**Student Id: (Solution) Section: BCS-3G Marks: 15**

**Quiz#01 - Paper A**

**Question#01**

1. You are given the head of a linked list, and an integer k.

Return the head of the linked list after swapping the values of the kth node from the beginning (the list is 1-indexed).

|  |
| --- |
| Input:  **Head: 1 -> 2 -> 3 -> 4 -> 5 and k: 2**  Output:  **Head: 1 -> 3 -> 2 -> 4 -> 5** |

|  |
| --- |
| Node\* swapNodeK(Node\* head, int k) {  if (!head || !head->next) return head;  Node\* tempNode = head;  Node\* k\_node = nullptr;  int count = 1;  while (tempNode && count < k) {  tempNode = tempNode->next;  count++;  }  if (tempNode == nullptr || tempNode->next == nullptr) return head;  k\_node = tempNode;  Node\* nextNode = tempNode->next;  // Swap  int temp\_val = k\_node->val;  k\_node->val = nextNode->val;  nextNode->val = temp\_val;  return head;  } |

1. You are given the head of a singly linked list. Write a function to delete all nodes that contain even numbers. After deleting these nodes, the function should return the head of the updated linked list.

|  |
| --- |
| Input:  **Head: 1 -> 2 -> 3 -> 4 -> 5 -> 6**  Output:  **Head: 1 -> 3 -> 5** |

|  |
| --- |
| Node\* deleteEvenNodes(Node\* head) {  Node\* tempHead = new Node(0);  tempHead->next = head;  Node\* current = tempHead;  while (current->next) {  if (current->next->val % 2 == 0) {  Node\* deletedNode = current->next;  current->next = deletedNode->next;  delete deletedNode;  } else {  current = current->next;  }  }  Node\* newHead = tempHead->next;  delete tempHead;  return newHead;  } |

**Question#02**

Consider the content of an integer array as below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| 23 | 4 | 9 | 3 | 11 | 10 | 2 | 1 | 12 | 20 |

How the content of the array will look like after the fourth iteration of the outer loop in the following cases:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bubble Sort | | | | | | | | | |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| 3 | 4 | 2 | 1 | 9 | 10 | 11 | 12 | 20 | 23 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selection Sort | | | | | | | | | |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| 1 | 2 | 3 | 4 | 11 | 10 | 9 | 23 | 12 | 20 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Insertion Sort | | | | | | | | | |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| 3 | 4 | 9 | 11 | 23 | 10 | 2 | 1 | 12 | 20 |

**Question#03**

You are given a singly linked list with the following values:

10 -> 7 -> 5 -> 8 -> 2 -> 6 -> 3 -> 4

Answer the following questions:

1. Identify the sorting algorithm being used to sort this singly linked list.

Insertion sort

1. ~~Complete the sorting of the remaining elements using the identified sorting algorithm. (excluded)~~
2. Write the time complexity of the sorting algorithm.

O(n2)

Dry Run:

|  |
| --- |
| **Initial List:**  10 -> 7 -> 5 -> 8 -> 2 -> 6 -> 3 -> 4  **List after first iteration:**  7 -> 10 -> 5 -> 8 -> 2 -> 6 -> 3 -> 4  **List after second iteration:**  5 -> 7 -> 10 -> 8 -> 2 -> 6 -> 3 -> 4 |