```
unit biListUnit;
interface uses sysUtils;
    type BiListNodeType = record next, last : ^BiListNodeType;
                                 value      : Pointer;
                                 valueType  : String;
                          end;
    type BiListType = record first, last : ^BiListNodeType; end;
    type BlockType = function(val : BiListNodeType) : Boolean;
    function init(var list : BiListType) : BiListType; overload;
    function last(const list : BiListType) : BiListNodeType; overload;
    function add(var list : BiListType;
                     node : BiListNodeType
                 )        : BiListType; overload;
    function unshift(var list : BiListType;
                         node : BiListNodeType
                     )        : BiListType; overload;
    function get(var list : BiListType; which : Integer) : BiListNodeType;
overload;
    function get(var node           : BiListNodeType;
                     counter, which : Integer
                 )                  : BiListNodeType; overload;
implementation
    function init(var list : BiListType) : BiListType; overload;
        begin list.first := nil; list.last  := nil; init := list; end;

    function last(const node : BiListNodeType) : BiListNodeType; overload;
        begin
            if Pointer(node.next) = nil
            then last := node
            else last := last(node.next^);
        end;
    function add(
            var list : BiListType;
            node : BiListNodeType
        ) : BiListType; overload;
        var last : BiListNodeType;
        begin
            if Pointer(list.first) = nil
            then begin
                list.first := @node;
                list.last  := @node;
            end
            else begin
                last := list.last^;
                last.next := @node;
                node.last := @last;
                list.last := @node;
            end;
            add := list;
        end;

    function unshift(var list : BiListType;
                         node : BiListNodeType
                     )        : BiListType; overload;
        var first : BiListNodeType;
        begin
            if Pointer(list.first) = nil
            then begin
```

```
                list.first := @node;
                list.last  := @node;
            end
            else begin
                first := list.first^;
                first.last := @node;
                node.next  := @first;
                list.first := @node;
            end;
            unshift := list;
        end;
    function get(var node           : BiListNodeType;
                     counter, which : Integer
                )                   : BiListNodeType; overload;
        begin
            if counter = which
            then get := node
            else
            if Pointer(node.next) = nil
            then writeln('No such node!')
            else get := get(node.next^, counter + 1, which);
        end;
    function get(var list : BiListType; which : Integer) : BiListNodeType;
overload;
        begin
            get := get(list.first^, 0, which);
        end;
    function insert(var list      : BiListType;
                        node      : BiListNodeType;
                        afterWhat : Integer
                   )              : BiListType; overload;
        var nodeBefore, nodeAfter : BiListNodeType;
        begin
            nodeBefore := get(list, afterWhat);
            nodeAfter  := nodeBefore.next^;
            nodeBefore.next := @node;
            nodeAfter.last  := @node;
            node.last := @nodeBefore;
            node.next := @nodeAfter;
            insert := list;
        end;
    function findBy(block : BlockType; node : BiListNodeType) : BiListNodeType;
overload;
        begin
            if block(node)
            then findBy := node
            else
            if Pointer(node.next) = nil
            then writeln('Not found')
            else findBy := findBy(block, node.next^);
        end;
    function findBy(block : BlockType; var list : BiListType) : BiListNodeType;
overload;
        begin
            findBy := findBy(block, list.first^);
        end;
end.
```