POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
Republic of the Philippines
Manila City Campus

COLLEGE OF COMPUTER AND INFORMATION SCIENCES
Department of Mechatronics and Robotics

MIDTERM LABORATORY EXAMINATION FOR

PROG 102

DATA STRUCTURE AND ALGORITHMS

Compiled by: Checked by:

Prof. Sydrik A, Mandino., LPT., Mech
Chairperson For Mechatronics

Approved by:

John Dustin D. Santos, MIT
Instructor III
Chairperson, Department of Information Technology
Gender and Development Coordinator

MIDTERM EXAMINATION IN PROG 102

POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
Republic of the Philippines
Manila City Campus

100 PTS

Laboratory Exam: Object-Oriented Programming with Data Structures

Title: "City Services Management System"
Focus: Simple Object-Oriented Programming (OOP) concepts and Data Structures applied to daily city applications.
Programming Environment: Python in VS Code/PyCharm (no external extensions or libraries installed).

Problem Statement

You are tasked to create a City Services Management System in Python to manage services such as garbage collection, water supply, and electricity repair requests. Your program should allow the city residents to submit service requests, view pending requests, and mark completed requests.

Functional Requirements

1. **Add a Service Request:** Residents can submit a request by specifying their name, service type (e.g., garbage collection, water supply, electricity repair), and address.
2. **View Pending Requests:** Display all service requests that are not yet completed.
3. **Mark a Request as Completed**: Mark a specific service request as completed.
4. **Exit the System:** Safely terminate the program.

Class Design

Class	Attributes	Methods
Service	`request_id`, `name`, `address`, `service_type`, `status`	`__init__()`: Initializes a service request. -`__str__()`: Formats the object as a readable string.
CityServices	`requests`: A list to store all service objects	`add_request()`: Adds a new service request. -`view_requests()`: Displays all pending requests. -`mark_completed()`: Marks a request as completed.

PART A: Code Implementation

```
import uuid
from enum import Enum

# Enum for predefined service statuses
class ServiceStatus(Enum):
    PENDING = "Pending"
    IN_PROGRESS = "In Progress"
    INACTIVE = "Inactive"
    IN_QUEUE = "In Queue"
    COMPLETED = "Completed"
    CANCELLED = "Cancelled"
    DECLINED = "Declined"

# Predefined service types
SERVICE_TYPES = [
    "Garbage Collection",
    "Water Supply",
    "Electricity Repair",
    "Road Maintenance",
    "Public Safety",
    "Appoint Public Meetings"
]

# Class to represent a single service request
class ServiceRequest:
    def __init__(self, request_id, name, address, service_type,
description=None):
        self.request_id = request_id
        self.name = name
        self.address = address
        self.service_type = service_type
        self.description = description
        self.status = ServiceStatus.PENDING  # Default status is Pending
    def __str__(self):
        return f"""
        Request ID: {self.request_id}
        Name: {self.name}
        Address: {self.address}
        Service Type: {self.service_type}
        Description: {self.description or 'No description provided'}
        Status: {self.status.value}
        """

# Class to manage all client service requests
class CityService:
```

```
    def __init__(self):
        self.requests = []
    def generate_request_id(self):
        """ Generate a unique request ID """
        return str(uuid.uuid4())[:8]  # Generates an 8-character unique ID
    def add_request(self, name, address, service_type, description=None):
        """ Add a new service request """
        request_id = self.generate_request_id()
        new_request = ServiceRequest(request_id, name, address, service_type,
description)
        self.requests.append(new_request)
        print(f"\nRequest added. Your Request ID: {request_id}")
    def view_pending_requests(self):
        """ Display all pending service requests """
        pending_requests = [request for request in self.requests if
request.status == ServiceStatus.PENDING]
        if not pending_requests:
            print("\nNo pending service requests found.")
        else:
            print("\nPending Service Requests:")
            for request in pending_requests:
                print(request)
    def update_request_status(self, request_id):
        """ Update the status of a request """
        request = next((req for req in self.requests if req.request_id ==
request_id), None)
        if not request:
            print("Request ID not found.")
            return
        print("\nSelect a new status:")
        for index, status in enumerate(ServiceStatus, start=1):
            print(f"{index}. {status.value}")
        while True:
            try:
                status_choice = int(input("\nEnter the number for the new
status: "))
                if 1 <= status_choice <= len(ServiceStatus):
                    request.status = list(ServiceStatus)[status_choice - 1]
                    print(f"Request ID {request_id} updated to
{request.status.value}")
                    return
                else:
                    print(f"Invalid selection. Choose a number from 1 to
{len(ServiceStatus)}.")
            except ValueError:
                print("Invalid input. Please enter a valid number.")

# Main Program
```

```
def main():
    city_service = CityService()

    while True:
        print("\nCity Service Management System")
        print("1. Add Service Request")
        print("2. View Pending Requests")
        print("3. Update Request Status")
        print("4. Exit")

        try:
            choice = int(input("Enter your choice: "))
        except ValueError:
            print("Invalid input. Please enter a number from 1 to 4.")
            continue

        if choice == 1:
            if choice == 1:
                while True:
                    name = input("\nEnter the person's name: ").strip()
                    address = input("Enter the address: ").strip()
                    if not name or not address:
                        print("Name and address cannot be empty. Please try
again.")
                        continue
                    selected_services = []

                    # Multi-service selection loop
                    while True:
                        # Display available service types
                        print("\nAvailable Service Types:")
                        for index, service in enumerate(SERVICE_TYPES, start=1):
                            print(f"{index}. {service}")
                        print("Type 'exit' if you are done selecting service
types.")
                        # Validate user input for service type
                        service_choice = input(
                            "\nEnter the number corresponding to your chosen
service type (or type 'exit' to finish): ").strip()
                        if service_choice.lower() == "exit":
                            break
                        try:
                            service_choice = int(service_choice)
                            if 1 <= service_choice <= len(SERVICE_TYPES):
                                service_type = SERVICE_TYPES[service_choice - 1]
                                selected_services.append(service_type)
                                print(f"Service '{service_type}' added to your
request.")
```

```
                            else:
                                print(f"Invalid choice. Please enter a number
from 1 to {len(SERVICE_TYPES)}.")
                        except ValueError:
                            print("Invalid input. Please enter a valid number or
type 'exit'.")

                    if not selected_services:
                        print("No services selected. Please try again.")
                        continue
                    # Prompt for a description
                    description = input("Enter a brief description of the issue
(optional): ").strip()

                    # Add each selected service request for this person
                    for service_type in selected_services:
                        city_service.add_request(name, address, service_type,
description)

                    # Ask if the user wants to enter another request
                    another_entry = input("\nWould you like to add another
request? (yes/no): ").strip().lower()
                    if another_entry != "yes":
                        break

        elif choice == 2:
            city_service.view_pending_requests()
        elif choice == 3:
            request_id = input("\nEnter Request ID to update: ")
            city_service.update_request_status(request_id)
        elif choice == 4:
            print("\nExiting the system. Goodbye.")
            break
        else:
            print("Invalid choice. Please select an option from 1 to 4.")


# Run the program
if __name__ == "__main__":
    main()
```

PART B: Guide Questions

Class Design:

- What are the attributes of the `Service` class, and why are they important?
- In my laboratory exam, the attribute of Service class can be repurposed to assign a value, modify, replace, and even delete. In the case of City Management Service, Service can be the main object of the program but it can modify through calling the attributes.

- Why do we use a `list` to store all requests in the `CityServices` class?
- When we stored value in memory we can write it in the form of "list" since it store's memory and after that it is available to call depends on the user modifies preferences. Additionally, list is **mutable** content can be changes and modifies, accessible for revision and calling.

Table Analysis

Input	**Expected Output**

Add a service request
{
person_name =John D., address=342 Main St, available_service_type =User.Selected, service_bried_description=None
}
	Service (available_service_type) added to your request.
	
	Request added, Your Request ID: 1

View Pending Requests	Request ID: cf46df86 Name: John D. Address: 342 Main St. Service Type: (available_service_type) Description: None Status: Pending #Default

Update Request Status
{
Input Request ID =request_ID, select_new.status = number_of_new_status.
}
	request_ID updated new_status