```java
public class Q4{

    /*
     * For this question I implemented my own version of a linked list that could be used for this problem.
     *
     * Colin Gallacher
     **/


    /*
     * Inner node class
     * Contains the methods and fields required by the linked list in this problem
     */
    class Node{

        public int value;
        public Node next;

        public Node(){
            value = 0;
            next = null;

        }

        public Node(int value_in, Node link){
            value = value_in;
            next = link;


        }


        public void setNext(Node next_link){

            next = next_link;
        }

        public Node getNext(){

            return next;
        }

        public void setData(int data_in){

            value = data_in;

        }

        public int getData(){

            return value;
        }


    }

    /*
     * Inner singly linked list class implements the methods associated with the singly linked list ADT
     */
    class SinglyLinkedList{

        protected Node start;
        protected Node end;
        protected int size;

        public SinglyLinkedList(){
            start = null;
```

```
68              end = null;
69              size = 0;
70
71          }
72
73          public boolean add(int value, int position){
74
75              Node iter = start;
76              Node newNode = new Node(value,null);
77
78
79           if (size == 0){
80
81               start = newNode;
82               newNode.setNext(end);
83               size++;
84
85           }
86           else if (position == 0){
87
88               newNode.setNext(start);
89               start = newNode;
90               size++;
91
92           }
93
94
95
96           else if (position <= size && position >= 0){
97
98                  for(int i =1; i<=position; i++){
99
100
101                     if(i == position){
102
103
104                         Node temp = iter.getNext();
105                         iter.setNext(newNode);
106                         newNode.setNext(temp);
107
108                     }
109
110                     iter=iter.getNext();
111
112                 }
113                 size++;
114                 return true;
115         }
116
117             return false;
118         }
119
120         public boolean addToStart(int value){
121
122
123             return add(value,0);
124
125         }
126
127         public boolean addToEnd(int value){
128
129             return add(value, size);
130         }
131
132         public void printList(){
133
134             Node temp = start;
135             System.out.println("Size: " + size);
136
```

```
137             System.out.print("[ ");
138             for(int i = 0; i<size; i++){
139 
140             System.out.print(temp.value + "  ");
141             temp=temp.getNext();
142 
143             }
144 
145             System.out.println(" ]");
146 
147         }
148 
149 
150 
151         /*
152          * This is the iterative call to count the size of the singly linked list
153          *
154          */
155 
156         public int CountIterative(){
157 
158             int count=0;
159 
160             Node temp = start;
161 
162             while(temp!=null){
163 
164                 temp = temp.getNext();
165                 count++;
166 
167 
168             }
169 
170 
171             return count;
172 
173         }
174 
175         /*
176          * This is the recursive call to count the size of the singly linked list
177          *
178          */
179 
180         public int CountRecursive(Node n){
181 
182 
183             int count;
184 
185             if(n == null){
186                 return 0;
187 
188             }
189 
190             count =1;
191 
192             if(n.getNext()!= null){
193 
194                  count = CountRecursive(n.getNext())+1;
195             }
196 
197             return count;
198 
199         }
200 
201 
202     }
203 
204 
205 
```

```
206
207
208
209
210
211
212         public static void main(String[] Args){
213
214             Q4 outer = new Q4();
215
216             Q4.SinglyLinkedList SLL =  outer.new SinglyLinkedList();
217
218             //Build a list and test the function methods
219
220             SLL.addToStart(5);
221             SLL.addToStart(4);
222
223             SLL.addToEnd(7);
224             SLL.add(2, 1);
225             SLL.add(7, 2);
226             SLL.add(9, 4);
227             SLL.add(10, 3);
228             SLL.add(4,  5);
229             SLL.add(2, 1);
230             SLL.add(7, 2);
231             SLL.add(9, 4);
232             SLL.add(10, 3);
233             SLL.add(4,  5);
234             SLL.add(4,  5);
235             SLL.add(2, 1);
236             SLL.add(7, 2);
237             SLL.add(9, 4);
238             SLL.add(10, 3);
239             SLL.add(4,  5);
240
241
242             //Print the list
243
244             SLL.printList();
245
246
247             // Question 4 problems.
248
249             int count = SLL.CountIterative();
250
251             System.out.println("Iterative call count: " + count);
252
253             count = SLL.CountRecursive(SLL.start);
254
255             System.out.println("Recursive call count: " + count);
256
257         }
258
259     }
260
261
262     OUTPUT:
263
264
265     Size: 19
266     [ 4  2  7  10  2  4  9  7  10  2  4  4  9  7  10  5  4  9  7  ]
267     Iterative call count: 19
268     Recursive call count: 19
269
270
```