National University of Computer & Emerging Sciences, Karachi
School of Computing (BSCS)
Quiz 01 (Fall 2024)
Date: 05th September, 24 Time: 12:00pm –1:00 pm

Course Code: CS2001	Course Name: Data Structures & Algorithms
Instructor Name: Ms. Ayesha Ali	
Student Roll No:	Section No:

MCQS (5 MARKS)

1. What is the primary reason for using Abstract Data Types?
    a) To write more code.
    b) To improve the performance of algorithms.
    c) To hide implementation details and expose only the necessary operations.
    d) To simplify data storage.
2. What type of memory allocation is used in dynamic arrays?
    a) Stack allocation
    b) Heap allocation
    c) Register allocation
    d) Static allocation
3. Which of the following statements is true about memory allocation in a singly linked list?
    a) Memory is allocated contiguously.
    b) Memory is allocated dynamically as nodes are added.
    c) Memory is allocated at the beginning for the entire list.
    d) Memory is pre-allocated for all nodes.
4. In a doubly linked list, what is the primary advantage of having both next and prev pointers in each node?
    a) It allows for random access to elements by index.
    b) It makes reversing the list more efficient.
    c) It reduces memory usage.
    d) It simplifies the sorting of the list.
5. Which of the following is true about pointer arithmetic?
    a) Adding 1 to a pointer increases its value by 1.
    b) Subtracting 1 from a pointer decreases its value by 1.

c) Adding 1 to a pointer advances it to the next memory location of its type.
d) Subtracting 1 from a pointer moves it to the previous byte.

SHORT Q/A (5 MARKS, 2.5 EACH)

Question 01

Explain why a doubly linked list uses more memory per node than a singly linked list.

Question 02

Explain how do you insert a node in a doubly linked list at position n?

CODING (5 MARKS, 2.5 EACH)

Question 01

Write the code of question 02 above.

Question 02

The following code attempts to delete the nth node in a singly linked list. Find the error and correct it.

```
void deleteAtPosition(Node*& head, int position) {
    if (head == NULL) return;

    Node* temp = head;

    if (position == 0) {
        head = head->next;
        delete head;
        return;
    }

    for (int i = 0; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) return;

    Node* toDelete = temp->next;
    temp->next = temp->next->next;
    delete toDelete;
}
```