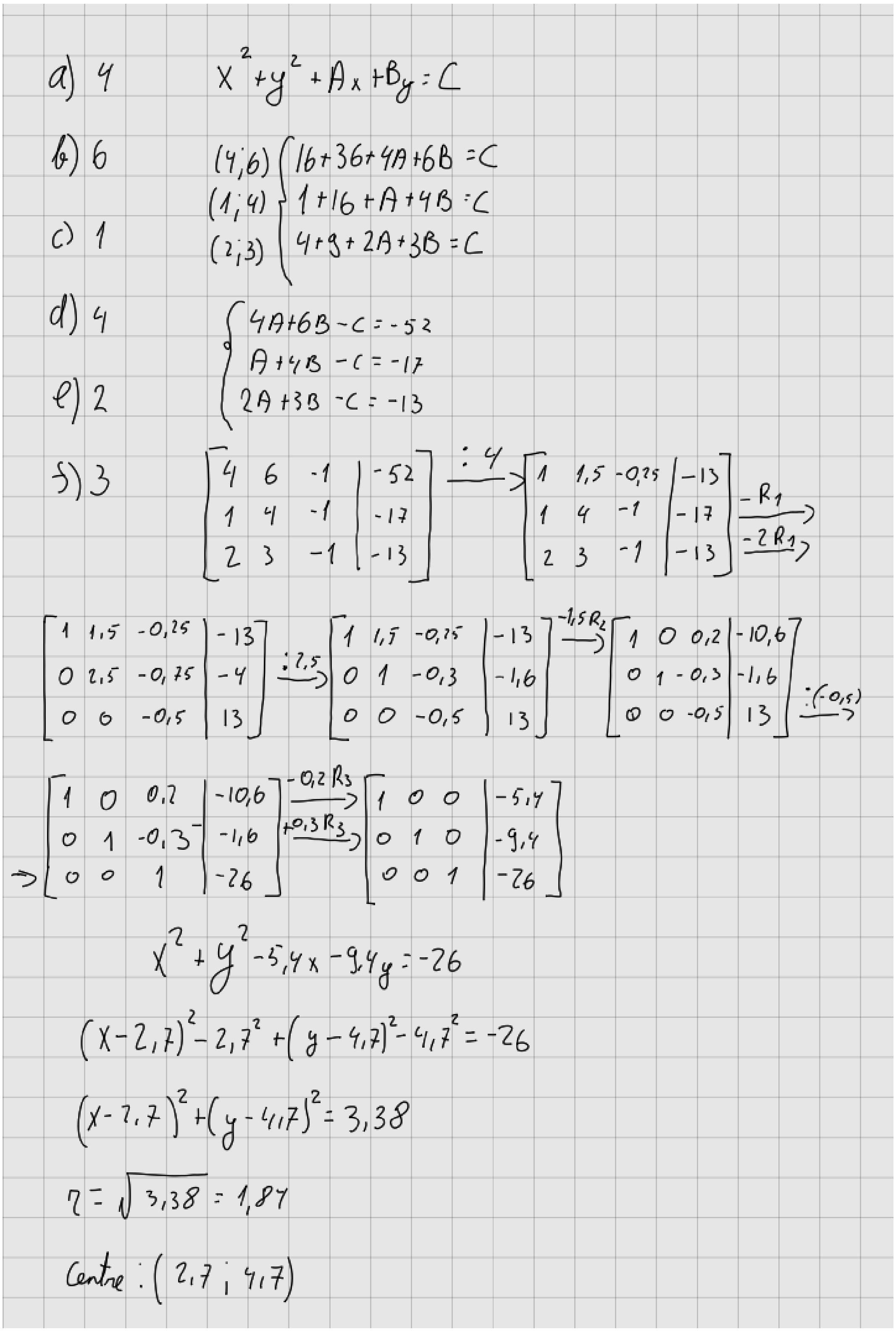
1. Calculation by hand



a) 4 $\quad x^2+y^2+Ax+By=C$

b) 6 $\quad \begin{cases} (4;6) & 16+36+4A+6B=C \\ (1;4) & 1+16+A+4B=C \\ (2;3) & 4+9+2A+3B=C \end{cases}$

c) 1

d) 4 $\quad \begin{cases} 4A+6B-C=-52 \\ A+4B-C=-17 \\ 2A+3B-C=-13 \end{cases}$

e) 2

f) 3

$$\left[\begin{array}{ccc|c} 4 & 6 & -1 & -52 \\ 1 & 4 & -1 & -17 \\ 2 & 3 & -1 & -13 \end{array}\right] \xrightarrow{:4} \left[\begin{array}{ccc|c} 1 & 1{,}5 & -0{,}25 & -13 \\ 1 & 4 & -1 & -17 \\ 2 & 3 & -1 & -13 \end{array}\right] \begin{array}{l} \xrightarrow{-R_1} \\ \xrightarrow{-2R_1} \end{array}$$

$$\left[\begin{array}{ccc|c} 1 & 1{,}5 & -0{,}25 & -13 \\ 0 & 2{,}5 & -0{,}75 & -4 \\ 0 & 0 & -0{,}5 & 13 \end{array}\right] \xrightarrow{:2{,}5} \left[\begin{array}{ccc|c} 1 & 1{,}5 & -0{,}25 & -13 \\ 0 & 1 & -0{,}3 & -1{,}6 \\ 0 & 0 & -0{,}5 & 13 \end{array}\right] \xrightarrow{-1{,}5R_2} \left[\begin{array}{ccc|c} 1 & 0 & 0{,}2 & -10{,}6 \\ 0 & 1 & -0{,}3 & -1{,}6 \\ 0 & 0 & -0{,}5 & 13 \end{array}\right] \xrightarrow{:(-0{,}5)}$$

$$\rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0{,}2 & -10{,}6 \\ 0 & 1 & -0{,}3 & -1{,}6 \\ 0 & 0 & 1 & -26 \end{array}\right] \begin{array}{l} \xrightarrow{-0{,}2R_3} \\ \xrightarrow{+0{,}3R_3} \end{array} \left[\begin{array}{ccc|c} 1 & 0 & 0 & -5{,}4 \\ 0 & 1 & 0 & -9{,}4 \\ 0 & 0 & 1 & -26 \end{array}\right]$$

$$x^2+y^2-5{,}4x-9{,}4y=-26$$

$$(x-2{,}7)^2-2{,}7^2+(y-4{,}7)^2-4{,}7^2=-26$$

$$(x-2{,}7)^2+(y-4{,}7)^2=3{,}38$$

$$r=\sqrt{3{,}38}=1{,}87$$

Centre: $(2{,}7\,;\,4{,}7)$

2. Python

```python
import csv  # importing CSV library to use csv functions
from typing import List

import numpy as np  # importing CSV library to use matrix-calculations

# The class Point initializes an object with x and y coordinates as float values.
class Point:
    def __init__(self, x, y):
        self.x = float(x)
        self.y = float(y)

    # [\]

    def print(self):
        """
        prints the values of the attributes "x" and "y".
        """
        print("{0} {1}".format(self.x, self.y))

    # [\]
    def circeq(self):
        return [self.x, self.y, -1]

    def circsq(self):
        return -self.x**2 - self.y**2

    x = 0
    y = 0

def dictionary(self):
    """
    The function returns the dictionary representation of an object's attributes.
    :return: The method `dictionary` is returning the dictionary representation of the object's
    attributes using the `__dict__` attribute.
    """
    return self.__dict__

# [\]
# `inputarray = []` initializes an empty list called `inputarray`. This list will be used to store
# `Point` objects created from the data in the input file.
inputarray = []
# [\]

# This code is opening a CSV file named "inputfile AS1.csv" in read mode using the `open()` function
# and assigning it to the variable `input`. Then, it is using the `csv.DictReader()` function to read
# the contents of the CSV file and convert them into a dictionary format. The resulting dictionary is
# assigned to the variable `locations`. The `with` statement is used to ensure that the file is
# properly closed after it has been read.
with open(r"inputfile AS1.csv", "r") as input:
    locations = csv.DictReader(input)
    # [\]

    # Essentially, this code is converting the data from the input CSV file into a list of `Point` objects
    # that can be used for further calculations.
    for pointentry in locations:
        point = Point(pointentry["x"], pointentry["y"])
        inputarray.append(point)
# [\]

# `calcarray = np.array(inputarray)` is converting the list of `Point` objects stored in `inputarray`
# into a numpy array called `calcarray`. This allows for easier manipulation and calculation of the
# data using numpy functions.
calcarray = np.array(inputarray)
# [\]

# `numOfRows = calcarray.shape[0]` is calculating the number of rows in the numpy array `calcarray`
# and assigning it to the variable `numOfRows`. This value is used later in the code to determine if
# there are exactly three points in the input file, in which case a circle will be calculated.
numOfRows = calcarray.shape[0]
# [\]

sol = []
eq = []
# This code block checks if the number of rows in the input CSV file is equal to 3. If it is, then it
# assumes that the input file contains three points and calculates the equation of the circle that
# passes through those three points.
if numOfRows == 3:
    print("Given three points a circle will be calculated")
    for n in range(3):
```

```python
        eq.append(inputarray[n].circeq())
        sol.append(inputarray[n].circsq())
    eqarray = np.array(eq)
    solarray = np.array(sol)
    solution = np.linalg.solve(eqarray, solarray)
# [\]

# additional +/- formatting  (You do not need to implement this but it is recommended to do)
if solution[0] < 0:
    pr_sol0 = str(solution[0])
else:
    pr_sol0 = "+" + str(solution[0])
if solution[1] < 0:
    pr_sol1 = str(solution[1])
else:
    pr_sol1 = "+" + str(solution[1])
if solution[2] < 0:
    pr_sol2 = str(solution[2])
else:
    pr_sol2 = abs(solution[2])

Equation = ["Equation:", "x^2+y^2{0}x{1}y = {2}".format(pr_sol0, pr_sol1, pr_sol2)]
print(Equation[0], Equation[1])

#This code is writing the equation of the circle calculated
# earlier to a new CSV file named "outputfile AS1.csv".
f = open("outputfile AS1.csv", "w")

writer = csv.writer(f)
writer.writerow(Equation)

f.close()
# [\]

```