1. A data base consisting of a one-dimensional list is used to store the last four digits of student numbers for all students enrolled in the COMP-1001 class. The list is given here:

```
students =  [1906, 1993, 2419, 2089, 4865, 2186, 3950, 1816, 2321, 3092,
             3457, 4410, 2157, 3197, 4717, 1539, 3940, 3928, 4881, 3270]
```

   Write a program that consists of two functions, **bubbleSort()** and **binarySearch()**, where the bubbleSort() function performs a bubble sort on the list of student numbers to sort them in increasing order, and the binarySearch() function searches the list of student numbers and returns true (boolean) if the passed student number is in the list. Also include a main() function, where the user is asked to input a student number and the resulting answer is printed out.

   Sample output:

```
Enter the last four digits of a student number (xxxx): 3940
The student is enrolled

Enter the last four digits of a student number (xxxx): 1993
The student is enrolled

Enter the last four digits of a student number (xxxx): 1399
The student is not enrolled
```

2. The formula for the distance between two points (x_1, y_1, z_1) and (x_2, y_2, z_2) is given by the formula:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

   Write a program to find the distance between all pairs of points in a two-dimensional list, given here:

```
points = [[4, 2, 1], [-1, 3, 5], [6, 9, -2], [8, -1, 5]]
```

   Output the co-ordinates of the two points that are closest to each other along with the distance between the points. Using insertion sort, output the distances between points in descending order. Be sure to include a function **distance(a, b)**, where **a** and **b** are points represented by lists of (x, y, z) co-ordinates (i.e., a = [4, 2, 1]), and a function to implement the insertion sort. Do not use any built-in Python functions for sorting.

   Sample output:

```
Sorted Distances:
12.37
11.58
 9.85
 7.87
 6.48
 6.40
Closest points are:
[4, 2, 1] and [8, -1, 5] with distance of 6.40
```

3. Write a program to create an Employee class that contains the following:

   - The private data fields name and salary, for the name of an employee and their salary
   - A constructor with arguments for name and salary, where name defaults to "No name" and salary defaults to 0
   - Accessor methods for both name and salary, as well as mutator methods to set name and salary
   - A method called **__str__()**, which returns an employee's information as a formatted string

   Add to the program a sub-class of Employee called Manager that contains the following:

   - The private data field department, which is the name of the manager's department
   - A constructor with argument for name, salary and department, where department defaults to "No department"
   - An accessor and mutator method for department, which get and set the value of department (respectively)
   - A method called **__str__()**, which returns a manager's information as a formatted string.

   Include a **main()** function to create a manager object and test all available methods on the manager object.

   Sample output:

   ```
   Name: Richard
   Salary: $253626985
   -----------------------
   Name: Caitlin
   Salary: $256000
   Department: House Wares
   -----------------------
   Set Richard's salary to 1000000:
   ```

```
Name: Richard
Salary: $1000000
-----------------------
Set Caitlin’s name to Cat:
Name: Cat
Salary: $256000
Department: House Wares
-----------------------
Make default Manager:
Name: No name
Salary: $0
Department: No department
```

4. A complex number is a number of the form $a + bi$, where a and b are real numbers and i is $\sqrt{-1}$. The a part of the complex number is its *real* part, while the bi part is its *imaginary* part.

   The addition, subtraction, multiplication, division and absolute value of complex numbers can be defined as follows:

$$(a + bi) + (c + di) \implies (a + c) + (b + d)i$$

$$(a + bi) - (c + di) \implies (a - c) + (b - d)i$$

$$(a + bi) \times (c + di) \implies (ac - bd) + (bc + ad)i$$

$$(a + bi) \div (c + di) \implies \frac{(ac + bd)}{(c^2 + d^2)} + \frac{(bc - ad)}{(c^2 + d^2)}$$

$$|a + bi| \implies \sqrt{a^2 + b^2}$$

   A complex number can be interpreted as a point on a plane by identifying the (a, b) values as the co-ordinates of the point. The absolute value of the complex number corresponds to the distance of the point to the origin. Although Python has a **complex** class for performing complex arithmetic, write a program to implement your own complex class. Create a class named **Complex** for representing complex numbers, which includes methods for performing the complex arithmetic presented above. Also include a **__str__** method that returns a string representation of the complex number as $(a + bi)$, where only a is returned if $b = 0$. The class’s constructor, **Complex(a, b)** should create a complex number, $a + bi$, where the default values of a and b is 0. Also, create methods **getRealPart()** and **getImaginaryPart()** for returning the real and imaginary parts of the complex number, respectively. Write a **main()** function to test the Complex class by prompting the user to enter the real and imaginary parts of two complex numbers and displaying the results of calling each method.

   Sample output:

```
Enter the real part of the first complex (a): 2
Enter the imaginary part of the first complex (b): 3
You have entered 2+3i
Enter the real part of the second complex (c): 3
Enter the imaginary part of the second complex (d): 5
You have entered 3+5i
----------------------------------------
The addition of 2+3i and 3+5i is: 5+8i
The subtraction of 2+3i and 3+5i is: -1+(-2)i
The multiplication of 2+3i and 3+5i is: -9+19i
The division of 2+3i and 3+5i is: 0
The absolute value of 2+3i is: 3.61
The absolute value of 3+5i is: 5.83
```

5. Write a program that implements a recursive function to compute the sum of the following series:

$$\sum_{n=1}^{\infty} m(n) = 2 + \frac{4}{5} + \frac{8}{25} + ... + \frac{2^n}{5^{n-1}}$$

Include a **main()** function to test the recursive function using a user-input value for **n**.

Sample output:

```
Enter the number of terms: 3
The sum of the first 3 terms is: 3.1200

Enter the number of terms: 100
The sum of the first 100 terms is: 3.3333
```

6. A local weather man has been collecting weather data and storing it in a file for quite some time. He has decided that now is a good time to calculate some statistics based on the data he has collected. The data file, **weather_stats.dat**, is formatted in the following way:

```
Year        Day    Avg Condition    Max Temperature
   1    Tuesday             Rain                 12
```

In the file, ”Year” corresponds to the year the data was collected, ”Day” corresponds to the day of the week the data was collected on, ”Avg Condition” corresponds to the average condition of that day during the year (clear, rain or snow), and ”Max Temperature” corresponds to the maximum temperature of that day during the year.

The weather man would like to know how many clear Mondays there were out of all years, as well as the average of the maximum temperatures of all Mondays, and the average of the maximum temperatures out of all years, which he would like to append to the bottom of the data file.

Write a program to read the data file, collect the necessary data, compute the requested averages and append them to the data file. Be sure to break each task into separate functions (i.e., the function **avgMondayTemp()** will calculate the average temperature for each Monday and return it), and include a **main()** function to use each function and append the data to the data file. Also, when working with the file, be sure to implement an exception in the case that the file is not found.

Sample output:

```
The number of clear Mondays is:  2
The average Monday temperature is: 21.79
The overall average temperature is: 15.74
File written!
```