**COE 428 QUIZ 4**

**STUDENT NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**STUDENT USERID (LOGIN):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**STUDENT NUMBER: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1) What does the following function do for a given Linked List with first node as head?

**void fun1(struct node\* head)**

**{**

**if(head == NULL) return;**

**fun1(head->next);**

**printf("%d  ", head->data);**

**}**

1. **Prints all nodes of linked lists**
2. **Prints all nodes of linked list in reverse order**
3. **Prints alternate nodes of Linked List**
4. **Prints alternate nodes in reverse order**

2) Which of the following points is/are true about Linked List data structure when it is **compared** with array:

1. **Arrays have better cache locality that can make them better in terms of performance.**
2. **It is easy to insert and delete elements in Linked List.**
3. **Random access is not allowed in a typical implementation of Linked Lists**
4. **The size of array has to be pre-decided, linked lists can change their size any time.**
5. **All of the above**

3) What is the output of following function for start pointing to first node of following linked list?

**1 ->2 -> 3 -> 4 -> 5 -> 6 ->NULL**

**void fun(struct node\* start)**

**{**

**if(start == NULL)**

**return;**

**printf("%d ", start->data);**

**if(start->next != NULL )**

**fun(start->next->next);**

**printf("%d ", start->data);**

**}**

1. **1 4 6 6 4 1**
2. **1 3 5 1 3 5**
3. **1 2 3 5**
4. **1 3 5 5 3 1**

4) What type of **linked list** was implemented as part of your Lab4 submission?

1. **Doubly linked list**
2. **Circular Linked list**
3. **Stack based Linked list**
4. **None of the above.**

5) The complexity (as a function of *n*, the number of items in the list) for the operations **add and delete** in your Lab4 submission is:

1. **O(1)**
2. **O(n)**
3. **It depends on size of each element added or deleted.**
4. **None of the above.**

6) **struct item**

**{**

**int data;**

**struct item \* next;**

**};**

**int f(struct item \*p)**

**{**

**return (**

**(p == NULL) ||**

**(p->next == NULL) ||**

**(( P->data <= p->next->data) && f(p->next))**

**); }**

For a given **linked list p above**, the function **f** returns **1** if and only if

1. **the list is empty or has exactly one element**
2. **the elements in the list are sorted in non-decreasing order of data value**
3. **the elements in the list are sorted in non-increasing order of data value**
4. **not all elements in the list have the same data value.**

7) Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is **not given**, can we **delete** the node X from given linked list?

1. **Possible if X is not last node. Use following two steps (i) Copy the data of next of X to X. (ii) Delete next of X.**
2. **Possible if size of linked list is even.**
3. **Possible if size of linked list is odd**
4. **Possible if X is not first node. Use following two steps (i) Copy the data of next of X to X. (ii) Delete next of X.**

8) Consider the following function to traverse a linked list.

**void traverse(struct Node \*head)**

**{**

**while (head->next != NULL)**

**{**

**printf("%d ", head->data);**

**head = head->next;**

**}**

**}**

Which of the following is **FALSE** about above function?

1. **The function may crash when the linked list is empty**
2. **The function doesn't print the last node when the linked list is not empty**
3. **The function is implemented incorrectly because it changes head**
4. **None of the above.**

9) The **print** operation as part of lab4 linked list implementation has a complexity of:

1. **O(n)**
2. **O(1)**
3. **O(logn)**
4. **None of the above**

10) Can a Singly linked list be implemented without using a head node?

1. **Yes**
2. **No**
3. **Depends on the length of the list**
4. **Only if last element is Null.**