**Assignment No 1:  
Task 1: Implementing a Stack using Python (20 Marks)**

**Problem Statement:**

Implement a stack using a class in Python. The stack should support the following operations:

* push(element): Adds an element to the top of the stack.
* pop(): Removes and returns the top element from the stack.
* peek(): Returns the top element without removing it.
* is\_empty(): Returns True if the stack is empty, False otherwise.
* size(): Returns the number of elements in the stack.

**Expected Output Example:**

stack = Stack()

stack.push(10)

stack.push(20)

print(stack.pop()) # Output: 20

print(stack.peek()) # Output: 10

print(stack.size()) # Output: 1

print(stack.is\_empty()) # Output: False

**Task 2: Implementing a Queue using Python (20 Marks)**

**Problem Statement:**

Implement a queue using a class in Python. The queue should support the following operations:

* enqueue(element): Adds an element to the end of the queue.
* dequeue(): Removes and returns the front element from the queue.
* front(): Returns the front element without removing it.
* is\_empty(): Returns True if the queue is empty, False otherwise.
* size(): Returns the number of elements in the queue.

**Expected Output Example:**

queue = Queue()

queue.enqueue(5)

queue.enqueue(15)

print(queue.dequeue()) # Output: 5

print(queue.front()) # Output: 15

print(queue.size()) # Output: 1

print(queue.is\_empty()) # Output: False

**Task 3: Sorting Algorithms (30 Marks)**

**Problem Statement:**

Implement the following sorting algorithms and compare their performance:

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**

For each sorting algorithm, write a function that takes a list of integers as input and returns a sorted list. Implement a performance comparison by sorting a list of 1000 random integers and measuring the execution time for each algorithm.

**Expected Output Example:**

arr = [64, 34, 25, 12, 22, 11, 90]

print(bubble\_sort(arr)) # Output: [11, 12, 22, 25, 34, 64, 90]

print(selection\_sort(arr)) # Output: [11, 12, 22, 25, 34, 64, 90]

print(insertion\_sort(arr)) # Output: [11, 12, 22, 25, 34, 64, 90]

**Performance Comparison Output Example:**

Bubble Sort Time: 0.0023s

Selection Sort Time: 0.0018s

Insertion Sort Time: 0.0015s

**Task 4: Insertion Algorithm in a Sorted List (30 Marks)**

**Problem Statement:**

Write a function that inserts an element into a sorted list while maintaining the sorted order. The function should return the updated sorted list.

**Expected Output Example:**

sorted\_list = [10, 20, 30, 40, 50]

print(insert\_in\_sorted\_list(sorted\_list, 35)) # Output: [10, 20, 30, 35, 40, 50]