UECM1703 INTRODUCTION TO SCIENTIFIC COMPUTING Dec 2022 Marking Guide

PART A (Answer **ONE** question only)

Q1. (a) Write a program script without importing any modules to read in two integers and print out (i) the sum of the two integers and (ii) the product of the two integers modulo 11 (i.e. the remainder of the division of the product of the two integers by 11). (6 marks)

Ans. A sample solution is listed below.

```
a = input("Enter an integer a: ")              # [1 mark]
a = int(a)                                     # [1 mark]
b = input("Enter an integer b: ")
b = int(b)                                     # [1 mark]
print("The sum of a and b          =", a+b)    # [1 mark]
prod = (a * b) % 11                            # [1 mark]
print("The product of a and b modulo 11 =", prod)
                                               # [1 mark]
```

(b) Write a Python script without importing any modules but only using appropriate data representation and for loops to generate a distance matrix for the 3-D points P_i, $i = 1,2,3,4,5$ in Table 1.1.

Point	x	y	z
P_1	2	3	5
P_2	2	5	3
P_3	6	4	2
P_4	5	3	4
P_5	3	5	4

Table 1.1: 3-D points.

Define a function to calculate the Euclidean distance between two 3-D points in your Python script and then go through all the pairs of the points P_1 to P_4 to generate a distance table in Table 1.2.

```
0.000000    2.828427    5.099020    3.162278    2.449490
2.828427    0.000000    4.242641    3.741657    1.414214
5.099020    4.242641    0.000000    2.449490    3.741657
3.162278    3.741657    2.449490    0.000000    2.828427
2.449490    1.414214    3.741657    2.828427    0.000000
```

Table 1.2: Distance matrix of the 3-D points.

(8 marks)

Ans. A sample solution is listed below.

```
points = [(2,3,5), (2,5,3), (6,4,2), (5,3,4), (3,5,4)]
print(points)                                        # [2 marks]

def euclid_dist(pt1, pt2):                           # [1 mark]
    from math import sqrt
    return sqrt((pt2[0]-pt1[0])**2 + (pt2[1]-pt1[1])**2
                + (pt2[2]-pt1[2])**2)                # [2 marks]

for pt1 in points:                                   # [1 mark]
    for pt2 in points:                               # [1 mark]
        dist = euclid_dist(pt1, pt2)
        print(f"{dist:10.6f} ", end="")   # same row
    print()  # Print a new line                      # [1 mark]
```

(c) Given the function below

$$A(k) = 1 - \frac{1}{2} + \frac{1}{3} - \cdots + \frac{(-1)^{k+1}}{k}$$

Write down the values of $A(1)$ and $A(5)$ and then write a Python program script to define the function $A(k)$ and to generate Table 1.3.

```
  k        A(k)       |A(k)-ln(2)|
----- ------------ ------------
   10   0.64563492   0.04751226
   25   0.71274750   0.01960032
   50   0.68324716   0.00990002
  100   0.68817218   0.00497500
  250   0.69115118   0.00199600
  500   0.69214818   0.00099900
 1000   0.69264743   0.00049975
 2000   0.69289724   0.00024994
```

Table 1.3: Analysis of the Convergence of $A(k)$.

(6 marks)

Ans. First, we calculate

$$A(1) = 1$$
$$A(5) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} = \frac{47}{60} = 0.78\dot{3}$$

[2 marks]

A sample solution is listed below.

```
from math import log
def A(k):
    return sum((-1)**(k+1)/k for k in range(1,k+1))
                                                        # [2~marks]

print("k".center(5), "A(k)".center(12), "|A(k)-ln(2)|".center(12))
print("----- ------------ ------------")
for k in [10, 25, 50, 100, 250, 500, 1000, 2000]:
    y = A(k)
    Err = abs(y-log(2))
    print(f"{k:5d} {y:12.8f} {Err:12.8f}")     # [2~marks]
```

[Total : 20 marks]

Q2. (a) Consider the following python script:

```
f = open('sherlock_holmes.txt', 'r')
text = f.read()
f.close()

text = text.lower()
alphabets = []
for c in text:
    if c.isalpha():
        alphabets.append(c)
    else:
        alphabets.append(' ')
text = ''.join(alphabets)
words = text.split()
print(words)
```

Rewrite the code so the resulting code will still have the same logic but uses the keywords `def`, `with`, and the appropriate modular structure and list comprehension. (8 marks)

Ans.

```
def get_words(fname):
    with open(fname, 'r') as f:
        text = f.read()
    text = text.lower()
    alphabets = [c if c.isalpha() else ' ' for c in text]
    text = ''.join(alphabets)
    words = text.split()
    return words

print(get_words('sherlock_holmes.txt'))
```

- correct use of `with` . . . . [2 marks]
- correct use of `def` and return values . . . . [4 marks]
- correct structure of list comprehension . . . . [2 marks]

(b) Write down the output of the following fragments of code:

(i)

```
alpha1 = 'abcdefghijklmnopqrstuvwxyz'
alpha2 = 'efghijklmnopqrstuvwxyzabcd'
d = dict(zip(alpha1,alpha2))
for c in 'secret?':
    print(d.get(c,c))
```

(3 marks)

Ans.

```
w
i
g
v
i
x
?
```

. . . . [3 marks]

(ii)

```
fruits = ['durian', 'mango', 'pineapple', 'grape']
prices = [56.30, 8.30, 5.20, 14.50]
price_list = list(zip(prices, fruits))
print(sorted(price_list))
```

(3 marks)

Ans.

```
[(5.2, 'pineapple'), (8.3, 'mango'),
 (14.5, 'grape'), (56.3, 'durian')]
```

. . . . [3 marks]

(c) The following code is supposed to calculate the moving average of some data with window size 3. However, there are 3 mistakes in the code. State the mistakes and suggest how to rectify them.

```
1 data = ['1.2','1.4','1.2','1.3','1.5','1.4','1.3','1.6']
2 for i in range(len(data)):
3     x1,x2,x3 = data[i], data[i+1], data[i+2]
4     average = (x1 + x2 + x3)/3
5     moving_average.append(average)
6
7 print(moving_average)
```

(6 marks)

Ans.

- line 3. Running index data[i+1], data[i+2] out of range when $i \geq 6$. Line 2 should be `for i in range(len(data)-2):` . . . . [2 marks]
- line 4. x1, x2, x3 are strings, need to convert to float before summation. Add a line `x1,x2,x3 = float(x1), float(x2), float(x3)` before line 4. . . . . [2 marks]
- Line 5. Variable `moving_average` is not defined. Add `moving_average = []` before the for loop. . . . . [2 marks]

```
data = ['1.2','1.4','1.2','1.3','1.5','1.4','1.3','1.6']
moving_average = []

for i in range(len(data)-2):
    x1,x2,x3 = data[i], data[i+1], data[i+2]
    x1,x2,x3 = float(x1), float(x2), float(x3)
    average = (x1 + x2 + x3)/3
    moving_average.append(average)

print(moving_average)
```

[Total : 20 marks]

PART B (Answer **ALL** questions)

Q3. (a) Given the following matrix

$$A = \begin{bmatrix} 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 25 & 27 & 29 & 31 & 33 & 35 & 37 \\ 39 & 41 & 43 & 45 & 47 & 49 & 51 \\ 53 & 55 & 57 & 59 & 61 & 63 & 65 \\ 67 & 69 & 71 & 73 & 75 & 77 & 79 \\ 81 & 83 & 85 & 87 & 89 & 91 & 93 \end{bmatrix}$$

(i) Write down the output of `print(A[2:5, 3:6])` (3 marks)

Ans. Computer output is as follows:

```
[[45 47 49]
 [59 61 63]
 [73 75 77]]
```

Full marks will be awarded if the concept is correct. .... [3 marks]

(ii) Write down the output of `print(A[[2,3,4],[3,4,5]])` (2 marks)

Ans. `[45 61 77]` ....[2 marks]

(iii) Write down the output of `print(A[1:3,-2:] - A[1:3,:2])` (2 marks)

Ans.

$$\begin{bmatrix} 35 & 37 \\ 49 & 51 \end{bmatrix} - \begin{bmatrix} 25 & 27 \\ 39 & 41 \end{bmatrix} = \begin{bmatrix} 10 & 10 \\ 10 & 10 \end{bmatrix}$$

.... [2 marks]

(iv) Write down the output of `print(A[:,3] < 50)` (2 marks)

Ans. `[ True True True False False False]` ....[2 marks]

(v) Write down the simplest Python command (with the appropriate indexing) to pick all the even rows and odd columns of `A` which results in:

$$\begin{bmatrix} 25 & 29 & 33 & 37 \\ 53 & 57 & 61 & 65 \\ 81 & 85 & 89 & 93 \end{bmatrix}.$$

(2 marks)

Ans. `A[1::2,::2]` .... [2 marks]

(vi) Write the command to construct the following matrix

$$\begin{bmatrix} 11 & 13 & 15 & 1 & 0 & 0 & 0 & 0 & 0 \\ 25 & 27 & 29 & 0 & 1 & 0 & 0 & 0 & 0 \\ 39 & 41 & 43 & 0 & 0 & 1 & 0 & 0 & 0 \\ 53 & 55 & 57 & 0 & 0 & 0 & 1 & 0 & 0 \\ 67 & 69 & 71 & 0 & 0 & 0 & 0 & 1 & 0 \\ 81 & 83 & 85 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

using a portion the matrix A without using any loops. (2 marks)

Ans. `np.hstack((A[:,:3], np.eye(A.shape[0])))` ....[2 marks]

(b) Given the matrix below:

$$D = \begin{bmatrix} 3000 & 1 \\ 4000 & 2 \\ 4500 & 1 \\ 3500 & 2 \\ 4500 & 3 \end{bmatrix}.$$

Suppose that the first column of D represents the salaries of the employees of a company and the second column of D represents the performance grades of the employees. Suppose grade 1 gets an increment of 150, grade 2 gets an increment of 250 and grade 3 gets and increment of 320. By using appropriate the array indexing and array operations, **write down the Python commands to generate a matrix** E with the first two columns the same as the matrix D and the third column is the salaries after the increment. Note that printing the first row of the matrix E gives

$$\begin{bmatrix} 3000 & 1 & 3150 \end{bmatrix}$$

In addition, write down the whole matrix E. (4 marks)

Ans. There are many ways to solve this problem. Just by using indexing and matrix stacking, one possible way is shown below.

```
import numpy as np
D = np.array([[3000,1],[4000,2],[4500,1],[3500,2],[4500,3]])
I = np.array([150,250,320])                              # [1 mark]
E = np.vstack((D.T, D[:,0] + I[D[:,1]-1])).T             # [2 marks]
print(E)
```

The whole matrix E is $\begin{bmatrix} 3000 & 1 & 3150 \\ 4000 & 2 & 4250 \\ 4500 & 1 & 4650 \\ 3500 & 2 & 3750 \\ 4500 & 3 & 4820 \end{bmatrix}$ .... [1 mark]

(c) Investigate the Python program below:

```
1 import numpy as np
2 F = np.array([[3,5],[-2,3],[4,9]])
3 print(F.T @ F)
4 col1 = F[:,0]
5 col1[0] = 30
6 col2 = F[:,1]
7 col2[1] = 40
8 print(F.T @ F)
```

By using the vector and matrix operations from linear algebra, explain the output of line 3 and explain why line 8 prints the matrix $\begin{bmatrix} 920 & 106 \\ 106 & 1706 \end{bmatrix}$. (3 marks)

Ans. The computation related to line 3 is shown below:

$$\begin{bmatrix} 3 & -2 & 4 \\ 5 & 3 & 9 \end{bmatrix} \begin{bmatrix} 3 & 5 \\ -2 & 3 \\ 4 & 9 \end{bmatrix} = \begin{bmatrix} 29 & 45 \\ 45 & 115 \end{bmatrix}$$ [1.5 marks]

However, `col1` and `col2` are just views and changing them will cause the original matrix F to be change to

$$\begin{bmatrix} 30 & -2 & 4 \\ 5 & 40 & 9 \end{bmatrix} \begin{bmatrix} 30 & 5 \\ -2 & 40 \\ 4 & 9 \end{bmatrix} = \begin{bmatrix} 920 & 106 \\ 106 & 1706 \end{bmatrix}$$ [1.5 marks]

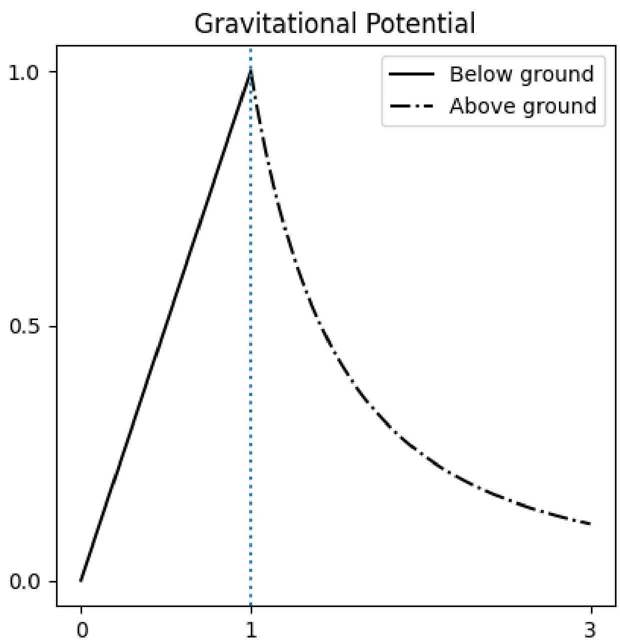
Q4. (a) Sketch the output of the following python script `question_4a.py`:

```
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(0, 1, 101)
y1 = x1
x2 = np.linspace(1, 3, 201)
y2 = 1/(x2**2)

fig, ax = plt.subplots(1,1)
ax.plot(x1, y1, 'k-')
ax.plot(x2, y2, 'k-.')
ax.set_xticks([0.0, 1.0, 3.0])
ax.set_yticks([0.0, 0.5, 1.0])
ax.axvline(1, linestyle=':')
ax.set_title('Gravitational Potential')
ax.legend(['Below ground', 'Above ground'],
            loc='upper right')
plt.show()
```

(12 marks)

Ans.

- Correct straight line and hyperbola. .... [2 marks]
- Correct line styles. .... [2 marks]
- Correct axis line and style. .... [2 marks]
- Correct legend labels. .... [2 marks]
- Correct title. .... [2 marks]
- Correct axes ticks. .... [2 marks]



(b) Consider the surface

$$z(x,y) = \frac{1}{2}\left(x^2 - y^2\right)$$

defined in the region $-3 \leq x, y \leq 3$. Write a Python program to

- Create 2 subplots of equal size.
- Plot the surface with the viridis colour map on the left subplot. Display the colour bar for the subplot.
- Plot the filled contour with eight level curves on the right subplot. Use the same viridis colour map for the filled contour.

(8 marks)

Ans.

- Correct preparation of mesh and z matrix. . . . . [2 marks]
- Correct creation of 2d and 3d axes. . . . . [2 marks]
- Correct surface plot. . . . . [2 marks]
- Correct filled contour plot. . . . . [2 marks]

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


x, y = np.linspace(-3,3), np.linspace(-3,3)
X, Y = np.meshgrid(x,y)
Z = 0.5*(X*X - Y*Y)

fig = plt.figure()
ax1 = fig.add_subplot(1,2,1, projection="3d")
ax2 = fig.add_subplot(1,2,2)

surf = ax1.plot_surface(X, Y, Z, cmap='viridis')
fig.colorbar(surf, ax=ax1)

ctr = ax2.contourf(X, Y, Z, levels=8, cmap='viridis')
plt.show()
```

[Total : 20 marks]

Q5. (a) Consider the following linear system:

$$\begin{aligned}38w+28x+23y+34z&=54\\33w+17x+49y+34z&=19\\32w+35x+17y+32z&=88\\32w+33x+52y+65z&=71\end{aligned}$$

(i) Write the above system in the form $\mathbf{AX}=\mathbf{b}$. (3 marks)

Ans.

$$\begin{bmatrix}38&28&23&34\\33&17&49&34\\32&35&17&32\\32&33&52&65\end{bmatrix}\begin{bmatrix}w\\x\\y\\z\end{bmatrix}=\begin{bmatrix}54\\19\\88\\71\end{bmatrix}$$ [3m]

(ii) Provide the Python codes to solve the linear system using matrix inversion. (4 marks)

Ans.

```
import numpy as np .................................[0.5m]
A = np.array([[38,28,23,34],[33,17,49,34],[32,35,17,32],
              [32,33,52,65]])........................[1m]
b = np.array([[54],[19],[88],[71]])..................[1m]
x = np.linalg.inv(A).dot(b)..........................[1m]
print(x) ...........................................[0.5m]
```

(iii) Suppose the unique solution to part (i) is $\mathbf{X}=\begin{bmatrix}w\\x\\y\\z\end{bmatrix}=\begin{bmatrix}-1.1335\\3.8378\\0.05955\\-0.3457\end{bmatrix}$, write down the Python codes to find $24w+30x+13y+24z$ and calculate the corresponding value. (4 marks)

Ans.

```
c = np.array([24,30,13,24])..................[1m]
bnew = c.dot(x)..............................[1m]
print("bnew = ",bnew)........................[1m]
```

$24w+30x+13y+24z=24(-1.1335)+30(3.8378)+13(0.05955)+24(-0.3457)=$ 80.4067 [1m]

(b) Consider the data shown below:

y	x	y	x
20	16	19	13
18	10	18	11
18	11	20	16
18	9	17	6
19	14	18	11
17	6	18	11
20	17	19	15
18	10	18	9

You fit the above data to $y=\beta_0+\beta_1x+\varepsilon$. You are given that:

- n is the number of observations.
- p is the number of parameters in the model.

- $\widehat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}$.
- $SSE = \mathbf{y}^{\mathbf{T}}\mathbf{y} - \widehat{\boldsymbol{\beta}}^T \mathbf{X}^{\mathbf{T}}\mathbf{y}$.

Write down the Python codes to compute SSE. (9 marks)

Ans.

```
import numpy as np .................................[0.5m]
y = np.array([20,18,18,18,19,17,20,18,19,18,20,17,18,18,19,18])
....................................................[1m]
x = np.array([[1,16],[1,10],[1,11],[1,9],[1,14],[1,6],[1,17],
             [1,10],[1,13],[1,11],[1,16],[1,6],[1,11],[1,11],
             [1,15],[1,9]])...........................[1m]
xtx = x.transpose().dot(x)..........................[0.5m]
xty = x.transpose().dot(y) .........................[0.5m]
b = np.linalg.inv(xtx).dot(xty).....................[0.5m]
n = len(y)..........................................[0.5m]
yty = y.transpose().dot(y)..........................[0.5m]
bxty = b.transpose().dot(xty).......................[1m]
J = np.ones((n,n))..................................[0.5m]
ytJy = y.transpose().dot(J).dot(y)..................[1m]
SSE = yty-bxty .....................................[1m]
print("SSE=", SSE)..................................[0.5m]
```

[Total : 20 marks]

Q6. The battery voltage drop in a guided missile motor observed over the time of missile flight is shown in Table 6.1.

Time, x_i (seconds)	1.0	2.0	3.0	4.0	5.0	6.0
Voltage Drop, y_i	1.91	1.89	1.94	2.03	2.16	2.29
Time, x_i (seconds)	7.0	8.0	9.0	10.0	11.0	12.0
Voltage Drop, y_i	2.42	2.53	2.60	2.61	2.55	2.40

Table 6.1: Battery voltage over time.

You fit the above data to $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$. The following output is obtained using Python's statsmodels and sklearn.preprocessing modules.

```
Results                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 5.750e+04
Date:                Wed, 26 Oct 2022   Prob (F-statistic):           1.14e-17
Time:                        14:19:06   Log-Likelihood:                 58.645
No. Observations:                  12   AIC:                            -109.3
Df Residuals:                       8   BIC:                            -107.3
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.0063      0.004    548.522      0.000       1.998       2.015
```

```
x1            -0.1436      0.002    -61.418      0.000      -0.149      -0.138
x2             0.0490      0.000    119.606      0.000       0.048       0.050
x3            -0.0029   2.08e-05   -137.592      0.000      -0.003      -0.003
==============================================================================
Omnibus:                        2.752   Durbin-Watson:                   3.083
Prob(Omnibus):                  0.253   Jarque-Bera (JB):                1.022
Skew:                           0.704   Prob(JB):                        0.600
Kurtosis:                       3.251   Cond. No.                     5.00e+03
==============================================================================
```

(a) Write down the Python codes to obtain the above results. (10 marks)

Ans.

```
import numpy as np ...............................................[1m]
import statsmodels.api as sm .....................................[1m]
from sklearn.preprocessing import PolynomialFeatures .............[1m]
y=[1.91, 1.89, 1.94,2.03,2.16,2.29,2.42,2.53,2.60,2.61,2.55,2.40].[1m]
n = len(y) .......................................................[1m]
x = np.array([1,2,3,4,5,6,7,8,9,10,11,12]).reshape(-1,1) .........[1m]
x3rd = PolynomialFeatures(degree=3, include_bias=True).fit_transform(x)
..................................................................[1m]
Mod = sm.OLS(y,x3rd) ........................................    [1m]
Results = Mod.fit() ........................................      [1m]
print("Results",Results.summary()) ...............................[1m]
```

(b) Write down the Python codes to determine the predicted value of the mean of the battery voltage drop when the time of missile flight is 2.0 seconds. (4 marks)

Ans.

```
Beta = Results.params ..............[1m]
xh = np.array([1,2,4,8]) ...........[1m]
yh = np.inner(xh, Beta) ............[1m]
print(yh)...........................[1m]
```

(c) Calculate the predicted value from part(b) using the above results. (3 marks)

Ans.

$y_h = 2.0063 - 0.1436(2) + 0.049(4) - 0.0029(8) = \boxed{1.8919}$ .... [3m]

(d) Is the model $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ significant for estimating voltage drop? Justify your answer using p-value. (3 marks)

Ans.

Since the p-value = $1.14e-17 < 1\%$. The null hypothesis of the model not significance for estimating voltage drop is rejected. Hence the model $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$ is significant for estimating voltage drop . ...[3m]

[Total : 20 marks]