CS 300 Data Structures
Problem Set #9 – Linked List cont.

1. Insertion at the tail requires a full traversal of this type of linked list:
   a. Linear, singly linked list, only a head pointer
   b. Linear, doubly linked list, head and tail pointers
   c. Circular, doubly linked list, only a head pointer
   d. a and c
   e. a, b and c
2. List two possible reasons to use a linked list instead of an array.

   Size of data is not known a priori.
   Some operations on the list are more efficient (eg. sorted insertion)

3. To store a fixed number of integers, which always uses more memory: a singly linked list or an array? Why?

   Singly linked list, because extra space is required for the pointers in each node.

4. Write the constructor for a typical singly linked list with head and tail pointers.

```
LinkedList::LinkedList ()
{
   head = NULL;
   tail = NULL;
}
```

5. Write a non-recursive function, countNodes, which counts and returns the number of nodes in a singly linked list.

```
struct Node
{
   int data;
   Node *next;
};

int countNodes ( Node* start )
{
   int count = 0;
   while (start != NULL)
   {
      count++;
      start = start->next;
   }
   return count;
}
```

6. Now rewrite the same function, countNodes, using recursion.

```
int countNodes ( Node *start )
{
   if (start == NULL)
      return 0;
   else
      return 1 + countNodes (start->Next);
}
```

CS 300 Data Structures

Problem Set #9 – Linked List cont.