IVE Information Technology

Information & Communications Technology

Programme Boards

Instructions:

(a) This paper has a total of TEN pages including the covering page and appendices.

(b) This paper contains TWO Sections.

(c) Section A is worth 40 marks and Section B is worth 60 marks.

(d) Answer ALL questions in Section A. Each question is worth 8 marks.

(e) Answer ALL questions in Section B. Each question is worth 15 marks.

Note: The result of this assessment will not be counted if you do not meet the minimum attendance requirement (if any) governed by the general academic regulations of your programme/course unless approval of the campus principal has been granted.

HIGHER DIPLOMA IN

SOFTWARE ENGINEERING
(IT114105)

MODULE TITLE:

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

MODULE CODE: **ITP4514**

SEMESTER 1
MAIN EXAMINATION

8 January, 2022
9:30 AM TO 11:30 AM (2 hours)

[This Page Intentionally Left Blank]

This paper contains TWO sections.

All working must be clearly shown on your answer book.

Section A (40 marks)

This section contains FIVE questions.
Answer ALL questions.
Each question is worth EIGHT marks.

A1 (a) Differentiate between ***Declarative Programming*** and ***Imperative Programming***. [2 marks]

(b) Write a Python program to print a ***triangle of permutations*** with a particular integer size, entered by the user. For example, if the user's input is 5, the program should display:

```
1
1 1
1 2 2
1 3 6 6
1 4 12 24 24
1 5 20 60 120 120
```

Note that each number on the triangle is computed by a particular nPr. For example, the third line is ${}_2P_0$, ${}_2P_1$ and ${}_2P_2$. The Python function `nPr` is given as follows:

```
import math
def nPr(n, r):
   f = math.factorial
return int(f(n) / f(n - r))
```

NOTE: You are required to include the code above in your answer. [4 marks]

(c) Define any **TWO** ***collection data types*** in Python. [2 marks]

A2 (a) Compare and contrast the **TWO** path searching algorithms: ***Best First search*** and ***A* search***. [4 marks]

(b) You are given the following Python code:

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
X = iris.data[:, 2:]  # petal length and width
y = iris.target
```

Write Python code to complete the following tasks:

(i) Create and train a ***K-Nearest Neighbors*** model with the **Iris** dataset. [2 marks]

(ii) Predict the Iris species if its *petal length and width* are 4 and 2 respectively. [2 marks]

A3 (a) Formulate the ***Map Colouring Problem*** as a ***Constraint Satisfaction Problem (CSP)***. *No coding is needed.* [3 marks]

(b) Illustrate ***unification*** in logic programming with an example. [2 marks]

(c) Suggest a cause of ***uncertainty*** in real world. [1 mark]

(d) Suggest **TWO** ways of ***probabilistic reasoning*** to solve problems with uncertain knowledge. [2 marks]

A4 (a) Briefly explain **TWO** differences between ***supervised learning*** and ***unsupervised learning***. [3 marks]

(b) ***Image analysis*** is a type of ***computer vision***. Suggest **THREE** other types of computer vision. [3 marks]

(c) List **TWO** ***cloud services*** which provides AI applications. [2 marks]

A5 (a) Explain what ***query*** is in a recommendation system and give an example of query. [3 marks]

(b) Write down any **TWO** ***activation functions*** which are commonly used in neural networks. [2 marks]

(c) State the **THREE** ***layers*** of a ***convolutional neural network (CNN)***. [3 marks]

Section B (60 marks)

This section contains FOUR questions.
Answer ALL questions from this section.
Each question is worth FIFTEEN marks.

B1 (a) (i) In this question, you are required to create **2020 Summer Olympics medal table** by utilizing *Pandas* library. Write Python code for assigning the following DataFrame to the variable, ***frame***.

	Rank	Team	Total
USA	1	United States	113
CHN	2	China	88
JPN(*)	3	Japan	58
GBR	4	Great Britain	65
ROC	5	ROC	71
AUS	6	Australia	46
NED	7	Netherlands	36
FRA	8	France	33
GER	9	Germany	37
ITA	10	Italy	400
	11-93	Remaining teams	493

[4 marks]

(ii) Write Python code for inserting the *Gold, Silver, Bronze* columns for the above table [You are not allowed to create a new DataFrame other than using ***frame*** as defined in **B1 (a)(i)**]:

	Rank	Team	Gold	Silver	Bronze	Total
USA	1	United States	39	41	33	113
CHN	2	China	38	32	18	88
JPN(*)	3	Japan	27	14	17	58
GBR	4	Great Britain	22	21	22	65
ROC	5	ROC	20	28	23	71
AUS	6	Australia	17	7	22	46
NED	7	Netherlands	10	12	14	36
FRA	8	France	10	12	11	33
GER	9	Germany	10	11	16	37
ITA	10	Italy	10	10	20	400
	11-93	Remaining teams	137	150	206	493

[3 marks]

QUESTION B1 CONTINUES ON THE NEXT PAGE

QUESTION B1 CONTINUES FROM THE PREVIOUS PAGE

B1 (a) (iii) Write Python code for removing the last row (Remaining teams) of the medal table:

	Rank	Team	Gold	Silver	Bronze	Total
USA	1	United States	39	41	33	113
CHN	2	China	38	32	18	88
JPN(*)	3	Japan	27	14	17	58
GBR	4	Great Britain	22	21	22	65
ROC	5	ROC	20	28	23	71
AUS	6	Australia	17	7	22	46
NED	7	Netherlands	10	12	14	36
FRA	8	France	10	12	11	33
GER	9	Germany	10	11	16	37
ITA	10	Italy	10	10	20	400

[1 mark]

(iv) List **ONE** benefit on creating arrays using the ***numpy.array*** method over the ***list*** method. [2 marks]

(b) Consider the following Python code, which implements the ***A* Algorithm***:

```
grid = [[0, 1, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 1, 0],
        [0, 1, 1, 1, 1, 0]]

heuristic = [[9, 8, 7, 6, 5, 4],
             [8, 7, 6, 5, 4, 3],
             [7, 6, 5, 4, 3, 2],
             [6, 5, 4, 3, 2, 1],
             [5, 4, 3, 2, 1, 0]]

init = [0, 0]
goal = [len(grid)-1, len(grid[0])-1]
cost = 1

delta = [[-1, 0 ], # go up
         [ 0, -1], # go left
         [ 1, 0 ], # go down
         [ 0, 1 ]] # go right
```

QUESTION B1 CONTINUES ON THE NEXT PAGE

QUESTION B1 CONTINUES FROM THE PREVIOUS PAGE

B1 (b) (*Continued*)

```
def search(grid, init, goal, cost, heuristic):
   closed = [[0 for col in range(len(grid[0]))] \
                  for row in range(len(grid))]
   closed[init[0]][init[1]] = 1
   expand = [[-1 for col in range(len(grid[0]))] \
                  for row in range(len(grid))]

   x = init[0]
   y = init[1]
   g = 0
   f = g + heuristic[x][y]
   open = [[f, g, x, y]]

   found = False  # flag that is set when search is complete
   resign = False # flag set if we can't find expand
   count = 0

   while not found and not resign:
      if len(open) == 0:
         resign = True
         return "Fail"
      else:
         open.sort()
         open.reverse()
         next = open.pop()
         f = next[0]
         g = next[1]
         x = next[2]
         y = next[3]

         expand[x][y] = count
         count += 1

         if x == goal[0] and y == goal[1]:
            found = True
         else:
            for i in range(len(delta)):
               x2 = x + delta[i][0]
               y2 = y + delta[i][1]
               if x2 >= 0 and x2 < len(grid) and \
                  y2 >=0 and y2 < len(grid[0]):
                  if closed[x2][y2] == 0 and grid[x2][y2] == 0:
                     g2 = g + cost
                     f = g2 + heuristic[x2][y2]
                     open.append([f, g2, x2, y2])
                     closed[x2][y2] = 1

   return expand
```

QUESTION B1 CONTINUES ON THE NEXT PAGE

QUESTION B1 CONTINUES FROM THE PREVIOUS PAGE

B1 (b) (*Continued*)

```
result = search(grid, init, goal, cost, heuristic)

for el in result:
    print(el)
```

Write down the output from the code. [5 marks]

B2 (a) (i) Consider the following implementation of the factorial function using the *logic programming* technique.

```
n = var()

@match(0)
def factl(n):
    return 1

@match(n)
def factl(n):
    return n * factl(n-1)
```

Write a Python function **`euler(n)`**, using the same technique, to compute the ***Euler's number*** of an integer `n`, where

```
euler(0) = 1/0! = 1
euler(1) = 1/0! + 1/1! = 2
euler(2) = 1/0! + 1/1! + 1/2! = 2.5
...
```

`n!` means the factorial of an integer **`n`**. [5 marks]

QUESTION B2 CONTINUES ON THE NEXT PAGE

QUESTION B2 CONTINUES FROM THE PREVIOUS PAGE

B2 (a) (ii) Imagine a number puzzle with **n×n** cells where **n≥3**. The integers from **1** to **n×n** are to be filled in to the puzzle. The following constraints should be satisfied:

- All cells should contain a unique number;
- The sum of the first and last elements of the ***first*** row should be **n×n**; *and*
- The sum of the first and last elements of the ***last*** row should be **n×n**.

Write Python code for solving this number puzzle as a CSP with the package **`python-constraint`**. You are given the following Python code:

```
from constraint import *
n = 3
vars = range(n*n)
dom = range(1, n*n+1)

problem = Problem()
problem.addVariables(vars, dom)
# Add your code here
solution = problem.getSolution()
```

[5 marks]

(b) You are given the following code fragment:

```
from sklearn.datasets import load_wine
wine = load_wine()
X = wine.data
y = wine.target
```

Write Python code with Scikit-Learn to complete the following tasks:

(i) Split X and y into training set and testing set with ratio **7:3**. [2 marks]

(ii) Use ***Decision Tree algorithm*** to train the given dataset and display the accuracy. [3 marks]

B3 (a) You are given the following code fragment:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
diabetes_X = diabetes_X[:, np.newaxis, 2]
```

Write Python code with Scikit-Learn to complete the following tasks:

(i) Split `diabetes_X` and `diabetes_y` into *training set* and *testing set* with ratio **4:1**; [2 marks]

(ii) Use *linear regression algorithm* to train the given dataset; [2 marks]

(iii) Display the *coefficients* of the model; [1 mark]

(iv) Display the *mean squared error* of the model; [1 mark]

(v) Display the *coefficient of determination* of the model. [1 mark]

(b) You are given the following 3 documents:

Document 1: ITP4514 is an AI course in SE
Document 2: SE students need to study ITP4514
Document 3: SE students take the AI exam

(i) Calculate the TF of the 3 documents, focusing on only the ***four words***: ITP4514, AI, SE, exam. [3 marks]

(ii) Calculate the IDF. [1 mark]

(iii) Calculate the TF-IDF of the 3 documents. [3 marks]

(iv) According to the TF-IDF, which ***word*** is the most important in the ***Document 2***? [1 mark]

B4 (a) Consider the following questions regarding recommender systems.

(i) ***Content-based filtering*** and ***collaborative filtering*** are two common candidate generation approaches. Define what ***content-based filtering*** does and give an example of content-based filtering in *video recommendation system.* [3 marks]

(ii) Freshness, diversity, and fairness can help improve a recommendation system. Suggest one method to enhance the ***freshness*** of the system. [2 marks]

(b) Consider the following questions regarding deep learning.

(i) Suggest **ONE** property of *artificial neural networks.* [1 mark]

(ii) What is the advantage of ***activation functions*** in *artificial neural networks*? [1 mark]

(iii) Write down **THREE** effective technologies to prevent ***overfitting***. [3 marks]

(c) Perform a ***convolution operation*** with the following image and filter. [5 marks]

Image

1	0	1	1
1	0	1	0
0	1	1	0
1	0	1	0

Filter

-1	1
-1	1

******* END OF PAPER *******