061306T4CSC

COMPUTER SCIENCE LEVEL 6

ICT/OS/CS/CR/08/6/A

Understand Artificial Intelligence Concepts

March/April 2025

TVET CURRICULUM DEVELOPMENT, ASSESSMENT AND CERTIFICATION COUNCIL (TVET CDACC)

PRACTICAL ASSESSMENT

INSTRUCTIONS TO ASSESSOR

1. Assess the candidate as the practical progresses observing the critical areas
2. You are required to mark the practical as the candidate perform the tasks
3. You are required to take video clips at critical points
4. Ensure the candidate has a name tag and registration code at the back and front

OBSERVATION CHECKLIST

Candidate's Name & Registration No.			
Assessors Name & Id Code			
Unit(s) of Competency	**Understand Artificial Intelligence Concepts**		
Venue of Assessment			
Date of Assessment			
Items to be Evaluated: *Please award marks as appropriate. Give a brief comment on your observation.*	**Marks Available**	**Marks Obtained**	**Comments**
TASK 1: Develop a Net-Salary calculator			
1. Observed computer laboratory rules. ***(Award 2 mark for each or zero)***	**2**		
2. Captured employee details - *Payroll Number* - *Name* - *Department* - *Gender* - *Basic Salary* ***(Award 1 mark for each or zero)***	**5**		
3. Calculated gross pay - *Basic salary + allowances* ***(Award 2 marks or zero)***	**2**		
4. Calculated the P.A.Y.E - *based on the gross pay* ***(Award 2 marks or zero)***	**2**		
5. Calculated deductions correctly - *N.H.I.F* - *N.S.S.F* - *Deductions* ***(Award 2 marks or 0 for each correct answer)***	**6**		
6. Computed the net pay correctly - Net Pay=Gross pay-deductions ***(Award 2 marks or 0 for each correct answer)***	**2**		
7. Displayed all the employee details. - *Payroll number* - *Name* - *Department* - *Gender*	**5**		

- *Net Pay* *(Award 1 mark or 0)*			
6. Displayed correct values for: - *Basic pay* - *NHIF* - *NSSF* - *Total Deductions* - *Net Pay* (*Award 1 mark or zero)*	**5**		
7. Designed a user interface as shown (*Award 1 mark or zero)*	**1**		
Sub-Total 1	**30**		
TASK 2: Create a disease diagnosis program.			
8. Launched the python IDE - *Python IDE* (*Award 2 marks or zero)*	2		
9. Captured patient's details - *Patient name* - *Gender* - *Age* - *Place of residence* (*Award 1 mark for each or 0)*	4		
10. Captured patient's symptoms - *Symptom 1* (*Award 2 marks or 0)*	**2**		
11. Captured patient's symptoms - *Symptom 1* (*Award 2 marks or 0)*	**2**		
12. Determined the possible diagnosis correctly - *Based on symptoms provided* *(Award 2 marks or 0)*	**2**		
13. Displayed a suitable message (*Award 2 marks for each entry or 0)*	**2**		
14. Formatted output as shown *(Award 3 marks for correct output or 0)*	**3**		

15. Compiled, debugged and executed program correctly (*Award 3 marks or 0)*	**3**		
Sub-Total 2	**20**		
GRAND TOTAL	**50**		

ASSESSMENT OUTCOME

The candidate was found to be:

Competent ☐ Not yet Competent ☐

(Please tick as appropriate)

(The candidate is competent if the candidate obtains at least 50%)

Feedback from the Candidate:

Feedback to the Candidate:

Candidate Signature **Date:**

_____ _____

Assessor's Signature **Date**

_____ _____

Sample code for Task 1

```
# Function to calculate the deductions and net pay
def calculate_payroll():
    # Get employee details
    payroll_number = input("Enter payroll number: ")
    name = input("Enter employee's name: ")
```

```
gender = input("Enter employee's gender (M/F): ")
department = input("Enter department: ")
basic_salary = float(input("Enter basic salary: "))

# Constants
house_allowance = 6500  # Uniform house allowance
medical_allowance = 5500  # Uniform medical allowance

# Calculate gross pay
gross_pay = basic_salary + house_allowance + medical_allowance

# Deductions
nhif = 0.02 * gross_pay  # 2% of gross pay for NHIF
nssf = 0.03 * basic_salary  # 3% of basic salary for NSSF

# Assume a fixed PAYE rate for simplicity (this can be extended based on income brackets)
# For this example, let's assume PAYE is a flat 10% of the gross pay
paye = 0.10 * gross_pay

# Calculate total deductions
total_deductions = paye + nhif + nssf

# Calculate net pay
net_pay = gross_pay - total_deductions

# Print the payroll details
print("\nPayroll Summary:")
print(f"Payroll Number: {payroll_number}")
```

```
    print(f"Name: {name}")
    print(f"Gender: {gender}")
    print(f"Department: {department}")
    print(f"Basic Salary: Ksh {basic_salary:,.2f}")
    print(f"House Allowance: Ksh {house_allowance:,.2f}")
    print(f"Medical Allowance: Ksh {medical_allowance:,.2f}")
    print(f"Gross Pay: Ksh {gross_pay:,.2f}")
    print(f"P.A.Y.E: Ksh {paye:,.2f}")
    print(f"N.H.I.F (2%): Ksh {nhif:,.2f}")
    print(f"N.S.S.F (3% of Basic Salary): Ksh {nssf:,.2f}")
    print(f"Total Deductions: Ksh {total_deductions:,.2f}")
    print(f"Net Pay: Ksh {net_pay:,.2f}")

# Call the function to calculate payroll for a single employee
calculate_payroll()
```


Sample code for Task 2

```
# Function to diagnose based on symptoms
def diagnose(symptoms):
    # A dictionary of symptoms and their corresponding diagnoses
    disease_symptoms = {
        ("vomiting ", " diarrhea "): "You may be suffering from the Typhoid.",
        ("fever", " vomiting "): "You may be suffering from Malaria.",
        ("wheezing sound ", " Chest pain "): "You may have a respiratory infection or pneumonia.",
        ("Frequent urination ", " fluctuations of sugar levels "): "You may be suffering from
Diabetes.",
        ("fatigue", "muscle pain"): "You may have a viral infection or even the Flu.",
        ("fever", "chills"): "You may have an infection or a common cold.",
    }
```

```
    # Check if the given symptoms match any known disease
    for symptom_pair, diagnosis in disease_symptoms.items():
        if set(symptoms) == set(symptom_pair):
            return diagnosis

    # If no known diagnosis is found
    return "Diagnosis unavailable based on the provided symptoms. Please consult a doctor."

# Function to capture patient information
def capture_patient_info():
    print("Welcome to Jeshi Hospital. Please provide your details.")

    # Get patient details
    name = input("Enter the patient's name: ")
    gender = input("Enter the patient's gender (M/F): ")
    age = int(input("Enter the patient's age: "))
    place_of_residence = input("Enter the patient's place of residence: ")

    # Display the captured information
    print("\nPatient Information:")
    print(f"Name: {name}")
    print(f"Gender: {gender}")
    print(f"Age: {age}")
    print(f"Place of Residence: {place_of_residence}")

    return name, gender, age, place_of_residence
```

```
# Function to capture and diagnose symptoms
def capture_and_diagnose():
    # Capture patient information
    name, gender, age, place_of_residence = capture_patient_info()

    # Get the symptoms from the user
    print("\nPlease enter up to two symptoms you are experiencing.")
    symptoms = []

    # Get first symptom
    symptom1 = input("Enter the first symptom: ").lower()
    symptoms.append(symptom1)

    # Ask for the second symptom if the user wants
    symptom2 = input("Enter the second symptom (if any, or press Enter to skip): ").lower()
    if symptom2:
        symptoms.append(symptom2)

    # Diagnose based on the symptoms provided
    diagnosis = diagnose(symptoms)

    # Output the diagnosis
    print("\nDiagnosis:")
    print(diagnosis)

# Run the system
capture_and_diagnose()
```