Shri G. S. Institute of Technology and Science
Department Of Computer Engineering
CO24057: Object Oriented Programming System
Lab Assignment # 04

Submission Date: 22nd November 2021@23:59 (Monday)
Late Submission: Not allowed
No copying allowed. If found then students involved in copying will get zero marks in this assignment.

1. A **special number** is a number in which the sum of the factorial of each digit is equal to the number itself.
   For example:- 145 = 1! + 4! + 5! = 1 + 24 + 120

   Design a class **Special** to check if a given number is a special number.
   Some of the members of the class are given below:
   Class name : Special
   Data members
   N : Integer
   Member functions
   Special() : constructor to assign 0 to n
   Special(int) : Parameterized constructor to assign a value to 'n'
   void sum() : calculate and display the sum of the first and last digit of n.
   void isSpecial() : check and display if the number n is a special number.

   Specify the class **Special** giving details of the constructor, void sum() and void isSpecial(). Define the main() function to create an object and call the member function accordingly to enable the task.

2. A **disarium number** is a number in which the sum of the digits to the power of their respective position is equal to the number itself.
   Example:$135=1^1+3^2+5^3$
   Hence,135 is a disarium number.

   Design a class **Disarium** to check if a given number is a disarium number or not. Some of the members of the class are given below:
   Classname: Disarium
   Data members/ instance variables:
   int num: stores the number
   int size: stores the size of the number
   Methods/Member functions:
   Disarium(int nn): parameterized constructor to initialize the data members num=nn and size=0
   void countDigit(): counts the total number of digits and assigns it to size

int sumofDigits(int n,int p): returns the sum of digits of the number(n) to the power of their respective positions(p) using **recursive** technique
void check(): checks whether the number is a disarium number and displays the result with an appropriate message

Specify the class **Disarium** giving the details of the constructor(),void countDigit(),int sumofDigits(int,int) and void check().Define the main() function to create an object and call the functions accordingly to enable the task.

3. Write a program that takes a positive integer N (in decimal) from the user and prints out its binary representation. (Hint: Repeatedly divide 2 into N and read the remainders backwards. First you can write a while loop to carry out this computation and print the bits in the wrong order. Then, use recursion to print the bits in the correct order.)

   Implement the functionality through the following class:

   Class name: BinaryConverter
   Data Members/Instance Variables:
   number: int type
   binNumber: long type
   Methods:
   convert() : takes an **int** parameter and converts it to an equivalent binary number. Returns the binary number as **long**.
   display() : displays the values of both number and binNumber
   Constructor method: receives an int parameter and stores it into a number. It then also initialises binNumber.

4. 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/instance variables:
   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 and 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.

5. A **happy number** is a number in which the eventual sum of the square of the digits of the number is equal to 1.
   Example:
   $28 = (2)^2 + (8)^2 = 4 + 64 = 68$
   $68 = (6)^2 + (8)^2 = 36 + 64 = 100$
   $100 = (1)^2 + (0)^2 + (0)^2 = 1 + 0 + 0 = 1$
   Hence, 28 is a happy number.
   Example:
   $12 = (1)^2 + (2)^2 = 1 + 4 = 5$
   Hence, 12 is not a happy number.

   Design a class **Happy** to check if a given number is a happy number. Some of the members of the class are given below:
   Class name: Happy
   Data Members/instance variables:
   n: stores the number Member functions
   Happy( ): constructor to assign 0 to n.
   void getnum (int nn): to assign the parameter value to the number n = nn.
   int sum_sq_digits (int x): returns the sum of the square of the digits of the number x, using the recursive technique
   void ishappy (): checks if the given number is a happy number by calling the function sum_sq_digits(int) and displays an appropriate message.

   Specify the class Happy giving details of the constructor (), void getnum (int), int sum_sq_digits (int) and void ishappy (). Also define a main function to create an object and call the methods to check for happy number.

6. Design a class **Perfect** to check if a given number is a perfect number or not. [A number is said to be perfect if sum of the factors of the number excluding itself is equal to the original number]
   Example: 6 = 1 + 2 + 3 (where 1, 2 and 3 are factors of 6, excluding itself)

   Some of the members of the class are given below:
   Class name: Perfect
   Data members/instance variables:
   num: to store the number
   Methods/Member functions:
   Perfect (int nn): parameterized constructor to initialize the data member num=nn

int sum_of_factors(int i): returns the sum of the factors of the number(num), excluding itself, using a recursive technique
void check(): checks whether the given number is perfect by invoking the function sum_of_factors() and displays the result with an appropriate message

Specify the class Perfect giving details of the constructor(), int sum_of_factors(int) and void check(). Define a main() function to create an object and call the functions accordingly to enable the task.

7. A class Revstr defines a recursive function to reverse a string and check whether it is a palindrome. The details of the class are given below:
Class name : Revstr
Data members
Str : stores the string.
Revst : stores the reverse of the string.
Member functions
void getStr() : to accept the string.
void recReverse(int) : to reverse the string using recursive technique.
void check() : to display the original string, its reverse and whether the string is a palindrome or not.

Specify the class Revstr giving details of the functions void getStr(), void recReverse(int) and void check(). Define a main() function to create an object and call the functions accordingly to enable the task.

8. Design a class **Change** to perform string related operations. The details of the class are given below:
Class name: Change
Data Members/instance variables:
str: stores the word
newstr: stores the changed word
len: store the length of the word
Member functions:
Change(): default constructor
void inputword( ): to accept a word
char caseconvert (char ch): converts the case of the character and returns it
void recchange (int): extracts characters using recursive technique and changes its case using caseconvert () and forms a new word
void display (): displays both the words

Specify the class Change, giving details of the Constructor ( ), member functions void inputword (), char caseconvert (char ch), void recchange (int) and void display (). Define

the main () function to create an object and call the functions accordingly to enable the above change in the given word.

9. An interface **Shape** is defined with a method area() which returns the area of the implementing shape. Create the interface Shape and the classes Circle and Rectangle which implement the interface Shape. These classes have attributes which reflect their dimensions (radius for a circle, height and width for a rectangle) which are set by their constructors. The details of the members of the interface and both the classes are given below:
   Interface name : Shape
   Member functions/methods:
   double area() : an abstract method that returns the area of the implementing shape

   Class name : Circle
   Data members/instance variables:
   radius : to store radius of the circle in decimal
   Member functions/methods:
   Circle(int r) : parameterized constructor to initialize radius=r
   double area() : to calculate area of the circle [area of a circle is 3.14*radius*radius]

   Class name : Rectangle
   Data members/instance variables:
   length : to store length of the rectangle in decimal
   breadth : to store breadth of the rectangle in decimal
   Member functions / methods:
   Rectangle(int l, int b) : parameterized constructor to initialize length=l, breadth=b
   double area() : to calculate area of the rectangle [area of a rectangle is length*breadth]

   Using the concept of inheritance, define the interface Shape and specify the classes Circle and Rectangle giving details of their constructors and double area( ) respectively.

10. An interface **Data** is defined with a data member and a method volume( ) which returns the volume of the implementing shape. A super class **Base** has been defined to contain the radius of a geometrical shape. Define a subclass **CalVol** which uses the property of the interface Data and the class Base and calculates the volume of a sphere. The details of the members of the interface and both the classes are given below:
    Interface name : Data
    Data member:
    double pi : initialize pi = 3.142
    Member functions/methods:
    double volume( ) : abstract method

Class name: Base
Data member/instance variable:
rad : to store the radius in decimal
Member functions/methods:
Base(...) : parameterized constructor to initialize the data member void show( ) : displays the radius with an appropriate message

Class name: CalVol
Data member/instance variable:
ht : to store the height in decimal
Member functions/methods:
CalVol(...) parameterized constructor to initialize the data members of both the classes double volume( ) : calculates the volume of a sphere by using the formula (pi x radius2 × height)
void show( ) : displays the data members of both the classes and the volume of the sphere with appropriate message

Define the interface Data and the super class Base. Using the concept of inheritance, specify the class CalVol giving the details of the constructor(.....), double volume( ) and void show ( ).