

SET 1

Q1. (Data Structures – 5 Marks)

Using AI assistance, write a Python program to implement a **Stack** using a list.
Perform the following operations:

- Push 5 elements
 - Pop 2 elements
 - Display the remaining stack
- Explain how AI helped generate or optimize your structure.

Q2. (Web Frontend – 5 Marks)

With AI tools, develop an interactive **To-Do List App** using HTML, CSS, and JavaScript.
Include features:

- Add task
- Delete task
- Mark as completed

SET 2

Q1. (Algorithms – 5 Marks)

Using AI assistance, generate Python code to implement **Merge Sort**.

Run the algorithm on the list: 45, 12, 3, 67, 34, 21.

Explain the time complexity and how AI improved correctness.

Q2. (Web Frontend – 5 Marks)

Use AI to generate a webpage showing a **student registration form** with:

- Input fields (Name, Email, Course)
- Submit button
- JavaScript form validation



SET 3

Q1. (Data Structures – 5 Marks)

Use AI to create a Python program implementing a **Queue** using the `collections.deque` module.

Perform:

- Enqueue 4 values
 - Dequeue 1 value
 - Display queue
- Explain AI's suggestion accuracy.

Q2. (Algorithms – 5 Marks)

With AI assistance, implement **Binary Search** in Python.

Test it on the sorted array:

[10, 20, 30, 40, 50, 60] to search for the number **40**.

SET 4

Q1. (Algorithms – 5 Marks)

Use AI to generate Python code for **Bubble Sort** and **Insertion Sort**.

Compare their execution time using Python's `time` module.

Q2. (Web Frontend – 5 Marks)

With AI help, create a webpage showing a **dynamic product list** using JavaScript arrays.

Each product must display:

- Name
 - Price
 - "Add to Cart" button
-

SET 5

Q1. (Data Structures – 5 Marks)

Using AI tools, implement a **Linked List** in Python with operations:

- Insert at beginning
 - Insert at end
 - Delete a node
- Display the list after operations.

Q2. (Web Frontend – 5 Marks)

With AI assistance, generate a responsive **Portfolio Website Layout** including:

- Header
- About section
- Projects section
- Contact form

SET-4:

Task-1:

Code:

```
Task1.py > ⏺ measure_time
1 import argparse
2 import random
3 import time
4 from typing import List, Callable
5
6 #!/usr/bin/env python3
7 """
8 Task1.py
9
10 Compare Bubble Sort and Insertion Sort execution times using Python's time module.
11
12 Usage:
13     python Task1.py          # runs with defaults (size=1000, runs=5)
14     python Task1.py --size 2000 --runs 3
15 """
16
17
18
19 def bubble_sort(a: List[int]) -> None:
20     """In-place optimized bubble sort."""
21     n = len(a)
22     for i in range(n):
23         swapped = False
24         # Last i elements are already in place
25         for j in range(0, n - i - 1):
26             if a[j] > a[j + 1]:
27                 a[j], a[j + 1] = a[j + 1], a[j]
28                 swapped = True
29         if not swapped:
30             break
```

```
def insertion_sort(a: List[int]) -> None:
    """In-place insertion sort."""
    for i in range(1, len(a)):
        key = a[i]
        j = i - 1
        # Move elements of a[0..i-1], that are greater than key, one position ahead
        while j >= 0 and a[j] > key:
            a[j + 1] = a[j]
            j -= 1
        a[j + 1] = key

def measure_time(sort_fn: Callable[[List[int]], None], data: List[int]) -> float:
    """Measure execution time (seconds) of sort_fn on a copy of data."""
    arr = data.copy()
    t0 = time.perf_counter()
    sort_fn(arr)
    t1 = time.perf_counter()
    # sanity check: ensure sorted
    if arr != sorted(data):
        raise RuntimeError(f"{sort_fn.__name__} did not sort correctly")
    return t1 - t0

def main():
    parser = argparse.ArgumentParser(description="Compare Bubble Sort and Insertion Sort timings.")
    parser.add_argument("--size", type=int, default=1000, help="Size of the random list (default: 1000)")
    parser.add_argument("--runs", type=int, default=5, help="Number of runs to average (default: 5)")
    args = parser.parse_args()

    size = args.size
    runs = args.runs
```

```

bubble_times = []
insertion_times = []

for r in range(runs):
    # generate random list (with possible duplicate values)
    data = [random.randint(0, size * 10) for _ in range(size)]

    bt = measure_time(bubble_sort, data)
    it = measure_time(insertion_sort, data)

    bubble_times.append(bt)
    insertion_times.append(it)

    print(f"Run {r+1}/{runs}: bubble={bt:.6f}s, insertion={it:.6f}s")

avg_bubble = sum(bubble_times) / runs
avg_insertion = sum(insertion_times) / runs

print("\nAverages:")
print(f"Bubble Sort : {avg_bubble:.6f} seconds (avg over {runs} runs)")
print(f"Insertion Sort: {avg_insertion:.6f} seconds (avg over {runs} runs)")

faster = "bubble sort" if avg_bubble < avg_insertion else "insertion sort"
print(f"\nConclusion: On this data and environment, {faster} is faster.")

if __name__ == "__main__":
    main()

```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\Namitha\OneDrive\Desktop\AIAC\LabTest-3> & C:/Users/Namitha/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Namitha/OneDrive/Desktop/AIAC/LabTest-3/Task1.py
Run 1/5: bubble=0.065898s, insertion=0.029053s
Run 2/5: bubble=0.061754s, insertion=0.029041s
Run 3/5: bubble=0.064743s, insertion=0.032937s
Run 4/5: bubble=0.080729s, insertion=0.033287s
Run 5/5: bubble=0.067744s, insertion=0.029900s

Averages:
Bubble Sort : 0.068174 seconds (avg over 5 runs)
Insertion Sort: 0.030844 seconds (avg over 5 runs)

Conclusion: On this data and environment, insertion sort is faster.
○ PS C:\Users\Namitha\OneDrive\Desktop\AIAC\LabTest-3>

```

Task-2:

Code:

```
task2.html > head
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width,initial-scale=1" />
6      <title>Dynamic Product List</title>
7      <style>
8          :root{font-family:system-ui,-apple-system,Segoe UI,Roboto,Arial;margin:0;padding:0}
9          body{padding:20px;background:#f5f7fb;color:#111}
10         header{display:flex;justify-content:space-between;align-items:center;margin-bottom:16px}
11         .products{display:grid;grid-template-columns:repeat(auto-fill,minmax(220px,1fr));gap:16px}
12         .card{background:#fff;border-radius:8px;padding:12px;box-shadow:0 1px 4px rgba(0,0,0,.08);display:flex}
13         .name{font-weight:600}
14         .price{color:#0a7a5f;font-weight:700}
15         button{padding:8px 10px;border:0;border-radius:6px;background:#0078d4;color:#fff;cursor:pointer}
16         button[disabled]{opacity:.6;cursor:default}
17         .cart{min-width:260px;border-left:1px solid #e6eef6;padding-left:16px}
18         .layout{display:flex;gap:20px;align-items:flex-start}
19         .cart-list{list-style:none;padding:0;margin:8px 0}
20         .cart-item{display:flex;justify-content:space-between;padding:6px 0;border-bottom:1px dashed #e9eef3}
21         .muted{color:#6b778c;font-size:.9rem}
22         .small{font-size:.9rem}
23     </style>
24 </head>
25 <body>
26     <header>
27         <h1>Products</h1>
28         <div aria-live="polite" id="cartStatus" class="muted">Cart: 0 items</div>
29     </header>
30
31     <div class="layout">
32         <main style="flex:1">
33             <section id="productGrid" class="products" aria-label="Product list">
34                 <!-- Products rendered here -->
35             </section>
36         </main>
```

```

8     <aside class="cart" aria-label="Shopping cart" style="width:320px">
9         <h2 class="small">Shopping Cart</h2>
0         <div class="muted small" id="cartTotal">Total: $0.00</div>
1         <ul id="cartList" class="cart-list" aria-live="polite"></ul>
2         <button id="clearCart" style="background:#d83b01; margin-top:8px">Clear Cart</button>
3     </aside>
4 </div>
5
6 <script>
7     // Sample product array
8     const products = [
9         { id: 1, name: 'Wireless Mouse', price: 24.99 },
0         { id: 2, name: 'Mechanical Keyboard', price: 89.5 },
1         { id: 3, name: 'USB-C Hub', price: 34.0 },
2         { id: 4, name: '27" Monitor', price: 219.99 },
3         { id: 5, name: 'Noise-cancelling Headphones', price: 129.0 }
4     ];
5
6     // In-memory cart (array of {id, name, price, qty})
7     const cart = [];
8
9     const productGrid = document.getElementById('productGrid');
0     const cartList = document.getElementById('cartList');
1     const cartStatus = document.getElementById('cartStatus');
2     const cartTotal = document.getElementById('cartTotal');
3     const clearCartBtn = document.getElementById('clearCart');
4
5     const fmt = new Intl.NumberFormat(undefined, { style: 'currency', currency: 'USD' });
6
7     function renderProducts() {
8         productGrid.innerHTML = '';
9         products.forEach(p => {
        ...
    });
}

```

```

const card = document.createElement('article');
card.className = 'card';
card.innerHTML =
    <div class="name">${escapeHtml(p.name)}</div>
    <div class="price">${fmt.format(p.price)}</div>
    <div style="margin-top:auto">
        <button data-id="${p.id}" aria-label="Add ${escapeHtml(p.name)} to cart">Add to Cart</button>
    </div>
`;
productGrid.appendChild(card);
});

function escapeHtml(str) {
    return String(str).replace(/[&<>'"]/g, s => ({
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#39;'
    })[s]);
}

function addToCart(productId) {
    const prod = products.find(p => p.id === productId);
    if (!prod) return;
    const existing = cart.find(i => i.id === productId);
    if (existing) {
        existing.qty += 1;
    } else {
        cart.push({ id: prod.id, name: prod.name, price: prod.price, qty: 1 });
    }
    renderCart();
}

```

```

        function removeFromCart(productId) {
            const idx = cart.findIndex(i => i.id === productId);
            if (idx === -1) return;
            cart.splice(idx, 1);
            renderCart();
        }

        function renderCart() {
            cartList.innerHTML = '';
            let total = 0;
            let items = 0;
            cart.forEach(item => {
                total += item.price * item.qty;
                items += item.qty;
                const li = document.createElement('li');
                li.className = 'cart-item';
                li.innerHTML =
                    `<div>
                     <div style="font-weight:600">${escapeHtml(item.name)}</div>
                     <div class="muted small">${item.qty} ${fmt.format(item.price)}</div>
                   </div>
                   <div style="text-align:right">
                     <div>${fmt.format(item.price * item.qty)}</div>
                     <button data-remove="${item.id}" style="margin-top:6px;background:#e11; padding:6px; border:1px solid black;">Remove
                   </div>
                `;
                cartList.appendChild(li);
            });
            cartStatus.textContent = `Cart: ${items} item${items !== 1 ? 's' : ''}`;
            cartTotal.textContent = `Total: ${fmt.format(total)}`;
        }
    }
}

```

```

0 // Event delegation for Add to Cart and Remove
1 productGrid.addEventListener('click', (e) => {
2     const btn = e.target.closest('button[data-id]');
3     if (!btn) return;
4     const id = Number(btn.getAttribute('data-id'));
5     addToCart(id);
6 });
7
8
9 cartList.addEventListener('click', (e) => {
0     const btn = e.target.closest('button[data-remove]');
1     if (!btn) return;
2     const id = Number(btn.getAttribute('data-remove'));
3     removeFromCart(id);
4 });
5
6 clearCartBtn.addEventListener('click', () => {
7     cart.length = 0;
8     renderCart();
9 });
0
1 // Initial render
2 renderProducts();
3 renderCart();
4 </script>
5 </body>
6 </html>

```

Output:

Products

Wireless Mouse
\$24.99

[Add to Cart](#)

Mechanical Keyboard
\$89.50

[Add to Cart](#)

USB-C Hub
\$34.00

[Add to Cart](#)

27" Monitor
\$219.99

[Add to Cart](#)

Shopping Cart
Total: \$0.00

[Clear Cart](#)

Cart: 0 items