Errors and Data Structures

Activity 1: Errors

Incorporate the following code into a Python program to handle exceptions.

```
try:
    # do something
    pass

except ValueError:
    # handle ValueError exception
    pass

except (TypeError, ZeroDivisionError):
    # handle multiple exceptions
    # TypeError and ZeroDivisionError
    pass

except:
    # handle all other exceptions
    pass
```

```
def get_user_info():
    try:
        # Ask for user's name
        name = input("Enter your name: ")

        # Check if name is empty
        if not name:
            raise ValueError("Name cannot be empty.")

        # Ask for user's age
        age = input("Enter your age: ")

        # Attempt to convert the age to an integer
        age = int(age)

        # If the age is less than 0, raise a ValueError
        if age < 0:
            raise ValueError("Age cannot be negative.")

        print(f"Hello, {name}! You are {age} years old.")

    except ValueError as e:
        # Handle ValueError (invalid input or negative age, or empty name)
        print(f"Error: {e}")

    except (TypeError, ZeroDivisionError):
        # Handle TypeError and ZeroDivisionError if they occur
        print("Error: A type error or division by zero occurred.")

    except:
        # Handle any other exceptions
        print("An unexpected error occurred. Please try again.")

get_user_info()
```

```
Enter your name: Tim
Enter your age: 21
Hello, Tim! You are 21 years old.
```

Activity 2: Data Structures

Set operations include:

Union
Intersection
Difference
Symmetric difference

1. Explain a use for each of these set operations within the context of your summative assessment.

Union (|)

The union operation combines the elements from two sets, removing any duplicates. This could be used to manage the robot’s schedule of deliveries.

Intersection (&)

The intersection operation finds common elements between two sets. This is useful when the robot needs to check which tasks are overlapping between two schedules

Difference (-)

The difference operation returns the elements that are in one set but not in another. It can be used to track tasks that are unique to a specific robot or to identify tasks that are pending.

Symmetric Difference (^)

The symmetric difference operation returns the elements that are in either of the two sets, but not in both. This is useful when you want to track tasks that are unique to each set and exclude the common ones.

2. Write a Python program to carry out a linear search on a list data structure.

```python
def linear_search(robot_names, target_name):
   # Loop through the list to find the target_name
   for index, name in enumerate(robot_names):
      if name == target_name:
         return index  # Return the index of the found name
   return -1  # Return -1 if the name is not found

robot_names = ["RoboX", "AutoBot", "DeliverBot", "Speedster", "RoboDroid"]
target_name = input("Enter the robot name to search for: ")

result = linear_search(robot_names, target_name)

# Output the result
if result != -1:
   print(f"{target_name} found at position {result}.")
else:
   print(f"{target_name} not found in the list.")
```

```
Enter the robot name to search for: RoboDroid
RoboDroid found at position 4.
```