to calculate and display the value of

n!

-----------------

r! \* (n-r)!

Specify the class RecFact giving the details of the constructor and member functions void readnum(), int factorial(int) and void factseries(). Also define the main function to create an object and call methods accordingly to enable the task.

28. An **Emirp** number is a number which is prime backwards and forwards. Example: 13 and 31 are both prime numbers. Thus 13 is an emirp number.

Design a class **Emirp** to check if a given number is Emirp number or not. Some of the members of the class are given below:

**Class Name : Emirp**

**Data Members**

n : stores the number

rev : stores the reverse of the number

f : stores the divisor

**Member functions**

Emirp(int nn) : to assign n=nn, rev=0, and f=2

int isprime(int x) : check if the number is prime using the recursive technique and

return 1 if prime otherwise return 0.

void isEmirp() : reverse the given number and check if both the original

number ad the reverse number are prime, by invoking the

function isprime(int) and display the result with an appropriate

message.

Specify the class **Emirp** giving details of the constructor(int), int isprime(int) and void isEmirp(). Define the main function to create an object and call the methods to check for Emirp number.

**29.** Design a class **VowelWord** to accept a sentence and calculate the frequency of words that begin with a vowel. The words in the input string are separated by a single blank space and terminated by a full stop.

The description of the class is given below:

**Class Name : VowelWord**

**Data members**

str : to store a sentence

freq : to store the frequency of words beginning with a vowel.

**Member functions**

VowelWord() : constructor to initialize data members to legal initial values.

Void readstr() : to accept a sentence.

void freq\_vowel( ) : counts the frequency of the words beginning with a vowel.

void display() : to display the original string and the frequency of the

words that begin with a vowel.

Specify the class **VowelWord** giving details of the constructor( ), void readstr(), void freq\_vowel() and void display(). Also defing the main function to create an object and call the methods accordingly to enable the task.

30. Input a sentence from the user and count the number of times, the words “an” and “and” are present in the sentence. Design a class Frequency using the description given below:

**Class name** : **Frequency**

**Data Members**

text : stores the sentence

countand : to store the frequency of the word and.

countan : to store the frequency of the word an.

len : stores the length of the string.

**Member functions**

Frequency() : constructor to initialize the data variables.

void accept(String n) : to assign n to text where the value of the parameter should be

in lowercase.

void checkandfreq() : to count the frequency of and using recursive technique.

void checkanfreq() : to count the frequency of an using recursive technique.

void display() : to display the frequency of “an” and “and” with suitable

messages.

Specify class Frequency giving details of the constructor(), void accept(String), void checkandfreq(), checkanfreq() and void display(). Also define the main function to create an object and call methods accordingly to enable the task.

**FILE OPERATION:**

31. A teacher wants to know the performance in Science subjects (Physics, Chemistry and Biology) of each student of her class. write a program to create a text file “performance.txt” to store the names and marks obtained in three subjects. the program calculates the total marks and the average. the grades have been awarded as per the criteria given below:

**Average Marks Grade**

>=80 A

>=60 && <80 B

>=40 && <60 C

<40 D

Take out the records in reader mode and display the name total average and grade of each student.

32. Code a program in Java to create a Binary file for the following purposes:

i) The binary file must be created to contain employee name, employee code and basic salary. the program should keep accepting the records till the user enters his choice as 1 and terminate it when the choice is 2.

ii) Use the file in appropriate mode to retrieve the records and find the following:

DA= 40% of the basic salary

HRA=30% of the basic salary

Gross=Basic salary + DA+HRA

A special allowance of 20% of the gross is calculated if employee code is less than or equal to 15 and the gross is less than or equal to 1500.

in-case the employee code is greater than 15 and the special allowance is Rs. 1000 fixed, then find the total amount = Gross+Special allowance.

display the salary of each employee along with the heading in the following format:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Code** | **Basic Salary** | **DA** | **HRA** | **Salary** | **Sp. Allowance** | **Total** |
| xxx | xxx | xxx | xxx | xxx | xxx | xxx | xxx |

\*\*\*\*\*