Burton Rosenberg

Midterm

OCTOBER 16, 1992. 2:00–2:50 PM

There are three problems for a total of 100 points. Show all your work, partial credit will be awarded. Write in the provided blue books, write and sign your name on each one. No notes, no collaboration. Good luck.

Name: ____________________

Problem	Credit
1	
2	
3	
Total	

1. (30 Points.)

   Write a function

   ```
   function eqList( a,b:listPntr ): boolean ;
   ```

   to compare two lists. Let the lists be lists of integers *without* dummy headers and define two lists to be equal if they have the same integers in the same place. (So that the list $(1, 2, 3)$ is *not* equal to the list $(2, 1, 3)$.) Make your code as close to Pascal as you can but don't worry too much about the semicolons. Your type definitions and while-loop test conditions must be stated clearly! Don't forget that `a` or `b` might be empty.

2. (30 Points.)

   Define a data structure that is a linked list and also has an internally memorized “window”. The window can be placed over elements in the list and one can ask for the value of the element under the window.

   In particular: define a data structure so that CreateList returns a pointer of type ListPntr. This pointer can be passed to ResetList, GetList and EndOfList to effectuate the following operations:

```
function CreateList : ListPntr ;
{Returns an empty list with undefined window.}

procedure ResetList( l: ListPntr ) ;
{Brings the window over the first element in the list}

function GetList( l: ListPntr ) : StringType ;
{Returns the element on the list under the window }
{and advances the window so that it is over the next}
{element in the list.}
{If there is no next element, the window is }
{placed in an EndOfList condition }

function EndOfList( l: ListPnter ) : Boolean ;
{Returns true if the window is in an EndOfList condition. }
```

   Write the data type definitions and the code for these subroutines.

3. (40 Points.)

   Show the function `split` on the following page is correct by the use of Loop Invariants. The Precondition, Loop Invariant and Postcondition have been given to you in the program. Please state why the Precondition is initially true, why the Loop Invariant is true each time through the loop and why the Postcondition is true when you exit the loop.

   The goal of this program is to divide the integers in an array into two piles, those smaller or equal to the *pivot element* and those larger than the pivot element. The two piles are themselves in the original array. The pivot element is always at `A[1]`. The function returns a value `split` such that the pile of small numbers is in the array at `A[1],...,A[split-1]` and the pile of big numbers is in the array at `A[split],...,A[N]`. If the pile of big numbers is empty, then `split` will be `N+1`.

```
{Function split, puts the values in an array}
{into big and small piles.}
{Prove it correct by loop invariants!}

const
  N = 25 ;
type
  arrayType = array[1..N] of integer ;

function split(var A : arrayType) : integer ;
var
  i,j, temp : integer ;
begin
  i := 2 ;
  j := 2 ;
  {Precondition: same as Loop Invariant}
  while j<=N do begin
    if A[j]>A[1] then
      j := j + 1
    else begin
      temp := A[i] ;
      A[i] := A[j] ;
      A[j] := temp ;
      i := i + 1 ;
      j := j + 1
    end {else}
    {LOOP INVARIANT: for any integer k<i, A[k] <= A[i] }
    {   and for any integer k, i <= k < j, A[k] > A[i].}
  end {while}
  {Postcondition, A[1] <= A[k] if and only if k < i.}
  split := i ;
end {split}
```