تاریخ: / /

موضوع: _____

Name: Ahmad Wali Safi
Department: Information Technology
Attendance No: 5

Object orinted programming Home Work

Q 1:

```
class person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
person1 = person("Ahmad wali", 20)
print(f"name: {person1.name}")
print(f"Age: {person1.age}")
```

Q2:

```
class person:
    def __init__(self, name)
        self.name = name
    def greet(self)
        print(f"Hello, {self.name} Have a great day")
person1 = person("Ahmad wali")
person1.greet()
```

تاریخ: / /

موضوع:

Q3:

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
    def display_details(self):
        print(f"Car Details: {self.year} {self.make} {self.model}")
my_car = Car("Toyota", "Corolla", 2024)
my_car.display_details()
```

Q4:

```
class Circle:
    def __init__(self, radius):
        self.radius = radius
    def calculate_area(self):
        return 3.14 * self.radius ** 2
circle = Circle(5)
area = Circle.Calculate_area()
print(f"The area of the circle with radius {circle.radius}
is: {area}")
```

تاريخ: / / موضوع:

Q5:

```
Class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)
```

Q6:

```
Class Animal:
    def speak(self):
        pass
Class Dog(Animal)
    def speak(self):
        return "Bark"
Class Cat(Animal):
    def speak(self):
        return "mew"
```

Q7.

```
class shape:
    def area(self):
        pass
class square(shape):
    def __init__(self, side_length)
        self.side_length** 2 = sidelength
    def area(self):
        return self.side_length ** 2
class Triangle(shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def area(self):
        return 0,5 * self.base * self.height
```

Q8:

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
class Manager(Employee):
    def __init__(self, name, salary, Department):
```

تاریخ: / /

```
        super().__init__(name, salary)
        self.department = department
```

Q9:

```
class Vehicle:
    def drive(self):
        return "Vehicle is being driven"
class Bike(Vehicle):
    def drive(self):
        return "Bike can be driven"
class Truck(Vehicle):
    def drive(self):
        return "Truck can be driven"
```

Q10:

```
class Bird:
    def fly(self):
        print("This bird can fly")
class Eagle(Bird):
    pass
class Penguin(Bird):
    def fly(self):
        print("Penguin cannot fly")
```

تاريخ: / /

موضوع:

Q11:

```
Class Account:
    def __init__(self, initial_balance=0):
        self._balance = initial_balance
    def deposit(self, amount):
        self._balance += amount
    def withdraw(self, amount):
        if self._balance >= amount:
            self._balance -= amount
        else:
            Print("Insufficient funds")
```

Q12:

```
Class Book:
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
    def get_title(self):
        return self.title
    def set_title(self, title):
        self.title = title
    def get_author(self, author):
        return self._authour
```

تاريخ: / /

موضوع:

```
    def set_author(self, author):
        self._author = author
    def get_pages(self):
        return self._pages
    def set_pages(self, pages):
        self._pages = pages
```

Q13:

```
class Laptop:
    def __init__(self, brand, model, price):
        self._brand = brand
        self._model = model
        self._price = price
    def apply_discount(self, discount):
        self._price -= (self._price * discount / 100)
    def display_details(self):
        print(f"Laptop: {self._brand} {self._model}, Price $:
        {self._Price}")
```

تاريخ: / /

موضوع:

Q14:

```
class Bank account:
    def __init__(self, account_number, initial_balance=0):
        self._account_number = account_number
        self._balance = initial_balance
    def deposit(self, amount):
        self._balance += amount
    def withdraw(self, amount):
        if self._balance >= amount:
            self._balance -= amount
        else:
            Print("Insufficient funds")
    def check_balance(self):
        return self._balance
```

Q15:

```
class student:
    def __init__(self, name, grade, age):
        self._name = name
        self._grade = grade
        self._age = age
    def get_name(self):
        return self._name
```

```
    def set_name(self, name):
        self._name = name
    def get_grade(self):
        return self._grade
    def set_grade(self, grade):
        self._grade = grade
    def get_age(self):
        return self._age
    def set_age(self, age):
        self._age = age
    def display_details(self):
        Print(f"Name: {self._name}, Grade: {self._grade},
        Age: {self._age}")
```

Q16:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
class Library:
    def __init__(self, name):
        self.name = name
        self.books = []
```

تاریخ: / /

موضوع:

```
Def add_book (self, title, author):
    new_book = Book(title, author)
    self.books.append(new_book)
def remove_book(self, title):
    self.book = [book for book in self.books
    if book.title != title]
```

Q17.

```
class student:
    def __init__ (self, name, age):
        self.name = name
        self.age = age
class School:
    def __init__ (self, name):
        self.name = name
        self.students = []
    def add_student (self, student):
        self.students.append(Student)
    def remove_student (self, student):
        self.students.remove(student)
```

Q18:

```
class person:
    def __init__(self, name)
        self.name = name
class Team:
    def __init__(self, name):
        self.name = name
        self.member = []
    def add_member(self, person):
        self.members.append(person)
    def remove_member(self, person):
        self.members.remove(person)
```

Q19:

```
class Employee:
    def __init__(self, name, emp_id):
        self.name = name
        self.emp_id = emp_id
class Company:
    def __init__(self, name):
        self.name = name
        self.employee = []
```

تاريخ: / /

موضوع:

```
def add_employee (self, employee):
    self.employee.eppend (employee)
def remove_employee (self, employee):
    self.employee.remove (employee)
```

Q20:

```
Class Animal:
    def_init_ (self, species):
        self.species = species
class Zoo:
    def_init_ (self, name):
        self.name = name
        self.animals = II
    def add_Animal (self, animal):
        self.Animal.append (animal)
    def remove_animal (self, animal()):
        self.animal.remove (animal)
```

تاریخ: / /

موضوع:

Q21:

```
Class FileManager:
    def __init__(self, file_name):
        self.file_name = file_name

    def read_file(self):
```

Q21:

```
class FileManager:
    def read_file(self, file_name):
        file = open(file_name, "r")
        content = file.read()
        file.close()
        return content
    def write_file(self, file_name, data):
        file = open(file_name, "w")
        file.write(data)
        file.close()
```

تاریخ: / /

موضوع:

Q22:

```
class Log:
    def __init__(self, log_file):
        self.log_file = log_file

    def write_error(self, error_message):
        with open(self.log_file, "a") as file:
            file.write(f"Error: {error_message}\n")
```

Q23:

```
class Config:
    def __init__(self, file_path):
        self.config = {}
        with open(file_path, "r") as file:
            for line in file:
                key, value = line.strip().split("=")
                self.config[key.strip()] = value.strip()
    def get_setting(self, key):
        return self.config.get(key, none)
    def set_setting(self, key, value):
        self.config[key] = value
```

تاريخ: / / موضوع:

Q24: import sqlite3

```
class Database:
    def __init__(self, name):
        self.name = name
        self.connection = none
        self.cursor = none
        self.connect()
    def connect(self):
        try:
            self.connection = sqlite3.connect(self.name)
            self.cursor = self.connection.cursor()
            print("Connected to the database")
        except sqlite3.Error as e:
            print(f"Error connecting to the database:{e}")
    def execute_query(self, query):
        try:
            self.cursor.execute(query)
            self.connection.commit()
            Print("Query executed successfully")
        except sqlite3.Error as e:
            print(f"Error executing query: {e}")
            print("Connection faild")
```

تاريخ: / / موضوع: _____

Q25:

```
class Report:
    def __init__(self, file_name):
        self.file_name = file_name
    def generate_report(self):
        try:
            with open(self.file_name, "r") as file:
                data = file.read()
                print("report gerated")
        except FileNotFoundError:
            Print("file Not found")
        except IOError:
            Print("Could not read file")
```

Q26:

```
class Ticket:
    def __init__(self, movie_name, seat_number, Price):
        self.movie_name = movie_name
        self.seat_number = seat_number
        self.Price = Price
    def display_details(self):
        Print(f"movie: {self.move_name}")
        Print(f"seat number: {self.seat_number}")
        Print(f"Price: {self.Price}")
```

```
    def apply_discount(self, discount):
        self.price = self.price * (1 - discount)
```

Q27:

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price
class ShoppingCart:
    def __init__(self):
        self.Item = []
    def add_item(self, item):
        self.item.append(item)
    def remove_item(self, item):
        self.item.remove(item)
    def display_items(self):
        for item in self.items:
            print(f"Item {item.name}, Price {item.price}")
```

تاريخ: / /

موضوع:

Q28:

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

class Restaurant:
    def __init__(self, name):
        self.name = name
        self.menu = []
    def add_item(self, item_name):
        self.menu.append(item)
    def remove_item(self, item_name):
        self.menu.remove(item_name)
    def disply_menu(self):
        print(f"menu for {self.name}")
        for item in self.menu:
            print(f"{item.name} - ${item.price}")
```

Q29:

```
class Flight:
    def __init__(self, flight_number, destination):
        self.flight_number = flight_number
        self.destination = destination
        self.passanger = []

    def add_passanger(self, person):
        self.passanger.append(person)

    def remove_passanger(self, person):
        self.passanger.remove(person)

    def display_passanger(self):
        for passanger in self.passanger:
            print(passanger.name)

class person:
    def __init__(self, name):
        self.name = name
```

تاریخ: / /

موضوع:

Q30:

```
class Room:
    def init_(self, room_number,):
        self.room_number = rome_number
        self.is_occupied = false
class Hotel:
    def init_(self, name):
        self.name = name
        self.rooms = []
    def add_room(self, room):
        self.rooms.append(room)
    def book_room(self, room_number):
        for room in self.rooms:
            if room.room_number = room_number and not
            room is occupied:
            room.is occoupied = True
            return f"Room {room_number} has been booked"
        return f"Room {room_number} is not available"
    def checkout_room(self, room_number):
        for room in self.rooms:
            if room.room_number = room_number and room
            is occupide:
            room is occoupide = False
```

return f"Room {room_number} is checked"
return f"Room {room_number} is not occupaide."

Q36:

```
import tkinter as tk
class CounterApp:
    def __init__(self, root):
        self.root = root
        self.Counter = 0
        self.Lable = tk.Lable(root, text = "Counter:0")
        self.lable.Pack()
        self.increment_button = tkButton(root, terte "I",
        Command = self.increment)
        self.increment_button.pack()
        self.decrement_button = tk.Button(root, text =
        "decrement", Camand = self.decrement)
        self.decremen_button.Pack()
    def increment(self):
        self.Counter += 1
        self.lable.config(text = f"Counter: {self.counter}")
    def decrement(self):
        self.Counter -= 1
        self.Lable.config(text = f"Counter: {self.Counter}")
```

تاریخ: / /

موضوع:

```
root = tk.Tk()
app = CounterApp(root)
root.mainloop()
```

Q37:

```
import tkinter as tk
class ToDoApp
    def __init__(self, root):
        self.root = root
        self.root.title("To-Do List App")
        self.tasks = []
        self.task_entry = tk.Entry(self.root, width=50)
        self.task_entry.pack(pady=20)
        self.add_button = tk.Button(self.root, text="Add Task",
        command = self.add_task)
        self.add_button.pack(pady=10)
        self.task_list = tk.Listbox(self.root, width=50)
        self.task_list.pack(pady=20)
        self.remove_button = tk.Button(self.root, text="
        Remove Task", command = self.remove_task)
        self.remove_button.pack(pady=10)
```

```
def add_task(self):
    task = self.task_entry.get()
    if task:
        self.tasks.append(task)
        self.task_list.insert(tk.END, task)
        self.task_entry.delete(0, tk.END)
def remove_task(self):
    task_index = self.task_list.curselection()
    if task_index:
        task = self.tasks.pop(task_index[0])
        self.task_list.delete(task_index)
root = tk.Tk()
app = ToDoApp
root.mainloop()
```

Q38:

```
import tkinter as tk
class CalculatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Simple Calculator")
        self.display = tk.Entry(self.root, width=20,
        borderwidth=5)
```

```
self.display.grid(row=0, column=0, columnspan=4
, padx=10, pady=10)
buttons=[
    '7', '8', '9', '/',
    '4', '5', '6', '*',
    '1', '2', '3', '-',
    'C', '0', '=', '+',
]
row=1
col=0
for button_text in buttons:
    tk.Button(self.root, text=button_text,
    width=5, command=lambda text=button_text:
    self.on_button_click(text)).grid(row=row,
    column=col, padx=5, pady=5)
    col+=1
    if col>3
        col=0
        row+=1
def on_button_click(self, text):
    if text=="=":
        try:
            result=eval(self.display.get())
```

```
            self.display.delete(0, tk.END)
            self.display.insert(0, str(result))
        except:
            self.display.delete(0, tk.END)
            self.display.insert(0, "Error")
    elif text == 'C':
        self.display.delete(0, tk.END)
    else:
        self.display.insert(tk.END, text)
root = tk.Tk()
app = CalculatorApp(root)
root.mainloop()
```

Q39:

```
import tkinter as tk
class LoginApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Login Form")
        self.label_username = tk.Label(root, text="Username:")
        self.label_username.pack()
        self.entry_username = tk.Entry(root)
        self.entry_username.pack()
```

```
        self.lable_password = tk.Label(root, text="password")
        self.lable_password.pack()
        self.entry_password = tk.Entry(root, show="*")
        self.Entry_password.pack()
        self.btn_login = tk.Button(root, text="Login", command=
        self.login)
        self.btn_login.pack()
    def login(self):
        username = self.entry_username.get()
        password = self.entry_password.get()
        if username == "ahmad wali" and password == "aalka":
            print("Login successfully")
        else:
            print("Login failed")
root = tk.Tk()
app = LoginApp(root)
root.mainloop()
```

تاريخ: / /

موضوع:

Q40:

```
import tkinter as tk
class WeatherApp:
    def __init__(self, master):
        self.master = master
        master.title("Weather Information")
        self.label = tk.Label(master, text="Today's
        Weather Forecast:", font=('Arial', 16))
        self.label.pack()
        self.temprature_label = tk.Label(master, text=
        "Temprature: 33°C", font=('Arial', 14))
        self.temprature_label.pack()
        self.weather_label = tk.Label(master, text="
        Weather: Sunny", font=('Arial', 14))
root = tk.Tk()
weather_app = WeatherApp(root)
root.mainloop()
```