Basic Operations on General Purpose Lists

CREATE: Create and return an empty list.

DESTROY($L$): Reclaim any remaining memory used by list $L$ (assumes that all of its vertices have already been deleted).

EMPTY($L$): Return $true$ if list $L$ is empty, $false$ otherwise.

SIZE($L$): Return the current number of vertices in list $L$.

FIRST($L$): Return the first vertex of list $L$ ($nil$ if $L$ is empty).

LAST($L$): Return the last vertex of list $L$ ($nil$ if $L$ is empty).

DATA($v,L$): Return the data stored at vertex $v$ in list $L$.

NEXT($v,L$): Return the vertex that follows vertex $v$ in list $L$ (or $nil$ if $v$ is the last vertex in $L$); NEXT($nil$) is defined to be $nil$.

PREV($v,L$): Return the vertex that precedes vertex $v$ in list $L$ (or $nil$ if $v$ is the first vertex in $L$); PREV($nil$) is defined to be $nil$. 

- 1 -
(basic list operations continued)

**INSERT**\((d,v,L)\): Create a new vertex to the right of vertex \(v\) in list \(L\) and initialize it to contain data \(d\) (if \(v=\textit{nil}\), then insert at the beginning of \(L\)).

**DELETE**\((v,L)\): Delete \(v\) from \(L\), reclaim memory, and return the data it contained.

**SPLICE**\((L_1,v,L_2)\): Make the vertices of \(L_1\) follow \(v\) in \(L_2\) (if \(v=\textit{nil}\) then insert \(L_1\) at the beginning of \(L_2\)).

**CUT**\((v_1,v_2,L)\): Remove and return as a new list the portion of list \(L\) that starts with \(v_1\) and ends with \(v_2\).

**Note:** SPLICE and CUT could be implemented as sequences of INSERT or DELETE operations but we shall view them as "basic" operations because in many implementations they can be performed in constant time.
(basic list operations continued)

**Note:**

When $L$ is understood, we may omit it as an argument, (or in an object oriented programming language, the syntax might be something like L.FIRST, L.LAST, etc.)
Example - Output a list $L$

procedure PRINT($L$)
  $v$ := FIRST($L$)
  while $v \neq nil$ do begin
    output DATA($v$)
    $v$ := NEXT($v$)
  end
end
Example - Make a copy of a list $L$

function COPY($L$)

$M := CREATE$

$v := FIRST(L)$

while $v \neq nil$ do begin

\text{INSERT}(\text{DATA}(v, L), \text{LAST}(M), M)$

$v := \text{NEXT}(v, L)$

end

return $M$

end
Example - Search for a data item \( d \) in a list \( L \)
(return \( nil \) if \( d \) is not in \( L \))

```
function SEARCH\( (d,L) \)
  \( v := \) FIRST\( (L) \)
  \( \textbf{while} \ v \neq nil \textbf{ and } DATA(v) \neq d \textbf{ do} \)
    \( v := \) NEXT\( (v) \)
  \( \textbf{return} \ v \)
end
```
Example - Add up the data in vertices $v$ through $w$ of $L$
(assumes $L$ stores numbers)

function ADD($v, w, L$)

$sum := DATA(w)$

while $v \neq w$ do begin

$sum := sum + DATA(v)$

$v := NEXT(v)$

end

end
Example, Stack Operations:
  PUSH\((d, S)\) is another name for INSERT\((d, nil, S)\)
  POP\((S)\) is another name for DELETE\((\text{FIRST}\(S), S)\) 

Example, Queue Operations:
  ENQUEUE\((d, Q)\) is another name for INSERT\((d, \text{LAST}(Q), Q)\)
  DEQUEUE\((Q)\) is another name for DELETE\((\text{FIRST}(Q), Q)\)