$\boxed{\text{Burt Rosenberg}}$

Problem Set 3

Out: 9 September, 1993
Due: 17 September, 1993

Goals

Practice using pointers, linked lists and text files.

Reading Assignment

Read Chapters 11 and 16 from *Oh! Pascal!* by Doug Cooper and pages 15–22 from *Algorithms* by Robert Sedgewick.

Programming Assignment

Write a program which places the words of a text file in a linked list then prints out the list. The text can be read either from the terminal or from a file whose name is requested when the program is run. In the first case, send output to the terminal; in the second case have the program request the name of an output text file.

Place each word separately at the head of the list in the order which the words are read. Print out the list from head to tail. Thus the output text will have the words *in reverse order* from the input text. For instance, input text "what light through yonder window breaks" becomes output text "breaks window yonder through light what."

Have the program free the storage for each element of the linked list before exiting.

Hints

Write subroutines!

1. Write a subroutine which when called gets the next *word* in the input text stream. A *word* will be defined as a sequence of *alphanumerics* terminated by a non-alphanumeric character, i.e. a blank, an eoln, control characters or punctuation. Test this subroutine before getting mixed up in linked lists.

2. If you feel that this program is hard, don't use dummy header nodes. Eventually you must learn this data structure technique, but the exercise at hand does not need it. If you feel confident about the program and want "to do it right", try dummy header nodes.

3. In the case of no dummy headers, write the list manipulation routines:

   ```
   function insertAtHead( s : MyStringType ;
                          p : MyListType ) : MyListType ;
   {   Input:  s a string.
               p a list.
       Output: returns a list with s at the
               head and p afterwards.
   }
   procedure printList( p : MyListType ) ;
   {   Input:  p a list.
       Output: prints all items in the
               list from front to back.
   }
   procedure destroyList( p : MyListType ) ;
   {   Input:  p a list.
       Output: frees all storage on list.
   }
   ```

   Test these separately from the word-reading code until they seem to work fairly well. Then combine all subroutines into the final program.

4. In the case of a dummy head, you will also need the subroutine,

   ```
   function createList : MyListType ;
   { Output: returns a pointer to a newly created
             empty list.
   }
   ```

   With dummy headers, `insertAtHead` does not need to be a function, since the pointer to the list does not change as elements are added.