Tutorial 2 Solutions– Linked List

1. **(getListLen)** One sample solution is provided below:

```
int getListLen(const ListNode* head) {
    int count = 0;
    const ListNode* current = head; // Pointer to traverse the list

     while (current != nullptr) {
         count++;             // Increment count for each node
         current = current->next; // Move to the next node
     }

    return count; // Return total number of nodes
}
```

2. **(stringArray2List)** One sample solution is provided below:

```
void arrayToLinkedList(const string* arr, int size, StringNode*& head) {
    if (size == 0) {
        head = nullptr;
        return;
    }

    // Initialize head with the first element
    head = new StringNode;
    head->name = arr[0];
    head->next = nullptr;  // The first node is special, as it points to nullptr

    // Insert the rest of the elements behind the head
    for (int i = 1; i < size; ++i) {
        StringNode* newNode = new StringNode;
        newNode->name = arr[i];
        newNode->next = head; // New node points to the current head
        head = newNode;       // Head now points to the new node
    }
}
```

3. **(insert2SortedList)** One sample solution is provided below:

```
void insertNode2SortedList(Node*& head, double number)
{
    Node *nodePtr, *previousNodePtr;

     Node* newNode = new Node; // new node
     newNode->value = number;
     newNode->next = nullptr;
    if (head == nullptr || head->value >= number) // A new node goes at the
beginning of the list.
    {
        nodePtr = head;
        head = newNode;
        newNode->next = nodePtr;
    } else
    {
        previousNodePtr = head;
        nodePtr = head->next;
        // Find the insertion point
        while (nodePtr != nullptr && nodePtr->value < number)
        {
            previousNodePtr = nodePtr;
            nodePtr = nodePtr->next;
        }
        // Insert the new node just before nodePtr.
        previousNodePtr->next = newNode;
        newNode->next = nodePtr;
    }
}
```

4. **(concatenateTwoLists)** One sample solution is provided below:

```
void concateTwoLists(Node*& firstList, Node*& secondList) {
    if (firstList == nullptr) {
        // If firstList is empty, point it to secondList
        firstList = secondList;
    } else {
        // Traverse to the end of firstList
        Node* temp = firstList;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        // Append secondList to the end of firstList
        temp->next = secondList;
    }

    // Set secondList to nullptr to avoid dangling pointers
    secondList = nullptr;
}
```