```
  1: program newList;
  2:
  3: uses
  4:   sysUtils;
  5:
  6: type
  7:   NodePtr = ^Node;
  8:
  9:   Node = record
 10:     name  : String;
 11:     next  : NodePtr;
 12: end;
 13:
 14:
 15: function CreateNode(data : String; next : NodePtr): NodePtr;
 16: begin
 17:   //alocate space on the heap
 18:   New(result);
 19:
 20:   //set up node details
 21:   result^.name := data;
 22:   result^.next := next;
 23: end;
 24:
 25: function FindNode(list: NodePtr; value: String): NodePtr;
 26: var
 27:   current : NodePtr;
 28: begin
 29:   current := list;
 30:   while (Assigned(current)) AND (current^.name <> value) do
 31:   begin
 32:     current := current^.next;
 33:     result := current;
 34:   end;
 35: end;
 36:
 37: //Function replaces node data
 38: function ReplaceNodeData(list: NodePtr; value: String): NodePtr;
 39: var
 40:   current : NodePtr;
 41: begin
 42:   current := list;
 43:   while (Assigned(current)) AND (current^.name = value) do
 44:   begin
 45:
 46:     result := current;
 47:   end;
 48: end;
 49:
 50: procedure PrependNode(list: NodePtr; value: String);
 51:
 52: begin
 53:
 54:   while (Assigned(list)) AND (list^.name = value) do
 55:   begin
 56:
 57:    CreateNode('Andrew', list^.next);
 58:
 59:   end;
 60: end;
 61:
 62: // Insert After Function
 63: function InsertAfter(data: String; n : NodePtr): NodePtr;
 64: begin
 65:   result := CreateNode(data, n^.next); // follow n
 66:   n^.next := result;
 67: end;
 68:
 69: function FindPrevious(list: NodePtr; value: String):NodePtr;
 70: var
 71:   previous, current : NodePtr;
 72: begin
 73:   current := list;
 74:   while (Assigned(current)) AND (current^.name <> value) do
 75:   begin
 76:     previous := current;
 77:     WriteLn('Previous Name ',previous^.name);
 78:     current := current^.next;
 79:   end;
 80:   result := previous;
 81: end;
 82:
 83: function InsertBefore(data, value : String; n : NodePtr): NodePtr;
 84: var
 85:   previous,current : NodePtr;
 86: begin
 87:   previous := FindPrevious(n,value);
 88:   WriteLn('InsertBefore Previous Name is ',previous^.name);
 89:   current := CreateNode(data,previous);
 90:   current^.next := previous;
 91:
 92:   WriteLn('Current Name is ',current^.name);
 93:
 94:
 95:   result := current;
 96: end;
 97:
 98: procedure PrintFrom(n: NodePtr);
 99: begin
100:   if n <> nil then
101:     begin
102:       Write(n^.name,' -> ');
103:       PrintFrom(n^.next);
104:      end
105:     else
106:     begin
107:       WriteLn('nil');
108:     end;
109:     //WriteLn();
110: end;
111:
112:
113: procedure Main();
114: var
115:   start: NodePtr;
116:
117: begin
118:   //create the node List Record
119:
120:
121:   start := CreateNode('Wayne', nil);
122:   PrintFrom(start);
123:   ReplaceNodeData(start,'Wayne');
124:   PrintFrom(start);
125:   InsertAfter('Andrew',start);
126:   PrintFrom(start);
```

```
127:   InsertBefore('NewName','Andrew',start);
128:   PrintFrom(start);
129: end;
130:
131: begin
132:   Main();
133: end.
```