```
 1: program LinkedList;
 2:
 3: uses
 4:   sysUtils;
 5: type
 6:   NodePtr = ^Node;
 7:
 8:   Node = record
 9:     data  : Integer;
10:     next  : NodePtr;
11:   end;
12:
13:   LinkedLst = record
14:     start   : NodePtr;
15:     finish  : NodePtr;
16:   end;
17:
18: procedure CreateList(var list: LinkedLst);
19: begin
20:   list.start := nil;
21:   list.finish := nil;
22: end;
23:
24:
25: // Create Node
26: function CreateNode(data : Integer; next : NodePtr): NodePtr;
27: begin
28:   New(result);
29:   result^.data := data;
30:   result^.next := next;
31: end;
32:
33: // Dispose all nodes
34: procedure DisposeNodes(var start: NodePtr);
35: begin
36:   if start <> nil then
37:   begin
38:   DisposeNodes(start^.next);
39:   Dispose(start);
40:   start := nil;
41:   end;
42: end;
43:
44: // Insert After Function
45: function InsertBefore(data: Integer; n : NodePtr): NodePtr;
46: begin
47:   result := CreateNode(data, n^.next); // follow n
48:   n^.next := result;
49: end;
50:
51: // Insert After Function
52: function InsertAfter(data: Integer; n : NodePtr): NodePtr;
53: begin
54:   result := CreateNode(data, n^.next); // follow n
55:   n^.next := result;
56:  end;
57:
58: // Insert At Start of List
59: function PrependNode(data: Integer; n : NodePtr): NodePtr;
60: begin
61:   result := CreateNode(data, n^.next^.next);
62:   n^.next := result;
63:
64: end;
65:
66: function FindNode(list, value : nodePtr): NodePtr;
67: begin
68:   result := nil;
69: end;
70:
71: // Print Nodes
72: procedure PrintBackTo(n: NodePtr);
73: begin
74:   if (n <> nil) then
75:   begin
76:   PrintBackTo(n^.next);
77:   Write(' <- ', n^.data);
78:   end
79:   else
80:   begin
81:     Write('nil');
82:   end;
83: end;
84:
85: // Print From Node
86: procedure PrintFrom(n: NodePtr);
87: begin
88:   if n <> nil then
89:     begin
90:       Write(n^.data,' -> ');
91:       PrintFrom(n^.next);
92:      end
93:     else
94:     begin
95:       WriteLn('nil');
96:     end;
97:     //WriteLn();
98: end;
99:
100: // Count nodes
101: function Count(n: NodePtr):Integer;
102: var
103:   current: NodePtr;
104: begin
105:   current := n;
106:   result := 0;
107:   while (current <> nil) do
108:   begin
109:     result +=1;
110:     current := current^.next;
111:   end;
112:
113:   // if n <> nil then
114:   // result := 1+Count(n^.next)
115:   // else
116:   //   result := 0;
117: end;
118:
119: procedure Main();
120: var
121:   start: NodePtr;
122:   list : LinkedLst;
123:
124: begin
125:   CreateList(list);
126:   start := CreateNode(1, nil);
```

```
127:   start := CreateNode(2, start);
128:   InsertBefore(8, start);
129:   start := CreateNode(3, start);
130:   InsertBefore(4, start);
131:   // AppendNode(start^.next, 4);
132:   // AppendNode(start^.next^.next, 5);
133:   WriteLn('Count is ', Count(start));
134:
135:   //PrependNode(start^.data, start^.prev);
136:
137:   PrintFrom(start);
138:   PrintBackTo(start);
139:   WriteLn();
140:   DisposeNodes(start);
141:
142:   // start is = nil
143: end;
144:
145: // Main executable
146: begin
147:   Main();
148: end.
```