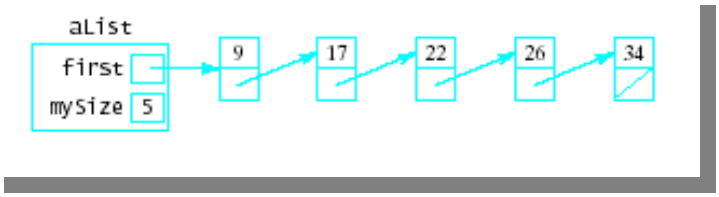
CST 370
Homework (Linked Lists)

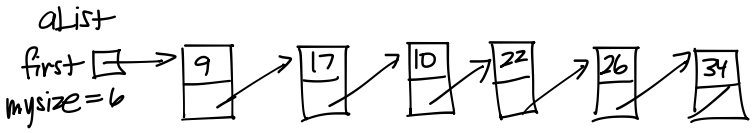
1. Suppose that you are given a linked list as shown below. You can read about **linked lists** from section 6.4 and 6.5 of the book. Source code describing the operation of a Linked List is available on iLearn (LinkedList.h, LinkedList.cpp and Sample_LinkedList_Tester.cpp). **(30 points)**

Assume that there is a new function "insertnew" (as defined below) to add a node in the linked list. Read the function very carefully.

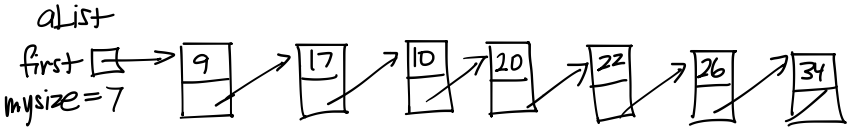
```
void List::insertnew(ElementType dataVal, int index)
{
      mySize++;
      Node * newPtr = new Node(dataVal);
      Node * predPtr = first;

      for(int i = 0; i <= index; i++)
    {
            predPtr = predPtr->next;
    }
      newPtr->next = predPtr->next;
      predPtr->next = newPtr;
}
```



(a) Draw the updated linked list after the execution of **aList.insertnew(10, 0)**.



(b) From the result of the above question (a), draw the updated linked list after the execution of **aList.insertnew(20, 1)**.



2. Write an algorithm (in Pseudocode) to search a linked list (first node pointed to by "*first*") for a given item "*item*". If the item is found, return a pointer to the predecessor of the node containing that item. Otherwise, return Null. (**30 points**)

```
searchList (item)

//Input: item that is being searched for
//Output: returns predecessor of the node containing the item being searched

   ptr = first
  prevPtr = null

  while (ptr != null)
        if (ptr.data == item)
           return prevPtr
     prevPtr = pointer
     ptr = pointer.next

  return null
```

3. Consider a linked list (first node pointed to by "*first*"). Write an algorithm (in Pseudocode) to find the mean of the values in a linked list. If the list is empty, return a value of 0. **(40 points).**

```
listMean ()

//Input: none
//Output: mean values of all linked list values or 0 if list is empty

  ptr = first

  if (ptr == null)
     return 0
   else
      while (ptr != null)
        count++
        sum += ptr.data
        ptr = ptr.next
   mean = sum / count
   return mean
```