Department of Computer Science – L2

Module Examination :Algorithms and Data Structures 3

Exam Correction (El-Oued, 19/01/2026)

Exercise 1 — Stack to Queue (08 points)

```c
/* Transfer Stack S to Queue Q */
QueueNode* stackToQueue(StackNode *S) {
    StackNode *T = NULL;
    QueueNode *Q = NULL;
    int x;

    /* Reverse S into T */
    while (!isEmptyStack(S)) {
        S = pop(S, &x);
        T = push(T, x);
    }

    /* Transfer T into Q */
    while (!isEmptyStack(T)) {
        T = pop(T, &x);
        Q = enqueue(Q, x);
    }

    return Q;
}
```

Exercise 2 — Time and Space Complexity (04 points)

Time Complexity:

Let n be the length of the string. At each recursive call, a constant number of operations is performed (comparison and swap). The function reduces the problem size by two characters at each call.

The recurrence relation is:

$$T(n) = T(n-2) + O(1)$$

Solving this recurrence gives:

$$T(n) = O(n)$$

Space Complexity:

The function does not use any additional data structures. However, due to recursion, each function call occupies space on the call stack.

The maximum depth of recursion is $\frac{n}{2}$, therefore the space complexity is:

$$O(n)$$

Conclusion:

- Time complexity: $O(n)$
- Space complexity: $O(n)$

Exercise 3 — Singly Circular Linked List (08 points)

```c
/* Delete the last node of a singly circular linked list */
Node* deleteLast(Node *head) {
    Node *p, *prev;

    /* Empty list */
    if (head == NULL)
        return NULL;

    /* List with only one node */
    if (head->next == head) {
        free(head);
        return NULL;
    }

    /* Find the last node and its predecessor */
    p = head;
    while (p->next != head) {
        prev = p;
        p = p->next;
    }

    /* Remove the last node */
    prev->next = head;
    free(p);

    return head;
}
```