SADAKOPA RAMAKRISHNAN T
31222225002109

CODE Q1:

Python

```
# -*- coding: utf-8 -*-
"""
 This module provides code for two classes. This is a part
 of the exercises given under the course UIT2312 (Programming
 and Design Patterns).

 In this source code I've executed my own logic and may contain bugs.
 The source code has followed good coding practices.

 Your comments and suggestions are welcome.

 Created on Fri Sept 15 2023

 Revised on Sat Sept 16 2023

 Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""


class Complex:
  '''Complex Number class which has two data members namely re and im
     and two member functions norm() and display()'''

  def __init__(self, re, im):
    '''
         Constructor to initialise data members
         '''
    self.re = re
    self.im = im

  def norm(self):
    '''
         Returns the norm value of a complex number
         '''
    return ((self.re * self. re) + (self.im * self.im)) ** 0.5

  def display(self):
    '''
         Displays the complex number along with its norm value
         '''
    string = "|" + str(self.re) + " + " + str(self.im) +"j |= " + str(self.norm())
    return string
```

SADAKOPA RAMAKRISHNAN T
31222225002109

```
class Rectangle:
  def __init__(self, length,breadth):
    """
          Constructor to initialise data members
          """
    self.length = length
    self.breadth = breadth

  def area(self):
    """
          Member function to calculate the area of rectangle
          """
    return self.length * self.breadth

  def perimeter(self):
    """
          Member function to calculate the perimeter of rectangle
          """
    return 2 * (self.length + self.breadth)

  def display(self):
    """
          Member function to display the rectangle class
          """
    result = f"The sides of rectangle are ({self.length}, {self.breadth}) and Area is
{self.area()}. The perimeter of the rectangle is {self.perimeter()}"
          return result

 #Test cases for the above code

 if __name__ == "__main__":
  #This part of the program will not be executed when the file is imported.

  c = Complex(3,4)
  print(c.display())

  R = Rectangle(5,4)
  print(R.display())
```

OUTPUT :

```
|3 + 4j |= 5.0
The sides of rectangle are (5, 4) and Area is 20. The perimeter of the rectangle is 18
```

SADAKOPA RAMAKRISHNAN T
31222225002109

CODE Q2.1:

Python

```
# -*- coding: utf-8 -*-
"""
This module provides code for Point class. This is a part
of the exercises given under the course UIT2312 (Programming
and Design Patterns).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Fri Sept 15 2023

Revised on Wed Sept 21 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""


class Point:
  def __init__(self, xcod = 0, ycod = 0):
    self.x = xcod
    self.y = ycod

  def getPoint(self):
    self.x = int(input("Enter x coordinate: "))
    self.y = int(input("Enter y coordinate: "))

  def showPoint(self):
    result = f"({self.x}, {self.y})"
        return result

if __name__ == "__main__":
  #Creating a Point Object
  P = Point()

  #Getting x and y coordinates
  P.getPoint()

  #Displaying x and y coordinates
  print(P.showPoint())
```

SADAKOPA RAMAKRISHNAN T
31222225002109

OUTPUT:

CODE Q2.2:

Python

```
# -*- coding: utf-8 -*-
"""
This module provides code for two classes Circle and Cone
which is inherited from Circle. This is a part
of the exercises given under the course UIT2312 (Programming
and Design Patterns).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Fri Sept 15 2023

Revised on Wed Sept 21 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""


class Circle:
  def __init__(self):
    self.radius = 0.0

  def getCircle(self):
    print("Enter the coordinates of Centre")
    x1 = int(input("Enter x coordinate of center: "))
    y1 = int(input("Enter y coordinate of center: "))

    print("Enter the coordinates of any point on circumference")
    x2 = int(input("Enter x coordinate of point: "))
    y2 = int(input("Enter y coordinate of point: "))

    self.calcRad(x1,y1,x2,y2)
```

```
    self.calcArea()

  def calcRad(self, x1, y1, x2, y2):
    self.radius = (((x2 - x1) ** 2) + ((y2 - y1) ** 2)) ** 0.5

  def calcArea(self):
    area = 3.14 * (self.radius ** 2)
    print(f"The radius of the circle is {self.radius}")
        print(f"Area of circle is {area}")

class Cone(Circle):
    def __init__(self):
        super().__init__()
        self.apex = None

    def getCone(self):
        self.getCircle()
        print("Enter the coordinates of apex")
    x = int(input("Enter the x coordinates of apex: "))
    y = int(input("Enter the y coordinate of apex: "))
    self.apex = (x, y)

  def calcVolume(self):
    height = ((self.radius ** 2) + (self.apex[1]**2)) ** 0.5
    volume = (1/3) * 3.14 * (self.radius**2) * (height)
    print(f"Apex Coordinates: {self.apex}")
        print(f"Height of the cone: {height}")
        print(f"Volume of the cone: {volume}")


if __name__ == "__main__":
  print("CIRCLE!!!")
  #Creating a circle object
  circle = Circle()

  #Printing details
  circle.getCircle()

  print("_____________________________________")
  print("CONE!!!")
  #Creating a Cone object
  cone = Cone()

  #Printing details
  cone.getCone()
```

```
cone.calcVolume()
```

OUTPUT:

```
CIRCLE!!!
Enter the coordinates of Centre
Enter x coordinate of center: 3
Enter y coordinate of center: 4
Enter the coordinates of any point on circumference
Enter x coordinate of point: 5
Enter y coordinate of point: 5
The radius of the circle is 2.23606797749979
Area of circle is 15.700000000000003

_______________________________
CONE!!!
Enter the coordinates of Centre
Enter x coordinate of center: 3
Enter y coordinate of center: 3
Enter the coordinates of any point on circumference
Enter x coordinate of point: 4
Enter y coordinate of point: 4
The radius of the circle is 1.4142135623730951
Area of circle is 6.280000000000002
Enter the coordinates of apex
Enter the x coordinates of apex: 1
Enter the y coordinate of apex: 9
Apex Coordinates: (1, 9)
Height of the cone: 9.1104335791443
Volume of the cone: 19.07117429234207
```

CODE Q2.3:

Python

```
# -*- coding: utf-8 -*-
"""
This module provides code for two classes Regular_Polygon
and Square which is inherited from Regular_Polygon. This is a part
of the exercises given under the course UIT2312 (Programming
```

```
and Design Patterns).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Fri Sept 15 2023

Revised on Wed Sept 21 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""


from ex02_1 import Point

class Regular_Polygon(Point):
 def __init__(self):
   super().__init__()
   self.points_array = []
   self.num_sides = 0

 def getPolygon(self):
   self.num_sides = int(input("Enter the number of sides: "))
   print("Enter the coordinates of polygon's vertices")
   for i in range(self.num_sides):
     point = Point()
     point.getPoint()
     self.points_array.append(point)

 def showPolygonDetails(self):
   print(f"Number of sides: {self.num_sides}")
      print("Coordinates of the polygon's vertices:")
      for i, point in enumerate(self.points_array, start=1):
          print(f"Vertex {i}: ({point.x}, {point.y})")

class Square(Regular_Polygon):
  def __init__(self):
      super().__init__()  # Call the constructor of the base class
(Regular_Polygon)
      self.side_length = 0

  def getSquareDetails(self):
```

```
      self.getPolygon()  # Reuse the getDetails method from the Regular_Polygon
class
      self.side_length = float(input("Enter the side length of the square: "))

  def calcArea(self):
      area = self.side_length ** 2
      return area

  def calcPerimeter(self):
      perimeter = self.side_length * self.num_sides
      return perimeter


if __name__ == "__main__":
 print("Polygon!!!")
 # Create an instance of the Regular_Polygon class
 polygon = Regular_Polygon()

 # Get details of the regular polygon
 polygon.getPolygon()

 # Display the polygon details
 polygon.showPolygonDetails()

 print("_____________________________________\n")
 print("SQUARE!!!")
 # Create an instance of the Square class
 square = Square()

 # Get details of the square
 square.getSquareDetails()

 # Display the square details
 square.showPolygonDetails()
 area = square.calcArea()
 perimeter = square.calcPerimeter()
 print(f"Area of the square: {area:.2f}")
  print(f"Perimeter of the square: {perimeter:.2f}")
```

OUTPUT:

```
Polygon!!!
Enter the number of sides: 3
Enter the coordinates of polygon's vertices
Enter x coordinate: 1
Enter y coordinate: 2
Enter x coordinate: 4
Enter y coordinate: 5
Enter x coordinate: 6
Enter y coordinate: 7
Number of sides: 3
Coordinates of the polygon's vertices:
Vertex 1: (1, 2)
Vertex 2: (4, 5)
Vertex 3: (6, 7)

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

SQUARE!!!
Enter the number of sides: 4
Enter the coordinates of polygon's vertices
Enter x coordinate: 1
Enter y coordinate: 1
Enter x coordinate: 0
Enter y coordinate: 0
Enter x coordinate: 0
Enter y coordinate: 1
Enter x coordinate: 1
Enter y coordinate: 0
Enter the side length of the square: 1
Number of sides: 4
Coordinates of the polygon's vertices:
Vertex 1: (1, 1)
Vertex 2: (0, 0)
Vertex 3: (0, 1)
Vertex 4: (1, 0)
Area of the square: 1.00
Perimeter of the square: 4.00
```

SADAKOPA RAMAKRISHNAN T
31222225002109

DATE PACKAGE:

CONVERT MODULE:

```python
class Convert:

 def Convert_hrs_days(eslf, hours):
  return hours / 24

 def Convert_days_hours(self, days):
  return days * 24

 def Convert_man_hrs_days(self, hours):
  return hours / 8
```

CURRENT MODULE:

```python
from datetime import datetime

class Current:

  def curr_time(self):
    return datetime.now().strftime("%H:%M:%S")

  def def_date(self):
    return datetime.today().strftime("%d.%m.%y")

  def dif_date(self):
    return datetime.today().strftime("%m.%d.%y")

  def str_time(self):
    return datetime.today().strftime("%m/%d/%Y, %H:%M:%S")
```

SADAKOPA RAMAKRISHNAN T
31222225002109

DATE MODULE:

```python
class Date:

 __slots__ = ["year", "month", "date"]

 def __init__(self, y=0, m=0, d=0):
   self.year = y
   self.month = m
   self.date = d

 def set_date(self, y, m, d):
   if len(y)==4 and y.isdigit():
     self.year = int(y)
   else:
     raise ValueError("Year must have 4 numbers")
   if len(m)==2 and m.isdigit():
     self.month = int(m)
   else:
     raise ValueError("Month must have 2 numbers")
   if len(d)==2 and d.isdigit():
     self.date = int(d)
   else:
     raise ValueError("Date must have 2 numbers")

 def show_date(self):
   print(f"{self.date}/{self.month}/{self.year}")
```

DIFFERENCE MODULE:

```python
from datetime import datetime , timedelta

class Difference:

  def difference_with_current(self, date):
    try:
      current_date = datetime.now().date()
      input_date = datetime.strptime(date, '%d.%m.%Y').date()
      return (current_date - input_date).days
    except ValueError:
```

```
    return None

def difference(self, date_str1,date_str2):
  try:
    date1 = datetime.strptime(date_str1 , '%d.%m.%Y').date()
    date2 = datetime.strptime(date_str2 , '%d.%m.%Y').date()
    return (date1 - date2).days
  except ValueError:
    return None

def days_after(self, days):
  try:
    current_date = datetime.now().date()
    future_date = current_date + timedelta(days = days)
    return future_date.strftime('%d.%m.%Y')
  except ValueError:
    return None

def days_before(self, days):
  try:
    current_date = datetime.now().date()
    past_date = current_date - timedelta(days = days)
    return past_date.strftime('%d.%m.%Y')
  except ValueError:
    return None

def month_after(self, months):
  try:
    current_date = datetime.now().date()
    future_date = current_date + timedelta(days = months * 30)
    return future_date.strftime('%d.%m.%Y')
  except ValueError:
    return None

def month_before(self, months):
  try:
    current_date = datetime.now().date()
    past_date = current_date - timedelta(days = months * 30)
    return past_date.strftime('%d.%m.%Y')
  except ValueError:
    return None
```

SADAKOPA RAMAKRISHNAN T
31222225002109

VALIDITY MODULE

Python

```
class Valid:
 def IsValidTime(self, time):
   hour, minute, second = time.split(':')
   if -1<int(second)<60 and -1<int(minute)<60 and -1<int(hour)<24:
     return True
   else:
     return False

 def IsValidDate(self, date):
   day, month, year = date.split('.')
   month = int(month)
   year = int(year)
   day = int(day)
   if 0<year<10000 and 0<month<13:
     if month == 2:
       if (year%400==0 or (year%4==0 and year%100!=0)) and day<30:
         return True
       elif day<29:
         return True
       else:
         return False
     elif (month==4 or month==6 or month==9 or month==11) and day<31:
       return True
     elif (month==1 or month==3 or month==5 or month==7 or month==8 or month==10 or
month==12) and day<32:
       return True
     else:
       return False
   else:
     return False
```

SADAKOPA RAMAKRISHNAN T
31222225002109

TIME PACKAGE:

FREQ MODULE:

```python
class Frequency:
 def frequency(self, file, word):
   cnt = 0
   with open(file, 'r') as f:
     words = f.read().split()
     for w in words:
       if w==word:
         cnt+=1
   return cnt
```

ISEXISTS MODULE:

```python
class IsExists:

 def is_existS(self, file, word):
   with open(file, 'r') as f:
     text = f.read()
     if word in text:
       return True
     return False
```

SADAKOPA RAMAKRISHNAN T
31222225002109

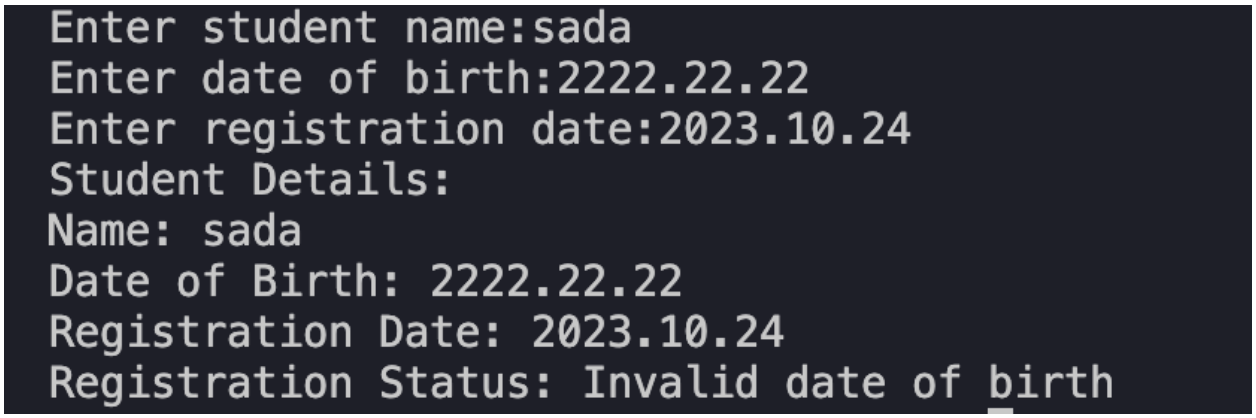
APPLICATION MODULE:

Python

```
# -*- coding: utf-8 -*-
"""
This module provides code for Modules and Packages. This is a part
of the exercises given under the course UIT2312 (Programming
and Design Patterns).

In this source code I've executed my own logic and may contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Thur Sept 21 2023

Revised on Wed Sept 26 2023

Original Author: T. Sadakopa Ramakrishnan <sadakopa2210221@ssn.edu.in>
"""


from Date.validity import Valid
from Date.difference import Difference

class StudentCompetition:

  def __init__(self,name,dob,registration_date):
    self.name = name
    self.dob = dob
    self.registration_date = registration_date

  def is_registration_valid(self):
    if not Valid().IsValidDate(self.dob):
      return "Invalid date of birth"
    age_in_years = Difference().difference_with_current(self.dob)
    if age_in_years is None or age_in_years>(17*365):
      return "Invalid age"
    registration_age =
Difference().difference_with_current(self.registration_date)
    if registration_age is None or registration_age>(0.5*365):
      return "Registration expired"
    else:
      return "Registration is valid"
```

```
if __name__ == "__main__":
  student = StudentCompetition(input("Enter student name:"), input("Enter date of
birth:"), input("Enter registration date:"))
  print(f"Student Details:\nName: {student.name}\nDate of Birth:
{student.dob}\nRegistration Date: {student.registration_date}")
    print("Registration Status:", student.is_registration_valid())
```

OUTPUT:



SADAKOPA RAMAKRISHNAN T
31222225002109

BINARYTREE PACKAGE:

NODE MODULE:

```python
class Node:
 def __init__(self, data):
   self.data = data
   self.left = None
   self.right = None
```

BINARYTREE MODULE:

```python
from abc import ABC, abstractmethod

class BinaryTree(ABC):
  def __init__(self):
    self.root = None

  @abstractmethod
  def create_tree(self, iterable):
    pass

  @abstractmethod
  def insert(self, data):
    pass

  @abstractmethod
  def traverse(self):
    pass
```

BST MODULE:

```python
from BinaryTree.binarytree import BinaryTree
from BinaryTree.node import Node
```

SADAKOPA RAMAKRISHNAN T
31222225002109

```
class BinarySearchTree(BinaryTree):
  def create_tree(self, iterable):
    for item in iterable:
      self.insert(item)

  def insert(self, data):
    self.root = self._insert_recursively(self.root, data)

  def _insert_recursively(self, node, data):
    if node is None:
      return Node(data)
    if data < node.data:
      node.left = self._insert_recursively(node.left, data)
    elif data > node.data:
      node.right = self._insert_recursively(node.right, data)
    return node

  def traverse(self):
    result = []
    self._inorder_traversal(self.root, result)
    return result

  def _inorder_traversal(self, node, result):
    if node:
      self._inorder_traversal(node.left, result)
      result.append(node.data)
      self._inorder_traversal(node.right, result)
```

SADAKOPA RAMAKRISHNAN T
31222225002109

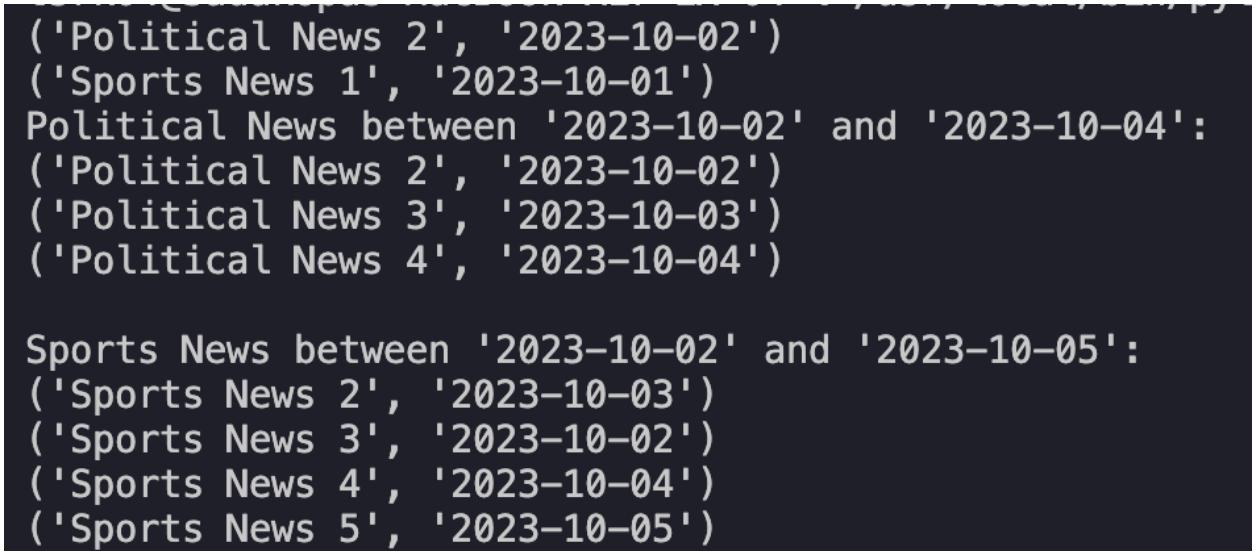
NEWSAPP APPLICATION

Python

```python
from BinaryTree.bst import BinarySearchTree

political_news_bst = BinarySearchTree()
sports_news_bst = BinarySearchTree()

def display_news_on_date(tree, date):
  news = tree.traverse()
  for news_item in news:
    if news_item[1] == date:
      print(news_item)

political_news = [("Political News 1", "2023-10-01"), ("Political News 2",
"2023-10-02"),("Political News 3", "2023-10-03"),
  ("Political News 4", "2023-10-04"),
  ("Political News 5", "2023-10-05")]
sports_news = [("Sports News 1", "2023-10-01"), ("Sports News 2",
"2023-10-03"),("Sports News 3", "2023-10-02"),
  ("Sports News 4", "2023-10-04"),
  ("Sports News 5", "2023-10-05")]

for news_item in political_news:
  political_news_bst.insert(news_item)

for news_item in sports_news:
  sports_news_bst.insert(news_item)

display_news_on_date(political_news_bst, "2023-10-02")
display_news_on_date(sports_news_bst, "2023-10-01")

def get_news_between_dates(tree, start_date, end_date):
  news = tree.traverse()
  filtered_news = []

  for news_item in news:
    if start_date <= news_item[1] <= end_date:
      filtered_news.append(news_item)

  return filtered_news

political_news_between_dates = get_news_between_dates(political_news_bst,
"2023-10-02", "2023-10-04")
```

```
print("Political News between '2023-10-02' and '2023-10-04':")
for news_item in political_news_between_dates:
  print(news_item)

sports_news_between_dates = get_news_between_dates(sports_news_bst, "2023-10-02",
"2023-10-05")
print("\nSports News between '2023-10-02' and '2023-10-05':")
for news_item in sports_news_between_dates:
  print(news_item)
```

OUTPUT:



SADAKOPA RAMAKRISHNAN T
31222225002109

CODE:

Q5.1:

Python

```
'''
Sadakopa Ramakrishnan T
ex 5 pdp lab qn 1
Define a vector inheriting from standard list. Let the vector is restricted to
having only integer
number in the list. Overload appropriate function and operator so that when any
other type of element
is inserted the program raises an error.
'''


class Vector(list):
  def __init__(self, sequence=None):
    super().__init__()
    if sequence is not None:
      try:
        for item in sequence:
          self.append(item)
      except TypeError:
        self.append(sequence)

  def append(self, item):
    if isinstance(item, int):
      super().append(item)
    else:
      raise TypeError('Only integers can be appended')

  def __add__(self, other):
    return sorted(Vector(set(self) | set(other)))

  def __sub__(self, other):
    union = set(self) | set(other)
    intersection = set(self) & set(other)
    return sorted(Vector(union - intersection))

  def get_ratios(self, other):
    if isinstance(other, Vector):
      if len(self) == len(other):
        result = Vector()
        for i in range(len(self)):
```

```
            try:
              result.append(self[i] // other[i])
            except ZeroDivisionError:
              result.append(0)
          return result
        else:
          raise IndexError("Vectors have different dimensions")
      else:
        raise TypeError(f"{other} is not a Vector object")


if __name__ == "__main__":
  v1 = Vector([1, 2, 3])
  v2 = Vector([3, 4, 5, 7])
  identity = Vector([1] * 4)

  v1.append(4)
  print("v1:", v1)
  print("v2:", v2)
  print("v1 + v2:", v1 + v2)
  print("v1 - v2:", v1 - v2)
  print("v1 / identity ratios:", v1.get_ratios(identity))

  try:
    v1.get_ratios(Vector([1, 0, 3, 2]))
  except ZeroDivisionError:
    print("Division by zero!")

  try:
    v1.get_ratios(Vector([1, 0, 3]))
  except IndexError as e:
    print(e)
```

```
v1: [1, 2, 3, 4]
v2: [3, 4, 5, 7]
v1 + v2: [1, 2, 3, 4, 5, 7]
v1 - v2: [1, 2, 5, 7]
v1 / identity ratios: [1, 2, 3, 4]
Vectors have different dimensions
```

SADAKOPA RAMAKRISHNAN T
31222225002109

Q5.2:

Python

```
'''
SADAKOPA RAMAKRISHNAN T
ex 5 pdp lab qn 2
Create a class ‘Employee’. The constructor must assign the first name and last
name of an employee.
Define a function called from_string() which gets a single string from the
user, splits and assigns to the
first and last name. For example, if the string is ‘Seetha Raman’, the function
should assign first name
as Seetha and second name as Raman and the function returns the object. Can you
design from_string()
as a class or instance function? Justify your response'''


class Employee:
  """Stores the details of an employee"""

  def __init__(self, fname=None, lname=None):
    """Constructor of the Employee class"""
    self.fname = fname
    self.lname = lname

  @classmethod
  def from_string(cls, name):
    """Create an Employee object from a full name string."""
    fname, lname = name.split()
    return cls(fname, lname)

  def __str__(self):
    """Return a string representation"""
    return "\nfirst name: {}\nlast name: {}\n".format(self.fname, self.lname)

if __name__ == "__main__":
  # Create Employee objects
  emp1 = Employee.from_string(input("\nEnter employee 1 full name: "))
  emp2 = Employee()
  emp2.from_string(input("\nEnter employee 2 full name: "))
  fname = input("\nEnter employee 3 first name: ")
  lname = input("Enter employee 3 last name: ")
  emp3 = Employee(fname, lname)

  # Display the employee details
  print("\nEmployee details:")
```

```
print(emp1)
print(emp2)
print(emp3)
```

Q5.3:

Python

```
'''
Sadakopa Ramakrishnan
ex 5 pdp lab qn 3
Implement the classes Movie() and MovieList() as described in the above figure.
Override the
appropriate functions so that the MovieList is generated based on the genre
assigned in the instance
variable when first object is created. For example, if the genre is defined as
thriller, the list accepts only
thriller movies.
When two lists are given as input, the list with more number of movies are
returned
'''

class Movie:
  def __init__(self, name, genre):
    self.name = name
    self.genre = genre

  def __str__(self):
    return "Movie name: {}\nGenre: {}".format(self.name, self.genre)

class MovieList(list):
  def __init__(self, genre):
    super().__init__()
```

```
    self.genre = genre

  def append(self, movie):
    if not isinstance(movie, Movie) or movie.genre != self.genre:
      raise TypeError("Only movies of the same genre can be added to the list")
    super().append(movie)

  def __str__(self):
    return "\nMovie list Genre: {}\nList: {}".format(self[0].genre,
super().__str__())

  def __gt__(self, other):
    if not isinstance(other, MovieList):
      raise TypeError("Only MovieList objects can be compared")
    return self if len(self) >= len(other) else other

if __name__ == "__main__":

  crime_mov_1 = Movie("The Dark Knight", "Crime Thriller")
  crime_mov_2 = Movie("Heat", "Crime Thriller")
  adven_mov_1 = Movie("Indiana Jones and the Last Crusade", "Adventure")
  adven_mov_2 = Movie("Jurassic Park", "Adventure")

  crime_mov_lst = MovieList("Crime Thriller")
  crime_mov_lst.append(crime_mov_1)
  crime_mov_lst.append(crime_mov_2)

  adven_mov_lst = MovieList("Adventure")
  adven_mov_lst.append(adven_mov_1)
  adven_mov_lst.append(adven_mov_2)

  print("\nMovie List:")
  print(crime_mov_lst)
  print(adven_mov_lst)

  print("Greater > :")
  print(crime_mov_lst > adven_mov_lst)
```