Div: B **Name: Siddhant Badadhe** **Roll No: 122**

Dr. D. Y. Patil Pratishthan's

D. Y. Patil Institute of Master of Computer Applications and Management

(Approved by AICTE, New Delhi & Affiliated to Savitribai Phule Pune University)

Dr. D. Y. Patil Educational Complex, Sector 29, Pradhikaran, Akurdi, Pune – 411 044

Tel No: (020)27640998, Website:www.dypimca.ac.in, E-mail : director@dypimca.ac.in

PART B

Sr.No	**Topic**	**DATE**	**Pg.No**	**Sign**
1	Write python code to display user defined exception for the condition" person having age less than 18 years i.e; voting age".			
2	Write a Python program to calculate the factorial of a number using multiple threads.			
3	Write python program for +, -, operator overloading(take user input)			
4	Write a Python program that creates two threads to find and print even and odd numbers from 30 to 50			
5	Write python program to create superclass named "Shapes" has a methodcalled "area()". Subclasses of "Shapes" can be "Triangle", "circle", "Rectangle", etc. Each subclass has its way of calculating area. Using Inheritance and Polymorphism means, the subclasses can use the "area()"method to find the area's formula for that shape.			
6	Write a python code to validate email id using regular expression			
7	Write a python code to validate URL using regular expression			
8	Write a python program to use following , methods of string: isdigit(), capitalize(), casefold(), encode(), isidentifier(),			
9	Write a python program which finds the maximum number from num1 tonum2 (num2 inclusive) based on the following rules. 1. Always num1 should be less than num2			

	2. Consider each number from num1 to num2 (num2 inclusive). Populate the number into a list, if the below conditions are satisfied a. Sum of the digits of the number is a multiple of 3 b. Number has only two digits c. Number is a multiple of 5 3. Display the maximum element from the list In case of any invalid data or if the list is empty, display -1.			
10	Write a python program to find and display the product of three positive integer values based on the rule mentioned below: It should display the product of the three values except when one ofthe integer value is 7. In that case, 7 should not be included in the product and the values to its left also should not be included. If there is only one value to be considered, display that value itself.If no values can be included in the product, display -1.			
11	Write a python program to solve following problem statement:You have x no. of 5 rupee coins and y no. of 1 rupee coins. You want to purchase an item for amount z. The shopkeeper wants you to provide exact change. Youwant to pay using minimum number of coins. How many 5 rupee coins and1 rupee coins will you use? If exact change is not possible then display -1			
12	**An organization has decided to provide salary hike to its employeesbased on their job level.** Employees can be in job levels 3 , 4 or 5. Hike percentage based on job levels are given below: Job level Hike Percentage (applicable on current salary)3 15 4 7 5 5 In case of invalid job level, consider hike percentage to be 0. Given the current salary and job level, write a python program to find and display the new salary of an employee.**(Note : You have touse Class , functions and exception handling concepts)**			
13	**FoodCorner home delivers vegetarian and non-vegetarian combos to itscustomer based on order.**			

A vegetarian combo costs Rs.120 per plate and a non-vegetarian combo costs Rs.150 per plate. Their non-veg combo is really famous that they getmore orders for their non-vegetarian combo than the vegetarian combo.
Apart from the cost per plate of food, customers are also charged for homedelivery based on the distance in kms from the restaurant to the delivery point. The delivery charges are as mentioned below:

Distance in kms	Delivery charge in Rs
per kmFor first 3kms	0
For next 3kms	3
For the remaining	6

Given the type of food, quantity (no. of plates) and the distancein kms from the restaurant to the delivery point, write a pythonprogram to calculate the final bill amount to be paid by a customer.

The below information must be used to check the validity of thedata provided by the customer:

Type of food must be 'V' for vegetarian and 'N' for non-vegetarian.
Distance in kms must be greater than 0.Quantity ordered should be minimum 1.
If any of the input is invalid, the bill amount should beconsidered as -1.
(Note : You have to use Class and functions , exception handlingconcepts)

14 **A traveler on a visit to India is in need of some Indian Rupees (INR)but he has money belonging to another currency.**

He wants to know how much money he should provide in thecurrency he has, to get the specified amount in INR.
Write a python program to implement a currency calculator which accepts the amount needed in INR and the name of the currency which the traveler has. The program should identify and display theamount the traveler should provide in the currency he has, to get thespecified amount in INR.
Note: Use the forex information provided in the table below for the calculation. Consider that only the currency names mentioned in thetable are valid. For any invalid currency name, display -1.

Currency	Equivalent of 1.00 INR
Euro	0.01417
British Pound	0.0100
Australian Dollar	0.02140
Canadian Dollar	0.02027

(Note : You have to use Class and functions , exception handlingconcepts)

15	Write a program to print following pattern (Make program dynamic, takelength and breadth from user)
	`********`
	`*      *`
	`*      *`
	`*      *`
	`*      *`
	`********`			

16	Write a program to print following pattern 1 1 2 1 1 2 3 2 1 1 2 3 4 3 2 1			
17	Write a program to print following pattern 1 2 4 2 2 4 6 8 10 2 4 6 8 10 12 14			
18	Write a Python program to create a calculator class. Include methods forbasic arithmetic operations.			
19	Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the totalprice.(attributes item_id,item_name,quantity)			
20	Write python program to perform file handling and perform following operations: open,read,write,tell,seek,rename,delete			
21	Write python program to demonstrate thread synchronization and resolvethe race condition.			

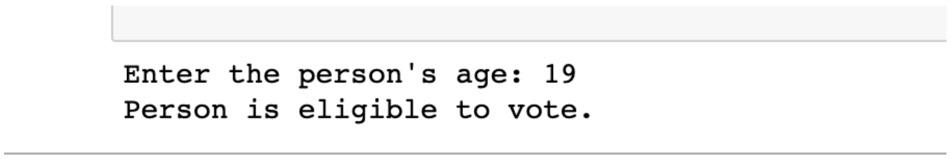
Div: B **Name: Siddhant Badadhe** **Roll No: 122**

22	Write a Python class Employee with attributes like emp_id, emp_name, emp_salary, and emp_department and methods like calculate_emp_salary,emp_assign_department, and print_employee_details. Sample Employee Data: "ADAMS", "E7876", 50000, "ACCOUNTING""JONES", "E7499", 45000, "RESEARCH" "MARTIN", "E7900", 50000, "SALES" "SMITH", "E7698", 55000, "OPERATIONS" • Use 'assign_department' method to change the department of anemployee. • Use 'print_employee_details' method to print the details of anemployee. • Use 'calculate_emp_salary' method takes two arguments: salary andhours_worked, which is the number of hours worked by the employee. If the number of hours worked is more than 50, the method computes overtime and adds it to the salary. Overtime is calculated as following formula: overtime = hours_worked - 50 Overtime amount = (overtime * (salary / 50))			
23	Write a Python class BankAccount with attributes like account_number, balance, date_of_opening and customer_name, and methods like deposit,withdraw, and check_balance.			
24	Create a class employee with attribute name and base_pay method get_pay() which shouldr eturn base_pay.Class SalesEmployee with attributename, base_pay, Sales_incentives inherit employee class over-rides get_pay() and return addition of base_pay and sales_incentive			

Div: B **Name: Siddhant Badadhe** **Roll No: 122**

1. **Write python code to display user defined exception for the condition" person having age less than 18 years i.e; voting age".**

```
class UnderAgeException(Exception):
 def __init__(self, age):
 self.age = age
super().__init__(f"UnderAgeException: Person's age is {self.age}. Voting age is 18 years.")

def check_voting_age(age):
 if age < 18:
    raise UnderAgeException(age)
 else:
    print("Person is eligible to vote.")

try:
   age = int(input("Enter the person's age: "))
   check_voting_age(age)
except ValueError:
   print("Invalid input. Please enter a valid age.")
except UnderAgeException as e:
   print(e)
```

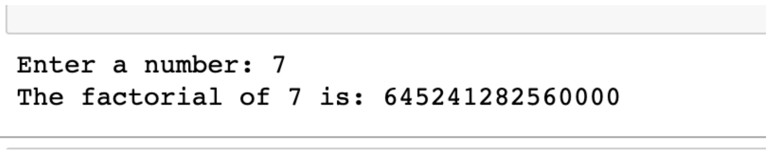


```
Enter the person's age: 19
Person is eligible to vote.
```

2. **Write a Python program to calculate the factorial of a number using multiple threads.**

```
import threading
class FactorialThread(threading.Thread):
   def __init__(self, number):
      threading.Thread.__init__(self)
      self.number = number
      self.result = 1

   def run(self):
      self.result = self.calculate_factorial(self.number)

   @staticmethod
   def calculate_factorial(number):
      result = 1
      for i in range(1, number + 1):
         result *= i
      return result

def calculate_factorial_with_threads(number):
   num_threads = 4  # Number of threads to use
   threads = []
```

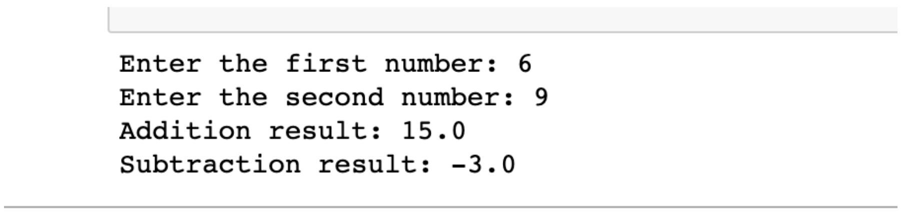
```
        # Split the range into equal segments for each thread
        segment_size = number // num_threads

        # Create and start the threads
        for i in range(num_threads):
            start = i * segment_size + 1
            end = start + segment_size - 1
            if i == num_threads - 1:
                end = number
            thread = FactorialThread(number)
            thread.start()
            threads.append(thread)

        # Wait for all threads to finish
        for thread in threads:
            thread.join()

        # Multiply the results of each thread to get the final factorial
        result = 1
        for thread in threads:
            result *= thread.result

        return result

    # Get the input from the user
    number = int(input("Enter a number: "))

    # Calculate factorial using multiple threads
    factorial = calculate_factorial_with_threads(number)

    print(f"The factorial of {number} is: {factorial}")
```



```
Enter a number: 7
The factorial of 7 is: 645241282560000
```

3. **Write python program for +, -, operator overloading(take user input)**

```
class Number:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        if isinstance(other, Number):
            return Number(self.value + other.value)
        else:
            return Number(self.value + other)
    def __sub__(self, other):
        if isinstance(other, Number):
            return Number(self.value - other.value)
        else:
            return Number(self.value - other)

    def __str__(self):
        return str(self.value)

# Get user input
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Create Number objects
number1 = Number(num1)
number2 = Number(num2)

# Perform addition
result_add = number1 + number2
print(f"Addition result: {result_add}")

# Perform subtraction
result_sub = number1 - number2
print(f"Subtraction result: {result_sub}")
```

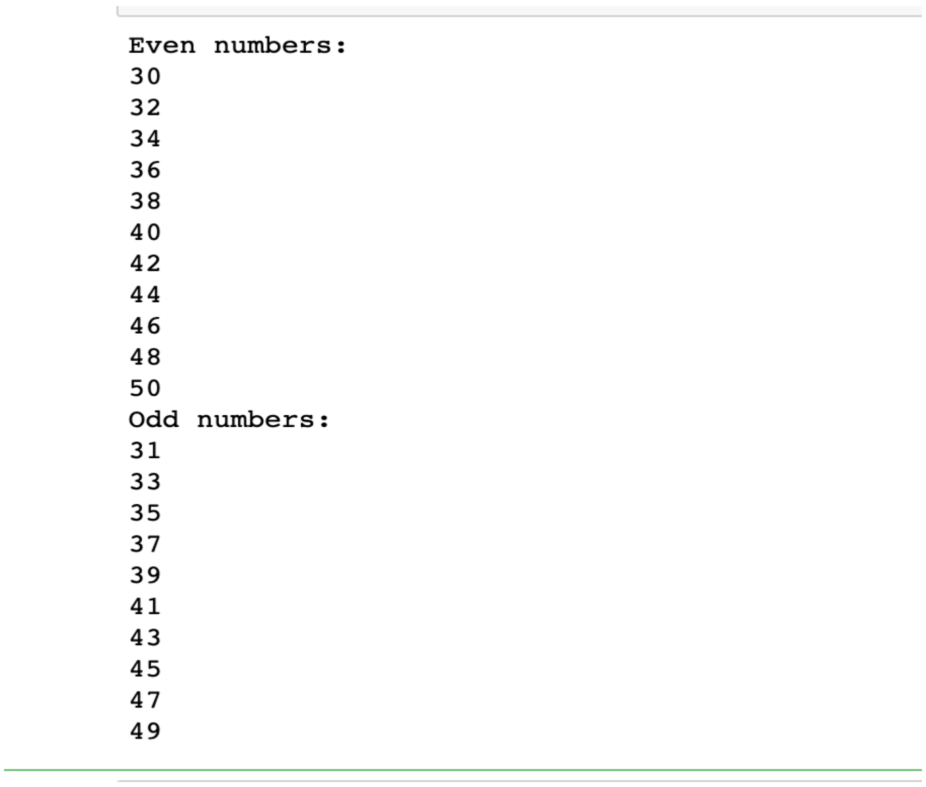


```
Enter the first number: 6
Enter the second number: 9
Addition result: 15.0
Subtraction result: -3.0
```

4. **Write a Python program that creates two threads to find and print even and odd numbers from 30 to 50.**

```
import threading
class EvenThread(threading.Thread):
    def run(self):
        print("Even numbers:")
        for num in range(30, 51):
            if num % 2 == 0:
                print(num)

class OddThread(threading.Thread):
    def run(self):
        print("Odd numbers:")
        for num in range(30, 51):
            if num % 2 != 0:
                print(num)
```

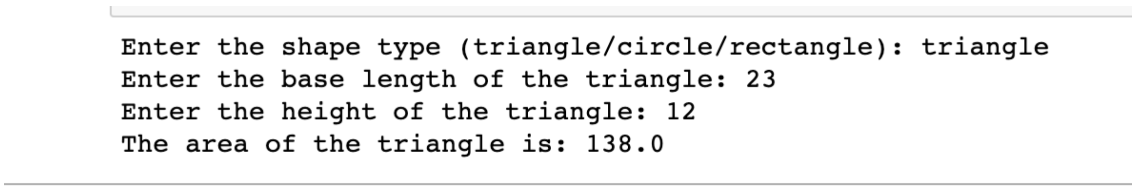
```
# Create the threads
even_thread = EvenThread()
odd_thread = OddThread()
        even_thread.start()
odd_thread.start()

even_thread.join()
odd_thread.join()
```



```
Even numbers:
30
32
34
36
38
40
42
44
46
48
50
Odd numbers:
31
33
35
37
39
41
43
45
47
49
```

5. **Write python program to create superclass named "Shapes" has a method called "area()". Subclasses of "Shapes" can be "Triangle", "circle", "Rectangle", etc. Each subclass has its way of calculating area. Using Inheritance and Polymorphism means, the subclasses can use the "area()" method to find the area's formula for that shape.**

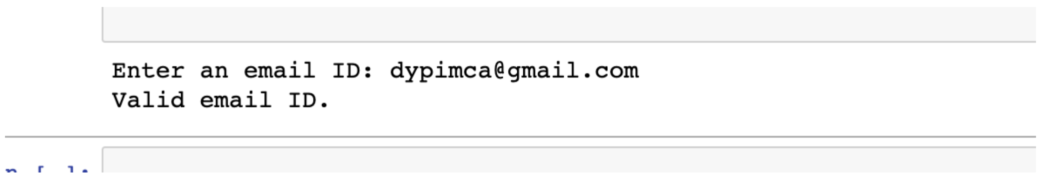
```
import math
class Shapes:
  def area(self):
    pass
class Triangle(Shapes):
  def __init__(self, base, height):
    self.base = base
    self.height = height
  def area(self):
    return 0.5 * self.base * self.height

class Circle(Shapes):
  def __init__(self, radius):
    self.radius = radius
    def area(self):
        return math.pi * self.radius**2

class Rectangle(Shapes):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width
shape_type = input("Enter the shape type (triangle/circle/rectangle): ")
if shape_type == "triangle":
    base = float(input("Enter the base length of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    shape = Triangle(base, height)
elif shape_type == "circle":
    radius = float(input("Enter the radius of the circle: "))
    shape = Circle(radius)
elif shape_type == "rectangle":
    length = float(input("Enter the length of the rectangle: "))
    width = float(input("Enter the width of the rectangle: "))
    shape = Rectangle(length, width)
else:
    print("Invalid shape type.")
    exit()
area = shape.area()
print(f"The area of the {shape_type} is: {area}")
```



```
Enter the shape type (triangle/circle/rectangle): triangle
Enter the base length of the triangle: 23
Enter the height of the triangle: 12
The area of the triangle is: 138.0
```

6. **Write a python code to validate email id using regular expression**

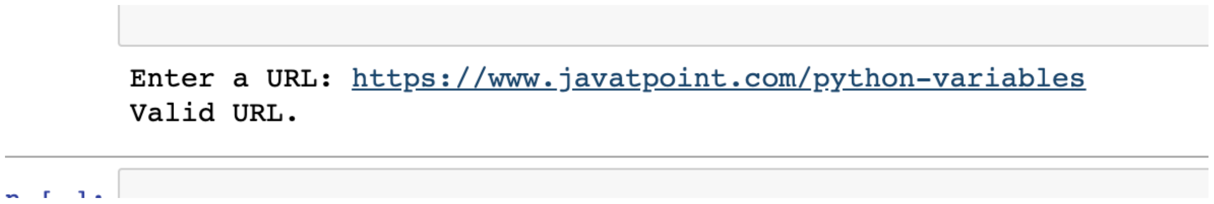
```
import re
def validate_email(email):
    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
    if re.match(pattern, email):
        return True
    else:
        return False
# Get the input from the user
email_id = input("Enter an email ID: ")

# Validate the email ID
if validate_email(email_id):
    print("Valid email ID.")
else:
    print("Invalid email ID.")
```



```
Enter an email ID: dypimca@gmail.com
Valid email ID.
```

7. **Write a python code to validate URL using regular expression.**

```
import re
def validate_url(url):
    pattern = r'^(http|https)://[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+([/?].*)?$'
    if re.match(pattern, url):
        return True
    else:
        return False

# Get the input from the user
url = input("Enter a URL: ")

# Validate the URL
if validate_url(url):
    print("Valid URL.")
else:
    print("Invalid URL.")
```

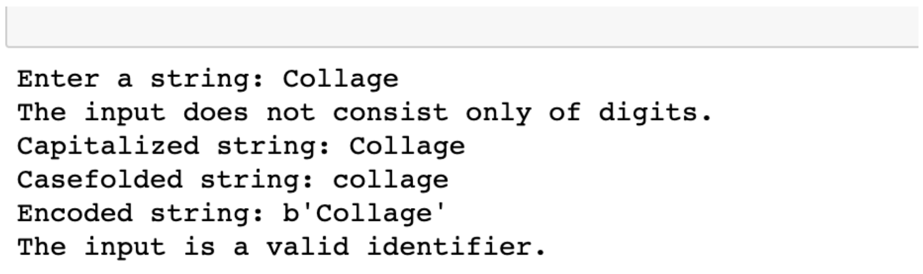


```
Enter a URL: https://www.javatpoint.com/python-variables
Valid URL.
```

8. **Write a python program to use following , methods of string: isdigit(), capitalize(), casefold(), encode(), isidentifier(),**

```
string = input("Enter a string: ")
if string.isdigit():
    print("The input consists only of digits.")
else:
    print("The input does not consist only of digits.")

# Using capitalize()
capitalized_string = string.capitalize()
print("Capitalized string:", capitalized_string)

# Using casefold()
casefolded_string = string.casefold()
print("Casefolded string:", casefolded_string)
```

```
encoded_string = string.encode()
print("Encoded string:", encoded_string)

if string.isidentifier():
    print("The input is a valid identifier.")
else:
    print("The input is not a valid identifier.")
```



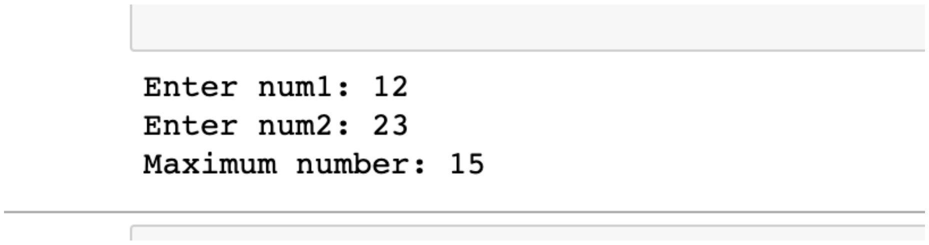
```
Enter a string: Collage
The input does not consist only of digits.
Capitalized string: Collage
Casefolded string: collage
Encoded string: b'Collage'
The input is a valid identifier.
```

9. **Write a python program which finds the maximum number from num1 to num2 (num2 inclusive) based on the following rules.**
   1. **Always num1 should be less than num2**

   2. Consider each number from num1 to num2 (num2 inclusive). Populate the number into a list, if the below conditions are satisfied
   a. Sum of the digits of the number is a multiple of 3
   b. Number has only two digits
   c. Number is a multiple of 5
   3. Display the maximum element from the list In case of any invalid data or if the list is empty, display -1.

```
def get_max_number(num1, num2):
    # Check if num1 is less than num2
    if num1 >= num2:
        return -1

    numbers = []

    # Iterate through each number from num1 to num2 (inclusive)
    for num in range(num1, num2 + 1):
        # Check if the number satisfies the conditions
        if sum(int(digit) for digit in str(num)) % 3 == 0 and len(str(num)) == 2 and num % 5
== 0:
            numbers.append(num)

    # Check if the list is empty
    if not numbers:
        return -1

    # Return the maximum element from the list
    return max(numbers)

# Get input from the user
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))
```

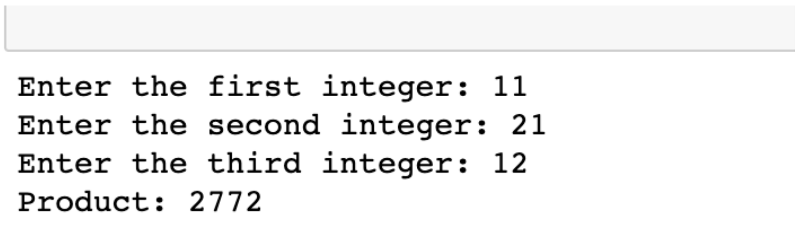
```
# Get the maximum number based on the given rules
max_number = get_max_number(num1, num2)

# Display the result
print("Maximum number:", max_number)
```



```
Enter num1: 12
Enter num2: 23
Maximum number: 15
```

10. **Write a python program to find and display the product of three positive integer values based on the rule mentioned below: It should display the product of the three values except when one of the integer value is 7. In that case, 7 should not be included in the product and the values to its left also should not be included. If there is only one value to be considered, display that value itself. If no values can be included in the product, display -1**

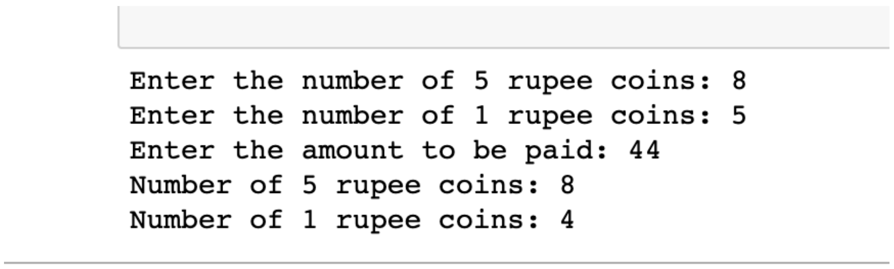
```
def calculate_product(a, b, c):
    if a == 7:
        return -1
    elif b == 7:
        return a
    elif c == 7:
        return a * b
    else:
        return a * b * c
a = int(input("Enter the first integer: "))
b = int(input("Enter the second integer: "))
c = int(input("Enter the third integer: "))
product = calculate_product(a, b, c)

# Display the result
print("Product:", product)
```



```
Enter the first integer: 11
Enter the second integer: 21
Enter the third integer: 12
Product: 2772
```

11. **Write a python program to solve following problem statement:You have x no. of 5 rupee coins and y no. of 1 rupee coins. You want to purchase an item for amount z. The shopkeeper wants you to provide exact change. You want to pay using minimum number of coins. How many 5 rupee coins and 1 rupee coins will you use? If exact change is not possible then display -1**

```
def calculate_change(x, y, z):
    total_amount = (x * 5) + y

    # Check if total_amount is less than z
    if total_amount < z:
        return -1

    # Calculate the number of 5 rupee coins needed
    num_5_rupee_coins = min(x, z // 5)

    # Calculate the remaining amount after using 5 rupee coins
    remaining_amount = z - (num_5_rupee_coins * 5)
     num_1_rupee_coins = min(y, remaining_amount)

    # Check if exact change is possible
    if (num_5_rupee_coins * 5) + num_1_rupee_coins != z:
        return -1

    return num_5_rupee_coins, num_1_rupee_coins

# Get input from the user
x = int(input("Enter the number of 5 rupee coins: "))
y = int(input("Enter the number of 1 rupee coins: "))
z = int(input("Enter the amount to be paid: "))

change = calculate_change(x, y, z)

# Display the result
if change == -1:
    print("Exact change is not possible.")
else:
    num_5_rupee_coins, num_1_rupee_coins = change
    print("Number of 5 rupee coins:", num_5_rupee_coins)
    print("Number of 1 rupee coins:", num_1_rupee_coins)
```



```
Enter the number of 5 rupee coins: 8
Enter the number of 1 rupee coins: 5
Enter the amount to be paid: 44
Number of 5 rupee coins: 8
Number of 1 rupee coins: 4
```

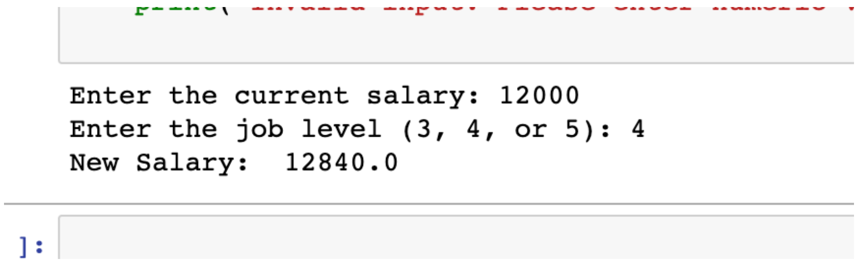
12. An organization has decided to provide salary hike to its employees based on their job level. Employees can be in job levels 3 , 4 or 5. Hike percentage based on job levels are given below:

Job level	**Hike Percentage (applicable on current salary)**
3	**15**
4	**7**
5	**5**

In case of invalid job level, consider hike percentage to be 0.
Given the current salary and job level, write a python program to find and display the

new salary of an employee.(Note : You have to use Class , functions and exception handling concepts)

```
class SalaryCalculator:
    def __init__(self, current_salary, job_level):
        self.current_salary = current_salary
        self.job_level = job_level

    def calculate_new_salary(self):
        try:
            job_level = int(self.job_level)
            if job_level == 3:
                hike_percentage = 15
            elif job_level == 4:
                hike_percentage = 7
            elif job_level == 5:
                hike_percentage = 5
            else:
                hike_percentage = 0

            hike_amount = (self.current_salary * hike_percentage) / 100
            new_salary = self.current_salary + hike_amount

            return new_salary

        except ValueError:
            return -1
current_salary = float(input("Enter the current salary: "))
job_level = input("Enter the job level (3, 4, or 5): ")

# Create an instance of the SalaryCalculator class
calculator = SalaryCalculator(current_salary, job_level)

# Calculate the new salary
new_salary = calculator.calculate_new_salary()

# Display the result
if new_salary == -1:
    print("Invalid job level. Please enter a valid job level (3, 4, or 5).")
else:
    print("New Salary:", new_salary)
```



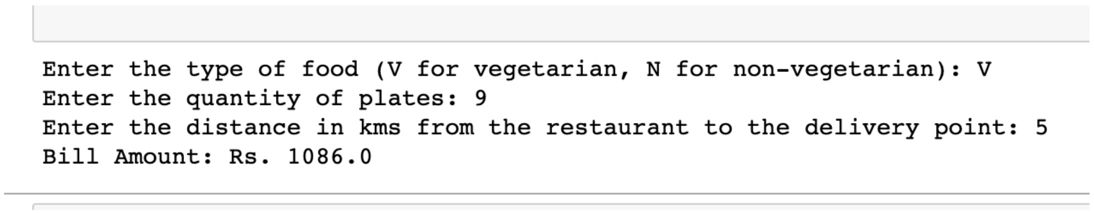
```
Enter the current salary: 12000
Enter the job level (3, 4, or 5): 4
New Salary:  12840.0
```

Div: B **Name: Siddhant Badadhe** **Roll No: 122**

13. Food Corner home delivers vegetarian and non-vegetarian combos to its customer based on order.
A vegetarian combo costs Rs.120 per plate and a non-vegetarian combo costs Rs.150 per plate. Their non-veg combo is really famous that they get more orders for their non-vegetarian combo than the vegetarian combo. Apart from the cost per plate of food, customers are also charged for home delivery based on the distance in kms from the restaurant to the delivery point. The delivery charges are as mentioned below: Distance in kms Delivery charge in Rs per km For first 3kms 0 For next 3kms 3 For the remaining 6 Given the type of food, quantity (no. of plates) and the distance in kms from the restaurant to the delivery point, write a python program to calculate the final bill amount to be paid by a customer. The below information must be used to check the validity of the data provided by the customer: Type of food must be 'V' for vegetarian and 'N' for nonvegetarian. Distance in kms must be greater than 0. Quantity ordered should be minimum 1. If any of the input is invalid, the bill amount should be considered as -1. (Note : You have to use Class and functions , exception handling concepts).

```
class FoodCorner:
    def __init__(self, food_type, quantity, distance):
        self.food_type = food_type
        self.quantity = quantity
        self.distance = distance

    def calculate_bill_amount(self):
        try:
            if self.food_type not in ['V', 'N'] or self.distance <= 0 or self.quantity < 1:
                return -1

            veg_combo_cost = 120
            non_veg_combo_cost = 150
            delivery_charge = 0

            if self.distance > 3:
                delivery_charge += 3 * min(3, self.distance - 3)
            if self.distance > 6:
                delivery_charge += 6 * min(6, self.distance - 6)

            if self.food_type == 'V':
                total_cost = veg_combo_cost * self.quantity + delivery_charge
            else:
                total_cost = non_veg_combo_cost * self.quantity + delivery_charge

            return total_cost

        except TypeError:
            return -1


# Get input from the user
food_type = input("Enter the type of food (V for vegetarian, N for non-vegetarian): ")
quantity = int(input("Enter the quantity of plates: "))
distance = float(input("Enter the distance in kms from the restaurant to the delivery point: "))

# Create an instance of the FoodCorner class
order = FoodCorner(food_type, quantity, distance)
```

```
# Calculate the bill amount
bill_amount = order.calculate_bill_amount()

# Display the result
if bill_amount == -1:
   print("Invalid input. Please provide valid data.")
else:
   print("Bill Amount: Rs.", bill_amount)
```



```
Enter the type of food (V for vegetarian, N for non-vegetarian): V
Enter the quantity of plates: 9
Enter the distance in kms from the restaurant to the delivery point: 5
Bill Amount: Rs. 1086.0
```

14. **A traveller on a visit to India is in need of some Indian Rupees (INR) but he has money belonging to another currency.**
15. **He wants to know how much money he should provide in the currency he has, to get the specified amount in INR. Write a python program to implement a currency calculator which accepts the amount needed in INR and the name of the currency which the traveler has. The program should identify and display the amount the traveler should provide in the currency he has, to get the specified amount in INR. Note: Use the forex information provided in the table below for the calculation. Consider that only the currency names mentioned in the table are valid. For any invalid currency name, display -1.**

Currency	**Equivalent of 1.00 INR**
Euro	**0.01417**
British Pound	**0.0100**
Australian Dollar	**0.02140**
Canadian Dollar	**0.02027**

(Note : You have to use Class and functions , exception handling concepts).

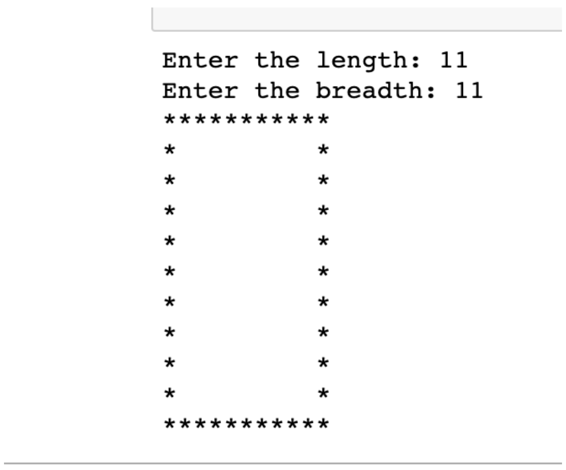
```
class CurrencyCalculator:
   forex_rates = {
      "Euro": 0.01417,
      "British Pound": 0.0100,
      "Australian Dollar": 0.02140,
      "Canadian Dollar": 0.02027
   }

   def __init__(self, amount_needed_inr, currency_name):
      self.amount_needed_inr = amount_needed_inr
      self.currency_name = currency_name

   def calculate_amount_needed(self):
      try:
         forex_rate = self.forex_rates.get(self.currency_name)
```

```
            if forex_rate is None:
                return -1

            amount_needed = self.amount_needed_inr / forex_rate

            return amount_needed

        except TypeError:
            return -1


# Get input from the user
amount_needed_inr = float(input("Enter the amount needed in INR: "))
currency_name = input("Enter the currency name (Euro, British Pound, Australian Dollar,
Canadian Dollar): ")

# Create an instance of the CurrencyCalculator class
calculator = CurrencyCalculator(amount_needed_inr, currency_name)

# Calculate the amount needed in the specified currency
amount_needed = calculator.calculate_amount_needed()

# Display the result
if amount_needed == -1:
    print("Invalid currency name. Please enter a valid currency name.")
else:
    print("Amount needed in", currency_name + ":", amount_needed)
```

15. Write a program to print following pattern (Make program dynamic, take length and breadth from user)

```
******
*.    *
*.    *
*.    *
* * * *
```

```
def print_pattern(length, breadth):
    for i in range(breadth):
      for j in range(length):
        if i == 0 or i == breadth - 1 or j == 0 or j == length - 1:
                print("*", end="")
else:
print(" ", end="")
print()
length = int(input("Enter the length: "))
breadth = int(input("Enter the breadth: "))
print_pattern(length, breadth)
```

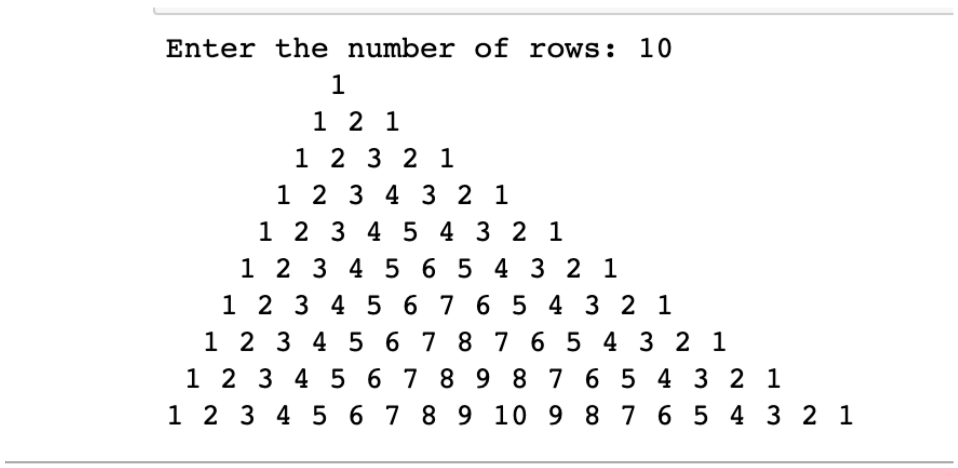


```
Enter the length: 11
Enter the breadth: 11
***********
*         *
*         *
*         *
*         *
*         *
*         *
*         *
*         *
*         *
***********
```

16. Write a program to print following pattern

```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

```
def print_pattern(rows):
    for i in range(1, rows + 1):
        # Print spaces before each number
        print(" " * (rows - i), end="")

        # Print numbers in ascending order
        for j in range(1, i + 1):
            print(j, end=" ")

        # Print numbers in descending order
        for k in range(i - 1, 0, -1):
            print(k, end=" ")

        print()
# Get input from the user
rows = int(input("Enter the number of rows: "))

# Print the pattern
print_pattern(rows)
```

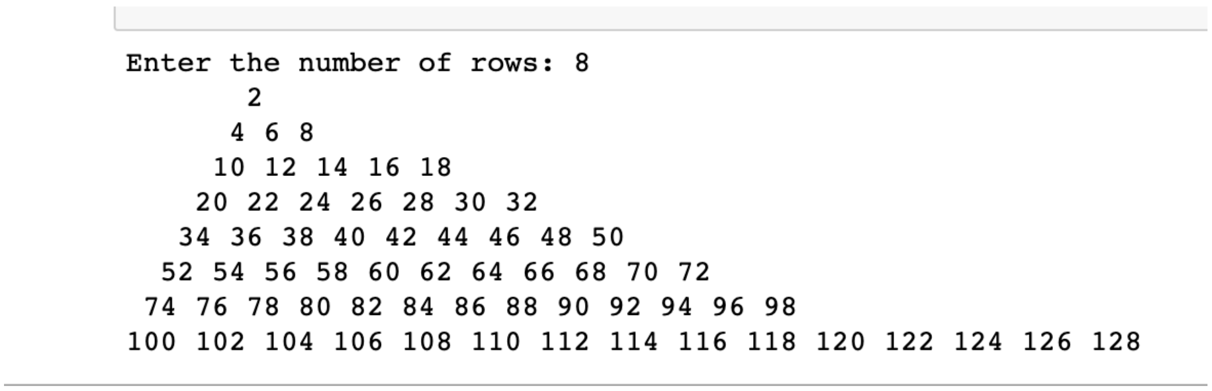


```
Enter the number of rows: 10
         1
        1 2 1
       1 2 3 2 1
      1 2 3 4 3 2 1
     1 2 3 4 5 4 3 2 1
    1 2 3 4 5 6 5 4 3 2 1
   1 2 3 4 5 6 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1
```

17. Write a program to print following pattern

```
      1
     2 4 2
   2 4 6 8 10
 2 4 6 8 10 12 14
```

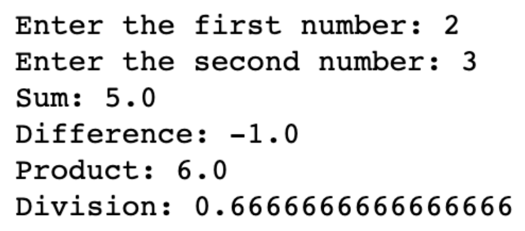
```
def print_pattern(rows):
    num = 2  # Starting number
    for i in range(1, rows + 1):
        # Print spaces before each number
        print(" " * (rows - i), end="")

        # Print numbers in increments of 2
        for j in range(1, 2 * i):
            print(num, end=" ")
            num += 2

        print()


# Get input from the user
rows = int(input("Enter the number of rows: "))

# Print the pattern
print_pattern(rows)
```



```
Enter the number of rows: 8
       2
      4 6 8
     10 12 14 16 18
    20 22 24 26 28 30 32
   34 36 38 40 42 44 46 48 50
  52 54 56 58 60 62 64 66 68 70 72
 74 76 78 80 82 84 86 88 90 92 94 96 98
100 102 104 106 108 110 112 114 116 118 120 122 124 126 128
```

18. Write a Python program to create a calculator class. Include methods for basic arithmetic operations.

```
class Calculator:
    def add(self, num1, num2):
        return num1 + num2

    def subtract(self, num1, num2):
        return num1 - num2

    def multiply(self, num1, num2):
        return num1 * num2

    def divide(self, num1, num2):
        if num2 != 0:
            return num1 / num2
        else:
            return "Error: Cannot divide by zero."


# Create an instance of the Calculator class
calculator = Calculator()

# Get input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Perform arithmetic operations using the Calculator methods
sum_result = calculator.add(num1, num2)
difference_result = calculator.subtract(num1, num2)
product_result = calculator.multiply(num1, num2)
division_result = calculator.divide(num1, num2)

# Display the results
print("Sum:", sum_result)
print("Difference:", difference_result)
print("Product:", product_result)
print("Division:", division_result)
```



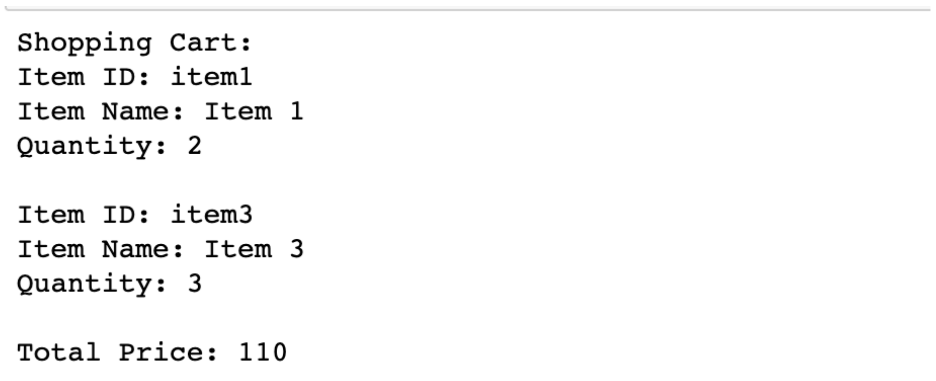
```
Enter the first number: 2
Enter the second number: 3
Sum: 5.0
Difference: -1.0
Product: 6.0
Division: 0.6666666666666666
```

Div: B **Name: Siddhant Badadhe** **Roll No: 122**

19. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.(attributes item_id,item_name,quantity)

```
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, item_id, item_name, quantity):
        item = {
            'item_id': item_id,
            'item_name': item_name,
            'quantity': quantity
        }
        self.items.append(item)

    def remove_item(self, item_id):
        for item in self.items:
            if item['item_id'] == item_id:
                self.items.remove(item)
                break

    def calculate_total_price(self):
        total_price = 0
        for item in self.items:
            # Assuming each item has a price attribute
            price = self.get_item_price(item['item_id'])
            total_price += price * item['quantity']
        return total_price

    # Example method to get the price of an item
    def get_item_price(self, item_id):
        prices = {
            'item1': 10,
            'item2': 20,
            'item3': 30
        }
        return prices.get(item_id, 0)


# Create an instance of the ShoppingCart class
cart = ShoppingCart()

# Add items to the cart
cart.add_item('item1', 'Item 1', 2)
cart.add_item('item2', 'Item 2', 1)
cart.add_item('item3', 'Item 3', 3)

# Remove an item from the cart
cart.remove_item('item2')

# Calculate the total price
total_price = cart.calculate_total_price()

# Display the cart contents and total price
```

```
print("Shopping Cart:")
for item in cart.items:
   print("Item ID:", item['item_id'])
   print("Item Name:", item['item_name'])
   print("Quantity:", item['quantity'])
   print()

print("Total Price:", total_price)
```

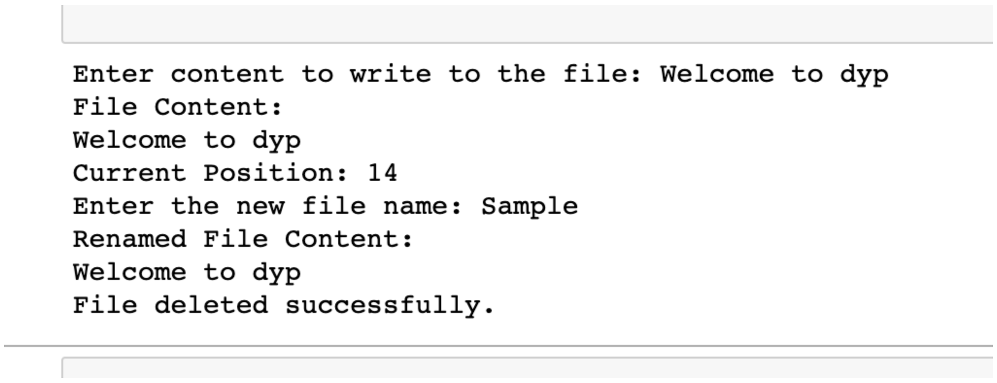


```
Shopping Cart:
Item ID: item1
Item Name: Item 1
Quantity: 2

Item ID: item3
Item Name: Item 3
Quantity: 3

Total Price: 110
```

20. Write python program to perform file handling and perform following operations: open,read,write,tell,seek,rename,delete

```
import os

# Open a file in write mode
file_name = "demo.txt"
file = open(file_name, "w")

# Write content to the file
content = input("Enter content to write to the file: ")
file.write(content)

# Close the file
file.close()

# Open the file in read mode
file = open(file_name, "r")

# Read and display the content of the file
file_content = file.read()
print("File Content:")
print(file_content)

# Get the current position (byte) in the file
position = file.tell()
print("Current Position:", position)

# Seek to the beginning of the file
file.seek(0)
```

```
# Rename the file
new_file_name = input("Enter the new file name: ")
os.rename(file_name, new_file_name)

# Close the file
file.close()

file = open(new_file_name, "r")

file_content = file.read()
print("Renamed File Content:")
print(file_content)

# Close the file
file.close()

# Delete the file
os.remove(new_file_name)
print("File deleted successfully.")
```

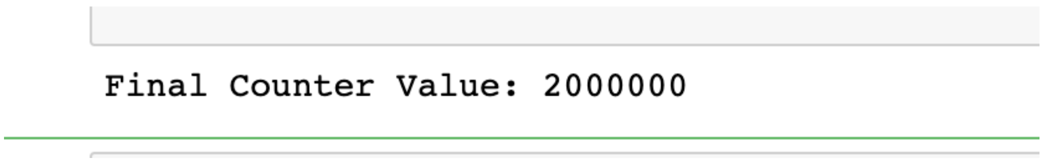


```
Enter content to write to the file: Welcome to dyp
File Content:
Welcome to dyp
Current Position: 14
Enter the new file name: Sample
Renamed File Content:
Welcome to dyp
File deleted successfully.
```

21. Write python program to demonstrate thread synchronization and resolve the race condition.

```
import threading
# Shared counter variable
counter = 0

# Lock for thread synchronization
lock = threading.Lock()
def increment_counter():
    global counter
    for _ in range(1000000):
        lock.acquire()
        counter += 1
        lock.release()

# Create two threads
thread1 = threading.Thread(target=increment_counter)
thread2 = threading.Thread(target=increment_counter)

# Start the threads
```

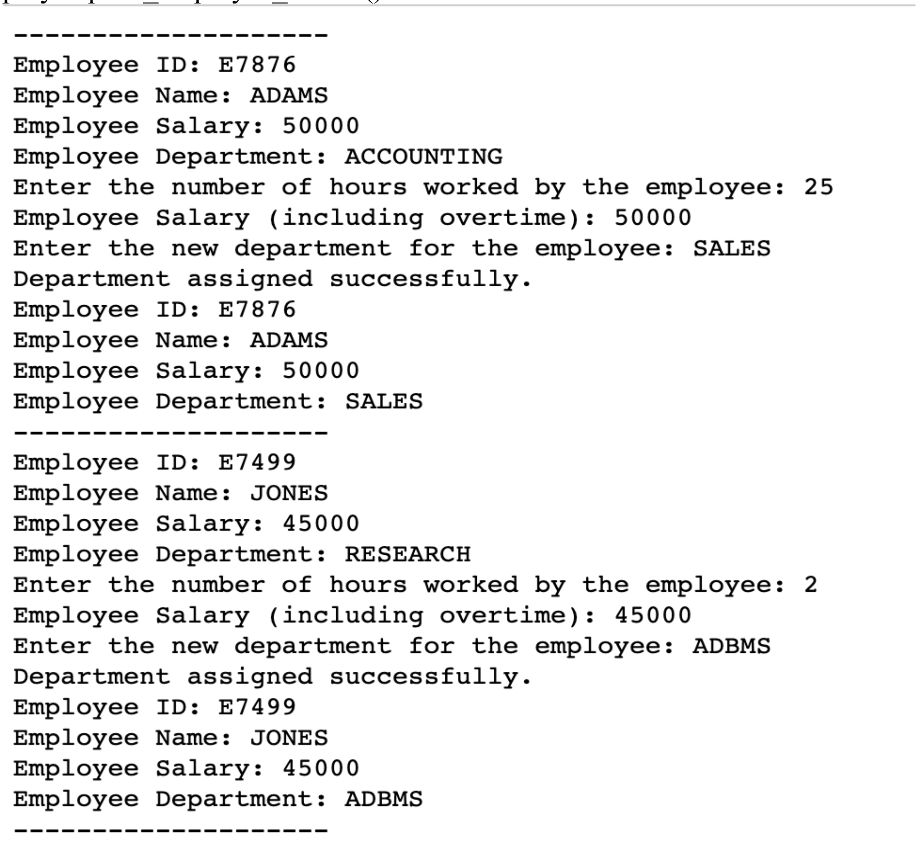
```
thread1.start()
thread2.start()
# Wait for the threads to finish
thread1.join()
thread2.join()

# Display the final value of the counter
print("Final Counter Value:", counter)
```



```
Final Counter Value: 2000000
```

22. Write a Python class Employee with attributes like emp_id, emp_name, emp_salary, and emp_department and methods like calculate_emp_salary, emp_assign_department, and print_employee_details. Sample Employee Data:
"ADAMS", "E7876", 50000, "ACCOUNTING"
"JONES", "E7499", 45000, "RESEARCH"
"MARTIN", "E7900", 50000, "SALES"
"SMITH", "E7698", 55000, "OPERATIONS"
● Use 'assign_department' method to change the department of an employee.
● Use 'print_employee_details' method to print the details of an employee.
● Use 'calculate_emp_salary' method takes two arguments: salary and hours_worked, which is the number of hours worked by the employee. If the number of hours worked is more than 50, the method computes overtime and adds it to the salary.
Overtime is calculated as following formula:
overtime = hours_worked - 50
Overtime amount = (overtime * (salary / 50))

```
class Employee:
    def __init__(self, emp_id, emp_name, emp_salary, emp_department):
        self.emp_id = emp_id
        self.emp_name = emp_name
        self.emp_salary = emp_salary
        self.emp_department = emp_department

    def calculate_emp_salary(self, hours_worked):
        if hours_worked > 50:
            overtime = hours_worked - 50
            overtime_amount = overtime * (self.emp_salary / 50)
            return self.emp_salary + overtime_amount
        else:
            return self.emp_salary

    def emp_assign_department(self, new_department):
        self.emp_department = new_department
        print("Department assigned successfully.")

    def print_employee_details(self):
        print("Employee ID:", self.emp_id)
        print("Employee Name:", self.emp_name)
        print("Employee Salary:", self.emp_salary)
        print("Employee Department:", self.emp_department)

# Create a list of employee data
employee_data = [
    ("E7876", "ADAMS", 50000, "ACCOUNTING"),
    ("E7499", "JONES", 45000, "RESEARCH"),
    ("E7900", "MARTIN", 50000, "SALES"),
    ("E7698", "SMITH", 55000, "OPERATIONS")
]

# Create Employee objects from the data
employees = []
for data in employee_data:
    emp_id, emp_name, emp_salary, emp_department = data
    employee = Employee(emp_id, emp_name, emp_salary, emp_department)
    employees.append(employee)

# Display employee details
for employee in employees:
    print("--------------------")
    employee.print_employee_details()

    hours_worked = int(input("Enter the number of hours worked by the employee: "))
    emp_salary = employee.calculate_emp_salary(hours_worked)
    print("Employee Salary (including overtime):", emp_salary)

    new_department = input("Enter the new department for the employee: ")
    employee.emp_assign_department(new_department)
    employee.print_employee_details()
```

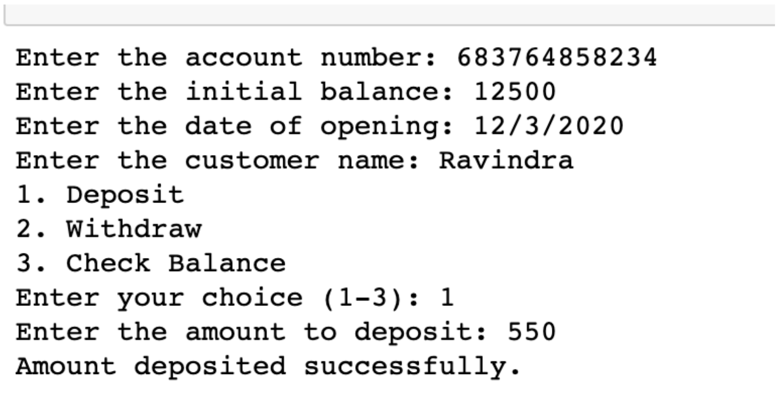


```
--------------------
Employee ID: E7876
Employee Name: ADAMS
Employee Salary: 50000
Employee Department: ACCOUNTING
Enter the number of hours worked by the employee: 25
Employee Salary (including overtime): 50000
Enter the new department for the employee: SALES
Department assigned successfully.
Employee ID: E7876
Employee Name: ADAMS
Employee Salary: 50000
Employee Department: SALES
--------------------
Employee ID: E7499
Employee Name: JONES
Employee Salary: 45000
Employee Department: RESEARCH
Enter the number of hours worked by the employee: 2
Employee Salary (including overtime): 45000
Enter the new department for the employee: ADBMS
Department assigned successfully.
Employee ID: E7499
Employee Name: JONES
Employee Salary: 45000
Employee Department: ADBMS
--------------------
```

Div: B **Name: Siddhant Badadhe** **Roll No: 122**

23. Write a Python class BankAccount with attributes like account_number, balance, date_of_opening and customer_name, and methods like deposit, withdraw, and check_balance.

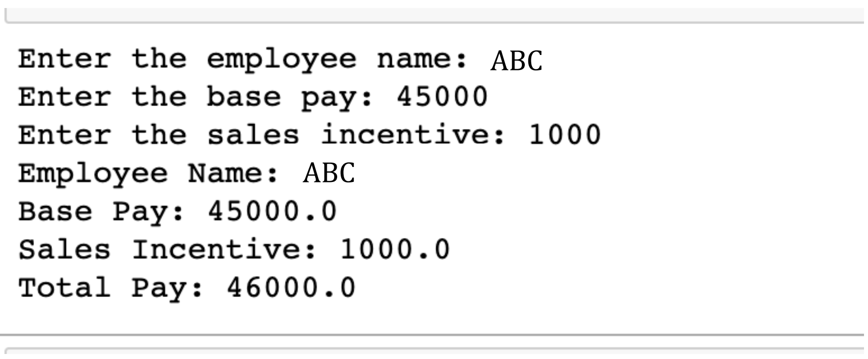
```
class BankAccount:
    def __init__(self, account_number, balance, date_of_opening, customer_name):
        self.account_number = account_number
        self.balance = balance
        self.date_of_opening = date_of_opening
        self.customer_name = customer_name

    def deposit(self, amount):
        self.balance += amount
        print("Amount deposited successfully.")

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print("Amount withdrawn successfully.")
        else:
            print("Insufficient balance.")

    def check_balance(self):
        print("Current balance:", self.balance)

account_number = input("Enter the account number: ")
balance = float(input("Enter the initial balance: "))
date_of_opening = input("Enter the date of opening: ")
customer_name = input("Enter the customer name: ")

account = BankAccount(account_number, balance, date_of_opening, customer_name)

print("1. Deposit")
print("2. Withdraw")
print("3. Check Balance")

choice = int(input("Enter your choice (1-3): "))

if choice == 1:
    amount = float(input("Enter the amount to deposit: "))
    account.deposit(amount)
elif choice == 2:
    amount = float(input("Enter the amount to withdraw: "))
    account.withdraw(amount)
elif choice == 3:
    account.check_balance()
else:
    print("Invalid choice.")
```



```
Enter the account number: 683764858234
Enter the initial balance: 12500
Enter the date of opening: 12/3/2020
Enter the customer name: Ravindra
1. Deposit
2. Withdraw
3. Check Balance
Enter your choice (1-3): 1
Enter the amount to deposit: 550
Amount deposited successfully.
```

24. Create a class employee with attribute name and base_pay method get_pay() which shouldr eturn base_pay.Class SalesEmployee with attribute name, base_pay, Sales_incentives inherit employee class over-rides get_pay() and return addition of base_pay and sales_incentive

```
class Employee:
    def __init__(self, name, base_pay):
        self.name = name
        self.base_pay = base_pay

    def get_pay(self):
        return self.base_pay

class SalesEmployee(Employee):
    def __init__(self, name, base_pay, sales_incentive):
        super().__init__(name, base_pay)
        self.sales_incentive = sales_incentive

    def get_pay(self):
        return self.base_pay + self.sales_incentive

name = input("Enter the employee name: ")
base_pay = float(input("Enter the base pay: "))
sales_incentive = float(input("Enter the sales incentive: "))

employee = SalesEmployee(name, base_pay, sales_incentive)

print("Employee Name:", employee.name)
print("Base Pay:", employee.base_pay)
print("Sales Incentive:", employee.sales_incentive)
print("Total Pay:", employee.get_pay())
```



```
Enter the employee name: ABC
Enter the base pay: 45000
Enter the sales incentive: 1000
Employee Name: ABC
Base Pay: 45000.0
Sales Incentive: 1000.0
Total Pay: 46000.0
```