DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BENGALURU – 560068

Programming is an Art

OBJECT ORIENTED PROGRAMMING

COURSE CODE: 23EN1202

AY: 2024-25

DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Object Oriented Programming Laboratory

(23EN1202)

NAME: ……………………………………………………

USN: ………………………………………………………

Vision

To be recognized as a department of eminence in Computer Science and Engineering focusing on sustainability, inclusive technologies, and societal needs.

Mission

The Department of Computer Science and Engineering is committed to:

- **M1:** Impart quality technical education by designing and delivering contemporary Computer Science Engineering curricula while emphasizing leadership, ethics, values, and integrity.
- **M2:** Transform professionals into technically competent through industry-academia collaboration and innovation ecosystem.
- **M3:** Prepare Computer Science and Engineering graduates to meet ever-growing societal needs.

Values

The values that drive DSU and support its vision:

The Pursuit of Excellence

- A commitment to strive continuously to improve ourselves and our systems with the aim of becoming the best in our field.

Fairness

- A commitment to objectivity and impartiality, to earn the trust and respect of society.

Leadership

- A commitment to lead responsively and creatively in educational and research processes.

Integrity and Transparency

- A commitment to be ethical, sincere, and transparent in all activities and to treat all individuals with dignity and respect.

DSU

DAYANANDA SAGAR UNIVERSITY

Laboratory Certificate

This is to certify that Mr./Ms.____________________________________bearing University Seat number (USN)_______________________has satisfactorily completed the experiments in above practical subject prescribed by the University for the ________________ semester B.Tech. program in the Object-Oriented Programming Laboratory of this university during the year 2024-2025.

Date: ___________________

____
Signature of the Faculty in charge

MARKS	
Maximum	Obtained

____
Signature of the Chairman

Object Oriented Programming Laboratory

SN	Faculty Name	Designation	Role
1	Dr. Sridhar S K	Associate Professor	Python Course Lead
2	Dr. Girisha G S	ChairPerson & Professor	Member
3	Dr. Sivananda L Reddy	Associate Professor	Member
4	Prof. Shreekanth Salatogi	Assistant Professor	Member
5	Prof. Manoj Kumar	Assistant Professor	Member
6	Prof. Mala B A	Assistant Professor	Member
7	Prof. Vinayaka	Assistant Professor	Member
8	Prof. Manjushree T L	Assistant Professor	Member
9	Prof. Benaka Santosha	Assistant Professor	Member
10	Prof. Priyanka S Marellavar	Assistant Professor	Member
11	Prof. Sowmya H D	Assistant Professor	Member
12	Prof. K Radhika	Assistant Professor	Member
13	Prof. Chethan K S	Assistant Professor	Member
14	Prof. Fineta	Assistant Professor	Member
15	Prof. Sumanth	Assistant Professor	Member

Instructions for Laboratory Exercises:

1.	Create your own subdirectory in the computer. Save the python code with program number into your subdirectory.
2.	Maintain a copy of Anaconda Navigator software files in your subdirectory. You can write comments for your instructions using Semicolon (#) symbol.
3.	Design algorithms for the program question in the lab set.
4.	Develop the programs as per the steps discussed in algorithm.
5.	Execute the program and note down the results in your observation book.
6.	Consult the Lab In-charge/Instructor before executing experiments.

List of Object-Oriented Programming Problem statements

PN	Problem statement	Date	Marks	Sign
1	a. Develop a program to read and print values of variables of different data types. b. Develop a program to calculate the bill amount for an item given its quantity sold, value, discount, and tax.			
2	a. Develop a program to calculate the roots of a quadratic equation using conditional statements. b. Develop a program to find the sum of squares of even numbers.			
3	a. Develop a program to use a lambda function with an ordinary function. b. Develop a program where a sub-function is defined in one module and called by code in another module.			
4	a. Develop a program that encrypts a message by adding a key value to every character. Hint: If key=3 then the string “HelloWorld” encrypted as “KhoorZruog” b. Develop a program that takes username and PAN number as input, validate the information using isX function and print the details.			

5	a. Develop a program that reads a file line by line. Each line read from the file is copied to another file with line numbers specified at the beginning of the line. b. Develop a program that fetches data from a specified URL and writes in a file.			
6	a. Write a program that prompts the user to enter a filename. Open the file and print the frequency of each word in it. b. Develop a program to illustrate 5 built-in methods of List, Tuple, Set and dictionaries.			
7	a. Develop a program to deposit or withdraw money in a bank account. b. Develop a program that has a class person. Inherit a class Faculty from Person which also has a class Publications.			
8	a. Develop a program that overloads the operator + so that it can add two objects of class Fraction. b. Develop a program to create subclasses of Exception class to handle exceptions in a better way.			

Internals	**Lab manual with programs and Lab file**	**Total Marks**
(20)	(20)	(40)

Signature of the Student **Signature of the Staff**

COURSE OVERVIEW:

In this course student will be able to learn the concepts of object-oriented programming using class, objects, methods, Polymorphism, and different levels of inheritance and make use of Python programming environment to develop programs using conditionals, iterations, functions, strings, and files to store and retrieve data in system.

Python is a general-purpose programming language created in the late 1980s, and named after Monty Python, that's used by thousands of people to do things from testing microchips at Intel, to powering Instagram, to building video games with the PyGame library.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Major Topics covered:

- Why should you learn to write programs?
- Variables, Expressions and Statements.
- Conditional Execution, Functions, Iterations, Strings.
- Lists, Dictionaries, Tuples, Arrays and Sets.
- Classes, Objects and Methods.
- Exception handling methods, Files and Python libraries.

Text Books:

1. Reema Thareja, "Python programming: Using problem solving approach", 2nd Edition, Oxford university press, 2019.

Reference Books:

1. John V Guttag, "Introduction to Computation and Programming Using Python", The MIT press, 3rd edition, 2021.
2. Tony Gaddis, "Starting out with python", 4th edition, Pearson, 2019.
3. Allen Downey, Jeffrey Elkner and Chris Meyers, "How to think like a computer Scientist, Learning with Python", Green Tea Press, 2014.
4. Richard L. Halterman, "Learning to Program with Python", 2011.
5. Charles Dierbach, "Computer Science Using Python: A Computational Problem-Solving Focus", John Wiley, 2012.

PROGRAM: 1A

AIM: To develop a program to read and print values of variables of different data types.

ALGORITHM:

```
Step 1: Start
Step 2: Read an integer value from the user and assign it to an integer variable.
Step 3: Read a floating-point value from the user and assign it to a float variable.
Step 4: Read a string value from the user and assign it to a character variable.
Step 5: Read a complex value from the user and assign it to a complex variable.
Step 6: Read a boolean value from the user and assign it to a boolean variable.
Step 7: Assign None value to variable.
Step 8: Print the values of all variables along with their respective data types.
Step 9: End
```

PYTHON CODE:

```python
num = int(input("Enter the value of num : "))
pi = float(input("Enter the value of p i : "))
name = str(input("Enter your name : "))
compvalue=complex(input("Enter the complex value :"))
boolean=bool(input("Enter the boolean value :"))
nonetype=None
#Print the values of variables
print("\nEntered integer value is =",num)
print("\nEntered pi value is =",pi)
print("\nEntered name is =",name)
print("\nEntered complex value is =",compvalue)
print("\nEntered boolean value is =",boolean)
print("\nNoneType value is =",nonetype)
```

OUTPUT DATA:

```
Entered integer value is = 100
Entered pi value is = 3.14
Entered name is = DSU
Entered complex value is = (3+4j)
Entered boolean value is = True
NoneType value is = None
```

PROGRAM: 1B

AIM: To develop a program to calculate the bill amount for an item given its quantity sold, value, discount, and tax.

ALGORITHM:

```
Step 1: Start
Step 2: Read the quantity of the item sold from the user and store it in a variable.
Step 3: Read the value of the item from the user and store it in a variable.
Step 4: Read the discount percentage from the user and store it in a variable.
Step 5: Read the tax percentage from the user and store it in a variable.
Step 6: Calculate the gross amount by multiplying the quantity of the item by value of the item.
Step 7: Calculate the discount amount by multiplying discount percent by the gross amount and divide by 100.
Step 8: Subtract the discount amount from the gross amount to get the discounted price.
Step 9: Calculate the tax amount by multiplying the tax percent by the subtotal and divide by 100.
Step 10: Calculate the net bill amount by adding the sub total and the tax amount.
Step 11: Print the net bill amount.
Step 12: End
```

PYTHON CODE:

```python
qty = float(input("Enter the quantity of item sold : "))
val = float(input("Enter the value of item: "))
discountpercent = float(input ("Enter the discount percentage : "))
taxpercent = float (input ("Enter the tax : "))
grossamt = qty*val
discountamt = (discountpercent/100)*grossamt
subtotal = grossamt-discountamt
taxamt = (taxpercent/100)*subtotal
netamt = subtotal + taxamt
print("**********BILL************")
print("Quantity sold : \t",qty)
print("Price per item : \t",val)
print("\n\t\t\t ----------")
print("Gross Amount : \t\t" , grossamt)
print("Discount :\t\t-", discountamt)
print("\t\t\t ---------- ")
print("Discounted Sub Total : \t " , subtotal)
print("Tax :\t\t\t+",taxamt)
print( " \t\t\t -----------")
print("Total net amount to be paid ",netamt)
```

OUTPUT DATA:

```
Enter the quantity of item sold : 80
Enter the value of item: 50
Enter the discount percentage : 5
Enter the tax : 8
**********BILL************
Quantity sold :           80.0
Price per item :          50.0
                          ----------
Gross Amount :          4000.0
Discount :              - 200.0
                          ----------
Discounted Sub Total : 3800.0
Tax :                     + 304.0
                          -----------
Total net amount to be paid  4104.0
```

PROGRAM: 2A

AIM: To develop a program to calculate the roots of a quadratic equation using conditional statements.

ALGORITHM:

```
Step 1: Start

Step 2: Read the coefficients (a, b, c) of the quadratic equation from the user and store them in variables.
    Step 2a: Prompt the user to enter the coefficient 'a'.
    Step 2b: Read the input as a floating-point number and store it in the variable 'a'.
    Step 2c: Prompt the user to enter the coefficient 'b'.
    Step 2d: Read the input as a floating-point number and store it in the variable 'b'.
    Step 2e: Prompt the user to enter the coefficient 'c'.
    Step 2f: Read the input as a floating-point number and store it in the variable 'c'.

Step 3: Calculate the discriminant using the formula: discriminant = b^2 - 4ac.
    Step 3a.1: If the value of a is equal to 0:
    Step 3a.2: Print “This is not a quadratic equation, 'a' should not be 0”.

Step 4: Else Check the value of the discriminant using conditional statements.
    Step 4a: If the discriminant is greater than 0:
        Step 4a.1: Calculate the first root using the formula: root1 = (-b + sqrt(discriminant)) / (2 * a).
        Step 4a.2: Calculate the second root using the formula: root2 = (-b - sqrt(discriminant)) / (2 * a).
        Step 4a.3: Print both roots.
    Step 4b: Else if the discriminant is equal to 0:
        Step 4b.1: Calculate the root using the formula: root = -b / (2 * a).
        Step 4b.2: Print the root (it will be a repeated root).
    Step 4c: Else (if the discriminant is less than 0):
        Step 4c.1: Calculate the real part using the formula: real_part = -b / (2 * a).
        Step 4c.2: Calculate the imaginary part using the formula = math.sqrt(abs(discriminant)) / (2 * a)
        Step 4c.3: Print the imaginary roots.

Step 5: End
```

PYTHON CODE:

```python
import math
a = float(input("Enter the coefficient a: "))
b = float(input("Enter the coefficient b: "))
```

```
c = float(input("Enter the coefficient c: "))
discriminant = b*b - 4*a*c
if a == 0:
    print("This is not a quadratic equation, 'a' should not be 0.")
else:
    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2 * a)
        root2 = (-b - math.sqrt(discriminant)) / (2 * a)
        print("The equation has real and distinct roots\n")
        print("Root 1=",root1)
        print("Root 2=",root2)
    elif discriminant == 0:
        root = -b / (2 * a)
        print("The equation has real and equal roots: ",root)
    else:
        real_part = -b / (2 * a)
        imaginary_part = math.sqrt(abs(discriminant)) / (2 * a)
        print("The equation has real and complex roots")
        print("Root1=",real_part,"+",imaginary_part,"i")
        print("Root2=",real_part,"-",imaginary_part,"i")
```

OUTPUT DATA:

```
Enter the coefficient a: 1
Enter the coefficient b: -3
Enter the coefficient c: 2
The equation has real and distinct roots
Root 1=  2.0
Root 2=  1.0

Enter the coefficient a: 1
Enter the values of b : -4
Enter the values of c : 4
EQUAL ROOTS
The equation has real and equal roots:  2.0

Enter the coefficient a: 1
Enter the coefficient b: 2
Enter the coefficient c: 5
The equation has real and complex roots
Root1= -1.0 + 2.0 i
Root2= -1.0 - 2.0 i
```

PROGRAM: 2B

AIM: To develop a program to find the sum of squares of even numbers.

ALGORITHM:

```
Step 1: Start
Step 2: Read an integer n (upper limit) from the user.
Step 3: Initialize a variable 's' to 0 to store the sum of squares of even numbers.
Step 4: Loop over each integer i from 1 to n.
    Step 4a: Check if i is even.
    Step 4b: If i is even:
        Step 4b.1: Square the value of i.
        Step 4b.2: Add the square of i to the 's' variable.
Step 5: Print the 's' variable, which holds the sum of squares of even numbers.
Step 6: End
```

PYTHON CODE:

```
n = int(input("Enter the number : "))
s=0
for i in range(1,n+1):
    if(i%2 == 0):
        term = i*i
    else:
        term=0
    s = s+term
print ("The sum of squares of even number less than", n, "is", s)
```

OUTPUT DATA:

```
Enter the number : 10
The sum of squares of even number less than 10 is 220
```

PROGRAM: 3A

AIM: To develop a program to use a lambda function with an ordinary function.

ALGORITHM:

```
Step 1: Start
Step 2: Prompt the user to input the first number and store it in the variable num1.
Step 3: Prompt the user to input the second number and store it in the variable num2.
Step 4: Call the add() function:
        Step 4a: Pass num1 and num2 as arguments to the add(x, y) function.
Step 5: Inside the add(x, y) function:
        Step 5a: Define a lambda function that takes two parameters (a and b), and returns the result of a + b.
Step 6: Store the returned result from add(num1, num2) in the variable sum.
Step 7: Print the result.
Step 8: End.
```

PYTHON CODE:

```python
def add(x,y):
   return (lambda a,b:a+b) (x,y)

num1=int(input("Enter first value"))
num2=int(input("Enter second value"))
sum=add(num1,num2)
print("Result=",sum)
```

OUTPUT DATA:

```
Enter first value 4
Enter second value 6
Result= 10
```

PROGRAM: 3B

AIM: To develop a program where a sub-function is defined in one module and called by code in another module.

ALGORITHM:

```
Module A.py:
    Step 1: Define the function add(x, y) :
        Takes two parameters x and y.
        Returns the sum of x and y.
    Step 2: Define the function sub(x, y):
        Takes two parameters x and y.
        Returns the difference of x and y.
    Step 3: Define the function mul(x, y):
        Takes two parameters x and y.
        Returns the product of x and y.
    Step 4: Define the function div(x, y):
        Takes two parameters x and y.
        Returns the division result of x divided by y.
Module B.py:
    Step 1: Import the add and sub functions from the module A.py.
    Step 2: Call the add() function with arguments 3 and 5:
        Pass 3 and 5 as arguments to add().
        Store the result of add(3, 5) in the variable sum.
    Step 3: Call the sub() function with arguments 6 and 4:
        Pass 6 and 4 as arguments to sub().
        Store the result of sub(6, 4) in the variable diff.
    Step 4: Print the result of the addition:
        Display the message "Addition of two numbers:" followed by the value of sum.
    Step 5: Print the result of the subtraction:
        Display the message "Difference of two numbers:" followed by the value of diff.
    Step 6: End.
```

PYTHON CODE:

Module A.py:

```
def add(x, y):
   return x + y
def sub(x, y):
   return x - y
def mul(x, y):
   return x * y
def div(x, y):
   return x/y
```

Module B.py:

```
from A import add, sub
sum = add(3, 5)
diff = sub(6, 4)
print("Addition of two numbers:", sum)
print("Difference of two numbers:", diff)
```

OUTPUT DATA:

```
Addition of two numbers: 8
Difference of two numbers: 2
```

PROGRAM: 4A

AIM: To develop a program that encrypts a message by adding a key value to every character.

ALGORITHM:

```
Step 1: Start
Step 2: Initialize the message variable with the string "HelloWorld".
Step 3: Initialize the index variable to 0.
Step 4: While the index is less than the length of the message, repeat steps 5 to 7.
    Step 4a: Get the character at the current index in the message and store it in the letter variable.
    Step 4b: Calculate the ASCII value of the letter using the ord() function.
    Step 4c: Add 3 to the ASCII value of the letter.
    Step 4d: Convert the resulting ASCII value back to a character using the chr() function.
    Step 4e: Print the encrypted character without a newline character.
    Step 4f: Increment the index by 1.
Step 5: End
```

PYTHON CODE:

```python
message = "HelloWorld"
index = 0
while index < len(message) :
    letter = message[index]
    print(chr(ord(letter)+ 3),end=' ')
    index += 1
```

OUTPUT DATA:

```
K h o o r Z r u o g
```

PROGRAM: 4B

AIM: To develop a program that takes username and PAN number as input, validate the information using isX function and print the details.

ALGORITHM:

```
Step 1: Start
Step 2: Prompt the user to enter their name and store it in the variable 'username'.
    Step 2a: Check if the name contains only alphabetic characters using the 'isalpha()' method.
        Step 2a.1: If name contains non-alphabetic characters, print an error message indicating invalid input.
    Step 2b: Else Prompt the user to enter their PAN card number and store it in the variable 'pan'.
    Step 2c: Check if the PAN card number contains only alphanumeric characters using the 'isalpha()' and 'isdigit()' methods.
        Step 2c.1: Print a message confirming the name and PAN card number.
    Step 2d: Else If the PAN card number contains non-alphanumeric characters, print an error message indicating invalid input.
```

PYTHON CODE:

```python
username=input("Enter Name:")
if username.isalpha()==False:
    print("Invalid Input")
else:
    pan=input("Enter PAN:")
    if len(pan)==10 and pan[0:5].isalpha()==True and pan[5:9].isdigit()==True and pan[9].isalpha()==True:
        print("Name is ",username)
        print("PAN is ",pan)
    else:
        print("Invalid Input")
```

OUTPUT DATA:

```
Enter name: DSU
Enter PAN: ABCDE1234F
Name is DSU
PAN is ABCDE1234F

Enter Name: CSE
Enter PAN: ABC123F
Invalid Input
```

PROGRAM: 5A

AIM: To develop a program that reads a file line by line. Each line read from the file is copied to another file with line numbers specified at the beginning of the line.

ALGORITHM:

```
Step 1: Start
Step 2: Open the file "file1.txt" in read mode and store the file object in a variable named 'file1'.
Step 3: Open or create a file named "file2.txt" in write mode and store the file object in a variable named 'file2'.
Step 4: Initialize a variable 'num' to 1 to keep track of the line numbers.
Step 5: Iterate over each line in the file 'file1':
    Step 5a: Read each line from 'file1'.
    Step 5b: Write the line number followed by a colon ":" and the line content to 'file2'.
    Step 5c: Increment the value of 'num' by 1 to prepare for the next line.
Step 6: Close the file 'file1'.
Step 7: Close the file 'file2'.
Step 8: End
```

PYTHON CODE:

```python
file1 = open("/Users/DSU/python/file1.txt", "r")
file2 = open("file2.txt", "w")
num = 1
for line in file1:
    file2.write(str(num)  + ":" +line)
    num = num + 1
file1.close()
file2.close()
print ("Contents read from file1.txt and copied to file2.txt")
```

OUTPUT DATA:

```
Contents read from file1.txt and copied to file2.txt
```

PROGRAM: 5B

AIM: To develop a program that fetches data from a specified URL and writes in a file.

ALGORITHM:

```
Step 1: Import the requests module to make HTTP requests.
Step 2: Set the variable url to "https://google.com/data", representing the URL from which you want to retrieve data.
Step 3: Set the variable filename to "urlgoogle.txt", which will be the name of the file where the content will be written.
Step 4: Use requests.get(url) to send a GET request to the URL and store the response in the variable response.
Step 5: Open the file with the name stored in filename in write-binary mode ('wb'), using a with statement to ensure the file is properly managed (opened and closed).
Step 6: Write the content of the response (response.content) to the file using f.write().
Step 7: Print the message "Data successfully written to" followed by the filename to confirm the data was written to the file.
Step 8: End.
```

PYTHON CODE:

```python
import requests
url="https://google.com/data"
filename="urlgoogle.txt"
response=requests.get(url)
with open(filename,'wb') as f:
    f.write(response.content)
print("Data successfully written to", filename)
```

OUTPUT DATA:

```
Data successfully written to urlgoogle.txt
```

PROGRAM: 6A

AIM: To write a program that prompts the user to enter a filename. Open the file and print the frequency of each word in it.

ALGORITHM:

```
Step 1: Prompt the user to input a filename and store the input in the variable fname.
Step 2: Open the file with the name stored in fname in read mode ('r') and store the file object in the variable fd.
Step 3: Initialize an empty dictionary counts to store the word frequencies.
Step 4: Start a loop to iterate over each line in the file fd.
    Step 4.1: For each line, split the line into a list of words using line.split().
    Step 4.2: Start another loop to iterate over each word in the list words.
        Step 4.2.1: Check if the word w is already present in the dictionary counts:
            If yes: Increment the count of that word by 1 (counts[w] += 1).
            If no: Add the word w to the dictionary with an initial count of 1 (counts[w] = 1).
Step 5: After the loops finish, print the counts dictionary, which now contains the frequency of each word in the file.
Step 6: End.
```

PYTHON CODE:

```
fname=input("Enter the filename: ")
fd = open(fname,'r')
counts = dict()
for line in fd:
    words = line.split()
    for w in words:
        if w in counts:
            counts[w] += 1
        else:
            counts[w] = 1
print(counts)
```

OUTPUT DATA:

```
Enter the filename: file1.txt
{'Python': 2, 'is': 1, 'a': 1, 'general-purpose': 1, 'programming': 1, 'language': 1, 'created': 1, 'in': 1, 'the': 2, 'late': 1, '1980s': 1, 'and': 1, 'named': 1, 'after': 1, 'Monty': 1, 'that’s': 1, 'used': 1, 'by': 1, 'thousands': 1, 'of': 1, 'people': 1, 'to': 3, 'do': 1, 'things': 1, 'from': 1, 'testing': 1, 'microchips': 1, 'at': 1, 'Intel': 1, 'powering': 1, 'Instagram,': 1, 'building': 1, 'video': 1, 'games': 1, 'with': 1, 'PyGame': 1, 'library.': 1}
```

PROGRAM: 6B

AIM: To develop a program to illustrate 5 built-in methods of List, Tuple, Set and dictionaries.

ALGORITHM:

```
Step 1: Start
Step 2: Define a function 'list_methods_demo()' to demonstrate list methods:
    Step 2a: Initialize a list 'my_list' with elements [1, 2, 3, 4, 5].
    Step 2b: Append the element 6 to 'my_list' using the append() method.
    Step 2c: Extend 'my_list' with the list [7, 8, 9] using the extend() method.
    Step 2d: Insert the element 10 at index 2 in 'my_list' using the insert() method.
    Step 2e: Remove the element 3 from 'my_list' using the remove() method.
    Step 2f: Pop the last element from 'my_list' using the pop() method and store it in 'popped_element'.
    Step 2g: Print the result of each operation.
Step 3: Define a function 'tuple_methods_demo()' to demonstrate tuple methods:
    Step 3a: Initialize a tuple 'my_tuple' with elements (16, 12, 35, 4, 52, 4).
    Step 3b: Find the index of the element 35 in 'my_tuple' using the index() method and store it in 'index'.
    Step 3c: Count the occurrences of the element 4 in 'my_tuple' using the count() method and store it in 'count'.
    Step 3d: Sort the elements of ‘my_tuple’ using the sorted() function.
    Step 3e: Find the minimum element of ‘my_tuple’ using min() function.
    Step 3f: Find the sum of elements of ‘my_tuple’ using sum() function.
    Step 3g: Print the result of each operation.
Step 4: Define a function 'set_methods_demo()' to demonstrate set methods:
    Step 4a: Initialize a set 'my_set' with elements {1, 2, 3, 4, 5}.
    Step 4b: Add the element 6 to 'my_set' using the add() method.
    Step 4c: Remove the element 3 from 'my_set' using the remove() method.
    Step 4d: Discard the element 4 from ‘my_set’ using the discard() method.
    Step 4e: Pop an arbitrary element from 'my_set' using the pop() method and store it in 'popped_element'.
    Step 4f: Clear the elements of ‘my_set’ using clear() method.
    Step 4g: Print the result of each operation.
Step 5: Define a function 'dictionary_methods_demo()' to demonstrate dictionary methods:
    Step 5a: Initialize a dictionary 'my_dict' with key-value pairs {'a': 1, 'b': 2, 'c': 3}.
    Step 5b: Get the value corresponding to the key 'b' from 'my_dict' using the get() method and store it in 'value'.
    Step 5c: Pop the value corresponding to the key 'c' from 'my_dict' using the pop() method and store it in 'popped_value'.
    Step 5d: Update 'my_dict' with the key-value pair {'d': 4} using the update() method.
    Step 5e: Display the list of all keys of ‘my_dict’ using keys() method.
    Step 5f: Display the list of all items of ‘my_dict’ using items() method.
    Step 5g: Print the result of each operation.
Step 6: Define the main function 'main()' to call all the methods and display their results:
    Step 6a: Call 'list_methods_demo()', 'tuple_methods_demo()', 'set_methods_demo()', and 'dictionary_methods_demo()' functions.
    Step 6b: Print appropriate messages to indicate the demonstration of each data structure.
Step 7: End
```

PYTHON CODE:

```
# List methods
def list_methods_demo():
    # Initialize a list
    my_list = [1, 2, 3, 4, 5]
    # Append method
    my_list.append(6)
    print("Append method:", my_list)
    # Extend method
    my_list.extend([7, 8, 9])
    print("Extend method:", my_list)
    # Insert method
    my_list.insert(2, 10)
    print("Insert method:", my_list)
    # Remove method
    my_list.remove(3)
    print("Remove method:", my_list)
    # Pop method
    popped_element = my_list.pop()
    print("Popped element using pop method:", popped_element)

# Tuple methods
def tuple_methods_demo():
    # Initialize a tuple
    my_tuple = (16, 12, 35, 4, 52, 4)
    # Index method
    index = my_tuple.index(35)
    print("Index of element 35:", index)
    # Count method
    count = my_tuple.count(4)
    print("Count of element 4:", count)
    # sorted function
    print("Sorted elements are:", sorted(my_tuple))
    # min function
    print("Minimum element is:", min(my_tuple))
    # sum function
    print("Sum of elements is:", sum(my_tuple))
    print("Updated dictionary using update method:", my_dict)

# Set methods
def set_methods_demo():
    # Initialize a set
    my_set = {1, 2, 3, 4, 5}
    # Add method
    my_set.add(6)
    print("Add method:", my_set)
    # Remove method
    my_set.remove(3)
    print("Remove method:", my_set)
    # Discard method
    my_set.discard(4)
    print("Discard method:", my_set)
    # Pop method
    popped_element = my_set.pop()
    print("Popped element using pop method:", popped_element)
    # Clear method
    my_set.clear()
    print("Clear method:", my_set)

# Dictionary methods
def dictionary_methods_demo():
    # Initialize a dictionary
    my_dict = {'a': 1, 'b': 2, 'c': 3}
    # Get method
    value = my_dict.get('b')
    print("Value for key 'b' using get method:", value)
    # Pop method
    popped_value = my_dict.pop('c')
    print("Popped value for key 'c' using pop method:", popped_value)
    # Update method
    my_dict.update({'d': 4})
    print("Updated dictionary using update method:", my_dict)
    # keys method
    print("Using keys method:", my_dict.keys())
    # values method
    print("Using values method:", my_dict.values())
    # items method
    print("Using items method:", my_dict.items())

# Main function
```

```
def main():
    print("List methods:")
    list_methods_demo()
    print("\nTuple methods:")
    tuple_methods_demo()
    print("\nSet methods:")
    set_methods_demo()
    print("\nDictionary methods:")
    dictionary_methods_demo()

main()
```

OUTPUT DATA:

```
List methods:
Append method: [1, 2, 3, 4, 5, 6]
Extend method: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Insert method: [1, 2, 10, 3, 4, 5, 6, 7, 8, 9]
Remove method: [1, 2, 10, 4, 5, 6, 7, 8, 9]
Popped element using pop method: 9

Tuple methods:
Index of element 35: 2
Count of element 4: 2
Sorted elements are: [4, 4, 12, 16, 35, 52]
Minimum element is: 4
Sum of elements is: 123

Set methods:
Add method: {1, 2, 3, 4, 5, 6}
Remove method: {1, 2, 4, 5, 6}
Discard method: {1, 2, 5, 6}
Popped element using pop method: 1
Clear method: set()

Dictionary methods:
Value for key 'b' using get method: 2
Popped value for key 'c' using pop method: 3
```

```
Updated dictionary using update method: {'a': 1, 'b': 2, 'd': 4}
Using keys method: dict_keys(['a', 'b', 'd'])
Using values method: dict_values([1, 2, 4])
Using items method: dict_items([('a', 1), ('b', 2), ('d', 4)])
```

PROGRAM: 7A

AIM: To develop a program to deposit or withdraw money in a bank account.

ALGORITHM:

```
Step 1: Start
Step 2: Define a class named 'Account':
    Step 2a: Define the '_ _init_ _' method:
        Step 2a.1: Initialize an instance variable 'balance' to 0.
        Step 2a.2: Print "New Account Created."
    Step 2b: Define the 'deposit' method:
        Step 2b.1: Prompt the user to enter the amount to deposit.
        Step 2b.2: Convert the input to a float and store it in the variable 'amount'.
        Step 2b.3: Add 'amount' to the 'balance' attribute of the object.
        Step 2b.4: Print the updated balance.
    Step 2c: Define the 'withdraw' method:
        Step 2c.1: Prompt the user to enter the amount to withdraw.
        Step 2c.2: Convert the input to a float and store it in the variable 'amount'.
        Step 2c.3: Check if 'amount' is greater than the 'balance':
            Step 2c.3.1: If true, print "Insufficient balance".
            Step 2c.3.2: If false, subtract 'amount' from the 'balance'.
        Step 2c.4: Print the updated balance.
    Step 2d: Define the 'enquiry' method:
        Step 2d.1: Print the current balance.
Step 3: Create an instance of the 'Account' class and store it in a variable named 'account'.
Step 4: Call the 'deposit' method on the 'account' object to deposit funds.
Step 5: Call the 'withdraw' method on the 'account' object to withdraw funds.
Step 6: Call the 'enquiry' method on the 'account' object to display the current balance.
Step 7: End
```

PYTHON CODE:

```
class Account:
    def _ _init_ _(self):
        self.balance = 0
        print( 'New Account Created.')

    def deposit(self):
        amount = float(input('Enter amount to deposit : '))
        self.balance += amount
        print( 'New Balance : %f' %self.balance)

    def withdraw(self):
        amount = float(input('Enter amount to withdraw : '))
        if(amount > self.balance):
            print( 'Insufficient balance')
        else:
            self.balance -= amount
            print ('New Balance : %f' %self.balance)

    def enquiry(self):
        print( 'Balance : %f' %self.balance)
account = Account()
account.deposit()
account.withdraw()
account.enquiry()
```

OUTPUT DATA:

```
New Account Created.
Enter amount to deposit : 1000
New Balance : 1000.000000
Enter amount to withdraw : 100
New Balance : 900.000000
Balance : 900.000000
```

PROGRAM: 7B

AIM: To develop a program that has a class person. Inherit a class Faculty from Person which also has a class Publications.

ALGORITHM:

```
Step 1: Start
Step 2: Define a class named 'Person':
    Step 2a: Define the '__init__' method:
        Step 2a.1: Initialize instance variables 'name', 'age', and 'gender' using the parameters passed to the constructor.
    Step 2b: Define the 'display' method:
        Step 2b.1: Print the name, age, and gender of the person.
Step 3: Define a class named 'Publications':
    Step 3a: Define the '__init__' method:
        Step 3a.1: Initialize instance variables 'no_RP', 'no_Books', and 'no_Art' using the parameters passed to the constructor.
    Step 3b: Define the 'display' method:
        Step 3b.1: Print the number of research papers published, number of books published, and number of articles published.
Step 4: Define a class named 'Faculty' which inherits from 'Person':
    Step 4a: Define the '__init__' method:
        Step 4a.1: Call the superclass '__init__' method to initialize the 'name', 'age', and 'gender' attributes.
        Step 4a.2: Initialize instance variables 'desig' and 'dept' using the parameters passed to the constructor.
        Step 4a.3: Create an instance of the 'Publications' class and assign it to the 'Pub' attribute.
    Step 4b: Define the 'display' method:
        Step 4b.1: Call the superclass 'display' method to display the name, age, and gender of the faculty member.
        Step 4b.2: Print the designation and department of the faculty member.
        Step 4b.3: Call the 'display' method of the 'Publications' object stored in the 'Pub' attribute.
Step 5: Create an instance of the 'Faculty' class with the given parameters.
Step 6: Call the 'display' method of the faculty member to display their information and publications.
```

Step 7: End

PYTHON CODE:

```
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender
    def display(self) :
        print ("NAME : ", self.name)
        print("AGE: ", self.age)
        print("GENDER : ", self.gender )
class Publications:
    def __init__(self, no_RP, no_Books, no_Art):
        self.no_RP = no_RP
        self.no_Books = no_Books
        self.no_Art = no_Art
    def display(self):
        print ("Number of Research papers Published: ", self.no_RP)
        print("Number of Books Published: ", self.no_Books)
        print ("Number of Articles Published : " , self.no_Art)
class Faculty(Person):
    def __init__(self, name, age, sex, desig, dept, no_RP, no_Books, no_Art):
        Person.__init__(self, name, age, gender)
        self.desig = desig
        self.dept = dept
        self.Pub = Publications(no_RP, no_Books, no_Art)
    def display(self):
        Person.display(self)
        print ("DESIGNATION: ", self.desig)
        print ("DEPARTMENT: ", self.dept)
        self.Pub.display()
```

```
F = Faculty("Pooja",38,"Female","TIC","Computer Science",22,1,3)
F. display()
```

OUTPUT DATA:

```
NAME :  Pooja
AGE:  38
GENDER:  Female
DESIGNATION:  TIC
DEPARTMENT:  Computer Science
Number of Research papers Published:  22
Number of Books Published:  1
Number of Articles Published :  3
```

PROGRAM: 8A

AIM: To develop a program that overloads the operator + so that it can add two objects of class Fraction.

ALGORITHM:

```
Step 1: Start
Step 2: Define a function 'GCD' to calculate the greatest common divisor (GCD) of two numbers 'num' and 'deno':
    Step 2a: If 'deno' is equal to 0:
        Step 2a.1: Return 'num' as the GCD.
    Step 2b: Else:
        Step 2b.1: Return the GCD of 'deno' and the remainder of 'num' divided by 'deno' using recursion.
Step 3: Define a class named 'Fraction':
    Step 3a: Define the '__init__' method:
        Step 3a.1: Initialize instance variables 'num' and 'deno' to 0 and 1, respectively.
    Step 3b: Define the 'get' method:
        Step 3b.1: Prompt the user to enter the numerator and denominator.
        Step 3b.2: Convert the inputs to integers and store them in 'num' and 'deno'.
    Step 3c: Define the 'simplify' method:
        Step 3c.1: Calculate the greatest common divisor (GCD) of 'num' and 'deno' using the 'GCD' function.
        Step 3c.2: Divide 'num' and 'deno' by their GCD to simplify the fraction.
    Step 3d: Define the '__add__' method to overload the addition operator:
        Step 3d.1: Create a temporary Fraction object 'Temp'.
        Step 3d.2: Calculate the numerator of 'Temp' as the sum of products of numerators and denominators.
        Step 3d.3: Calculate the denominator of 'Temp' as the product of denominators.
        Step 3d.4: Return the 'Temp' object.
    Step 3e: Define the 'display' method:
        Step 3e.1: Simplify the fraction.
        Step 3e.2: Print the simplified fraction.
Step 4: Create three instances of the 'Fraction' class: 'F1', 'F2', and 'F3'.
Step 5: Prompt the user to enter the numerator and denominator for 'F1' and 'F2' using the 'get' method.
```

Step 6: Perform addition of 'F1' and 'F2' using the overloaded '+' operator and store the result in 'F3'.

Step 7: Print "RESULTANT FRACTION IS : ".

Step 8: Display the resultant fraction 'F3' using the 'display' method.

Step 9: End

PYTHON CODE:

```
def GCD(num, deno):
    if(deno == 0):
        return num
    else:
        return GCD(deno,num%deno)
class Fraction:
    def __init__(self):
        self.num = 0
        self.deno = 1
    def get(self):
        self.num = int(input ("Enter the numerator : "))
        self.deno = int(input("Enter the denominator : "))
    def simplify(self):
        common_divisor = GCD(self.num, self.deno)
        self.num //= common_divisor
        self.deno //= common_divisor
    def __add__(self, F):
        Temp = Fraction()
        Temp.num = (self.num * F.deno) + (F.num * self .deno)
        Temp.deno = self.deno * F.deno
        return Temp
    def display(self):
        self.simplify()
        print(self.num, "/", self.deno)
```

```
F1 = Fraction()
F1.get()
F2 = Fraction()
F2.get()
F3 = Fraction()
F3 = F1 + F2
print ("RESULTANT FRACTION IS : ")
F3.display()
```

OUTPUT DATA:

```
Enter the numerator : 4
Enter the denominator : 10
Enter the numerator : 2
Enter the denominator : 5
RESULTANT FRACTION IS :
4 / 5
```

PROGRAM: 8B

AIM: To develop a program to create subclasses of Exception class to handle exceptions in a better way.

ALGORITHM:

```
Step 1: Start
Step 2: Define a base class named 'Error':
    Step 2a: Define a method 'message' which raises a NotImplementedError.
        Step 2a.1: Raise a NotImplementedError.
Step 3: Define a subclass named 'InputError' which inherits from 'Error':
    Step 3a: Define the '__init__' method:
        Step 3a.1: Initialize instance variables 'expr' and 'msg' using the parameters passed to the constructor.
    Step 3b: Define the 'message' method:
        Step 3b.1: Print the error message indicating an error in the input expression.
Step 4: Try block to handle exceptions:
    Step 4a: Prompt the user to enter a value for 'a' and store it in the variable 'a'.
    Step 4b: Raise an 'InputError' exception with the expression "input(\"Enter a : s\")" and a custom message "Input Error".
Step 5: Except block to handle 'InputError' exceptions:
    Step 5a: Catch the 'InputError' exception and store it in the variable 'ie'.
    Step 5b: Call the 'message' method of the 'ie' object to print the error message.
Step 6: End
```

PYTHON CODE:

```python
class Error(Exception):
    def message(self):
        raise NotImplementedError()
class InputError(Error):
    def __init__(self, expr, msg):
        self.expr = expr
        self.msg = msg
    def message(self):
        print("Error in input in expression"),
        print(self.expr)
try:
    a = input("Enter a : ")
    raise InputError("input(\"Enter a : s\")", "Input Error")
except InputError as ie:
    ie.message()
```

OUTPUT DATA:

```
Enter a : 10
Error in input in expression
input("Enter a : s")
```

OPEN ENDED EXPERIMENTS

PROGRAM: 1

AIM: To Make a class Book with members, title, author, publisher and ISBN number. The functions of the class should read and display the data.

PYTHON CODE:

```
class Book:
    def __init__(self):
        self.title = input('Enter the title: ')
        self.author = input('Enter the author: ')
        self.publisher = input('Enter the publisher: ')
        self.isbn_number = input('Enter the ISBN number: ')

    def read_data(self):
        self.title = input('Enter the title: ')
        self.author = input('Enter the author: ')
        self.publisher = input('Enter the publisher: ')
        self.isbn_number = input('Enter the ISBN number: ')

    def display_data(self):
        print('Title: {}'.format(self.title))
        print('Author: {}'.format(self.author))
        print('Publisher: {}'.format(self.publisher))
        print('ISBN number: {}'.format(self.isbn_number))

def main():
    book = Book()
    book.display_data()
    print()
    book.read_data()
    book.display_data()

if __name__ == '__main__':
    main()
```

OUTPUT DATA:

```
Enter the title: Bhagvadgita
Enter the author: Krishna
Enter the publisher: Gita Press
Enter the ISBN number: 3456
Title: Bhagvadgita
Author: Krishna
Publisher: Gita Press
ISBN number: 3456

Enter the title: Ramayana
Enter the author: Valmiki
Enter the publisher: Gita Press
Enter the ISBN number: 1234
Title: Ramayana
Author: Valmiki
Publisher: Gita Press
ISBN number: 1234
```

PROGRAM: 2

AIM: To develop an abstract class Vehicle, derive three classes Car, Motorcycle and Truck from it and define appropriate methods and print the details of the vehicle.

PYTHON CODE:

```
from abc import ABC, abstractmethod

class Vehicle(ABC):
    @abstractmethod
    def __init__(self, name, num_wheels):
        self.name = name
        self.num_wheels = num_wheels

    @abstractmethod
    def display_details(self):
        pass

class Car(Vehicle):
    def __init__(self, name, num_doors):
        super().__init__(name, 4)
        self.num_doors = num_doors

    def display_details(self):
        print('Name: {}'.format(self.name))
        print('Number of wheels: {}'.format(self.num_wheels))
        print('Number of doors: {}'.format(self.num_doors))

class Motorcycle(Vehicle):
    def __init__(self, name, has_sidecar):
        super().__init__(name, 2)
        self.has_sidecar = has_sidecar

    def display_details(self):
        print('Name: {}'.format(self.name))
        print('Number of wheels: {}'.format(self.num_wheels))
        print('Has sidecar: {}'.format(self.has_sidecar))
```

```
class Truck(Vehicle):
    def __init__(self, name, payload_capacity):
        super().__init__(name, 6)
        self.payload_capacity = payload_capacity

    def display_details(self):
        print('Name: {}'.format(self.name))
        print('Number of wheels: {}'.format(self.num_wheels))
        print('Payload capacity: {}'.format(self.payload_capacity))

def main():
    car = Car('Honda Civic', 4)
    car.display_details()
    print()

    motorcycle = Motorcycle('Harley Davidson', False)
    motorcycle.display_details()
    print()

    truck = Truck('Ford F-150', 3000)
    truck.display_details()

if __name__ == '__main__':
    main()
```

OUTPUT DATA:

```
Name: Honda Civic
Number of wheels: 4
Number of doors: 4
Name: Harley Davidson
Number of wheels: 2
Has sidecar: False
Name: Ford F-150
Number of wheels: 6
Payload capacity: 3000
```

PROGRAM: 3

AIM: To develop a program to generate a simple calculator and an year wise calendar.

PYTHON CODE:

```
import calendar

def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        raise ValueError("Cannot divide zero")
    return x / y

def calculate():
    print('Please type in the math operation you would like to complete:')
    print('+ for addition')
    print('- for subtraction')
    print('* for multiplication')
    print('/ for division')

    operation = input()

    num1 = float(input('Enter your first number: '))
    num2 = float(input('Enter your second number: '))

    if operation == '+':
        print('{} + {} = '.format(num1, num2))
        print(add(num1, num2))
    elif operation == '-':
        print('{} - {} = '.format(num1, num2))
        print(subtract(num1, num2))

    elif operation == '*':
        print('{} * {} = '.format(num1, num2))
        print(multiply(num1, num2))

    elif operation == '/':
        print('{} / {} = '.format(num1, num2))
        print(divide(num1, num2))

    else:
        print('You have not typed a valid operator, please run the program again.')

def again():
    calc_again = input('Do you want to calculate again? Please type Y for YES or N for
NO.\n')
    if calc_again.upper() == 'Y':
        calculate()
    elif calc_again.upper() == 'N':
        print('See you later.')
    else:
        again()

def welcome():
    print('Welcome to Calculator')

def year_wise_calendar():
    year = int(input('Enter the year: '))

    print(calendar.calendar(year))

welcome()
```

```
while True:
    print("\n1. Calculator")
    print("2. Year-wise Calendar")
    print("3. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':
        calculate()
        again()
    elif choice == '2':
        year_wise_calendar()
    elif choice == '3':
        print('See you later.')
        break
    else:
        print('You have not typed a valid choice, please try again.')
```

OUTPUT DATA:

```
Welcome to Calculator

1. Calculator
2. Year-wise Calendar
3. Exit
Enter your choice: 1
Please type in the math operation you would like to complete:
+ for addition
- for subtraction
* for multiplication
/ for division
+
Enter your first number: 22
Enter your second number: 11
22.0 + 11.0 =
33.0
Do you want to calculate again? Please type Y for YES or N for NO.
N
See you later.

1. Calculator
2. Year-wise Calendar
3. Exit
Enter your choice: 2
Enter the year: 2024
```

```
                                  2024

      January                   February                   March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                1  2  3  4                   1  2  3
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       4  5  6  7  8  9 10
15 16 17 18 19 20 21      12 13 14 15 16 17 18      11 12 13 14 15 16 17
22 23 24 25 26 27 28      19 20 21 22 23 24 25      18 19 20 21 22 23 24
29 30 31                  26 27 28 29               25 26 27 28 29 30 31

       April                      May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7             1  2  3  4  5                      1  2
 8  9 10 11 12 13 14       6  7  8  9 10 11 12       3  4  5  6  7  8  9
15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
29 30                     27 28 29 30 31            24 25 26 27 28 29 30

        July                     August                  September
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                1  2  3  4                         1
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       2  3  4  5  6  7  8
15 16 17 18 19 20 21      12 13 14 15 16 17 18       9 10 11 12 13 14 15
22 23 24 25 26 27 28      19 20 21 22 23 24 25      16 17 18 19 20 21 22
29 30 31                  26 27 28 29 30 31         23 24 25 26 27 28 29
                                                    30

      October                   November                  December
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                   1  2  3                         1
 7  8  9 10 11 12 13       4  5  6  7  8  9 10       2  3  4  5  6  7  8
14 15 16 17 18 19 20      11 12 13 14 15 16 17       9 10 11 12 13 14 15
21 22 23 24 25 26 27      18 19 20 21 22 23 24      16 17 18 19 20 21 22
28 29 30 31               25 26 27 28 29 30         23 24 25 26 27 28 29
                                                    30 31


1. Calculator
2. Year-wise Calendar
3. Exit
Enter your choice: 3
See you later.
```

PROGRAM: 4

AIM: To develop a program to find the resolution of an given Image.

PYTHON CODE:

```
def find_res(filename):
    with open(filename,'rb') as img_file: # open image in binary mode
        # height of image is at 164th position
        img_file.seek(163)
        # read the 2 bytes
        a = img_file.read(2)
        # calculate height
        height = (a[0] << 8) + a[1]
        # read next 2 bytes which stores the width
        a = img_file.read (2)
        # calculate width
        width = (a[0] << 8) + a[1]
    print ("IMAGE RESOLUTION SI : ",width, "x", height)
find_res("/Users/radhikakolavali/python/pythonimage.jpeg")
```

OUTPUT DATA:

```
IMAGE RESOLUTION SI :  8704 x 769
```

THANK YOU ALL