

Program 3

CS 141
Spring 1992
Craig A. Rich

The following are an Ada package specification for an *implementation* of the list ADT and an Ada procedure which uses the package:

`list_implementation_.ada`

```
generic
  type ELEMENT is private;
  CAPACITY: POSITIVE := 80; (omitted in some implementations)

package LIST_implementation is

  type POSITION is private;
  type LIST is private;

  function EMPTY (L: LIST) return BOOLEAN;
  function LENGTH (L: LIST) return NATURAL;
  function FIRST (L: LIST) return POSITION;
  function EOL (L: LIST) return POSITION;
  function NEXT (L: LIST; p: POSITION) return POSITION;
  function PREVIOUS (L: LIST; p: POSITION) return POSITION;
  function LOCATE (L: LIST; X: ELEMENT) return POSITION;
  function RETRIEVE (L: LIST; p: POSITION) return ELEMENT;
  procedure INSERT (L: in out LIST; p: in out POSITION; X: ELEMENT);
  procedure DELETE (L: in out LIST; p: in out POSITION);

  POSITION_ERROR: EXCEPTION;
  CAPACITY_ERROR: EXCEPTION; (omitted in some implementations)

private
  implementation-dependent full type declarations for POSITION and LIST

end LIST_implementation;
```

`program_3.ada`

```
with TEXT_IO, LIST_implementation;
use TEXT_IO;

procedure PROGRAM_3 is

  package LIST_CHARACTER is new LIST_implementation (CHARACTER);
  use LIST_CHARACTER;

  L: LIST;

  procedure GET (L: in out LIST) is ...
  procedure PURGE (L: in out LIST) is ...
  procedure PUT (L: LIST) is ...

begin
  GET (L);
  PURGE (L);
  PUT (L);
end PROGRAM_3;
```

1. Create package specifications for the following four *implementations* of the list ADT defined in class:

<i>implementation</i>	description
<code>list_bounded_.ada</code>	array-based list plus length component
<code>list_unbounded_single_.ada</code>	pointer-based singly-linked list
<code>list_unbounded_double_.ada</code>	pointer-based circular doubly-linked list
<code>list_bounded_single_.ada</code>	cursor-based singly-linked list

2. Create package bodies for the four *implementations* of the list ADT. Each package body should give full declarations for the ten specified list operations. The operations should execute as efficiently as possible given the physical constraints of the particular *implementation*.
3. Create procedure `program_3.ada` which inputs a line of characters into a list, purges duplicate characters from the list, and outputs a line of characters from the purged list. Operations from a list ADT package can be used as necessary. It does not matter which *implementation* you use, but you should try the procedure using each of the four *implementations* of the list ADT.

- `GET(L)` inputs a line of characters, inserting them successively at the end of L.
- `PURGE(L)` purges duplicate characters from the list L by deleting all occurrences of a character in L except the first occurrence.
- `PUT(L)` outputs a line of characters, retrieving them from successive positions in L.

4. Hand in one contiguous printout containing compiler listings of the following files, in this order:

```
list_bounded_.ada
list_bounded.ada
list_unbounded_single_.ada
list_unbounded_single.ada
list_unbounded_double_.ada
list_unbounded_double.ada
list_bounded_single_.ada
list_bounded_single.ada
program_3.ada
```