Burton Rosenberg

Final

December 14, 1993. 5:00–7:30 PM

There are six problems each counting equally. Show all your work, partial credit will be awarded.

Name: _____

Problem	Credit
1	
2	
3	
4	
5	
6	
Total	

On my honor, I have neither given nor received aid on this examination-assignment.

Signature: _____

1. Show that these two program fragments are identical. The variables `A`, `B` and `C` are declared as `boolean`, and `S1` represents a statement.

   Program Fragment 1:

   ```
   if A then
   begin
      if B then S1
      else if C then S1
   end ;
   ```

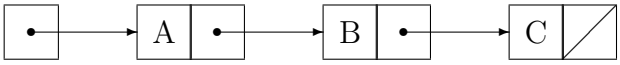
   Program Fragment 2:

   ```
   if ((A AND B) OR (A AND C)) then S1 ;
   ```

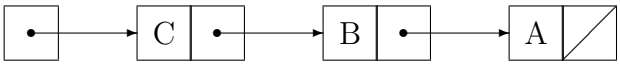
2. Change the following `while` loop into an exactly equivalent program which uses *only* recursion. That is, define a procedure or function which *does not* use any `while` or `repeat` loops (nor any goto's), but which can call itself, and replace the line labeled "Replace Me" with a call to this function or procedure.

```
a[N] := 0 ;
i := 1 ;
while a[i]>0 do i := i + 1 ;  {REPLACE ME}
{Postcondition: i is minimum >= 1
                such that a[i]<=0. }
```

3. Write a function that reverses the order of the elements on a list. For instance, if the list `L` looks like:



then `Rev(L)` should look like:

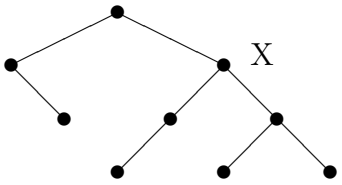


Take as a list definition:

```
Type
    ListPntr = ^ ListRec ;
    ListRec = record
        d : dataType ;
        n : ListPntr ;
        end ;
    List = ListPntr ;
```

HINT: You can do it in a single front to back pass over the list, without the need for `new` or `dispose`. Or you might try recursion, but this is not really simpler.

4. Rotate the following tree at `X`. That is, make node `X` the root via a single rotation.



5. For this problem and the next, let `X[1..N]` be a global array of integers.

   Consider the following procedure which exchanges the largest element among `X[a..b]` with `X[a]`.

```
procedure FindLarge(a,b:integer) ;
var i,j, temp : integer ;
begin
{Prec: 1<=a<=b<=N }
   i := a ;
   j := i ;
{Loop Inv: X[i] largest in X[a..j]}
   while j<=b do begin
      if X[j]>X[i] then i := j ;    { COUNT ME }
      j := j + 1
   end ;
   temp := X[a] ;
   X[a] := X[i] ;
   X[i] := temp
end ;
```

   Calling the procedure twice, we can find the *second largest* in the array `X[1..n]`:

```
function Slow : integer ;
begin
   FindLarge(1,N) ;
   FindLarge(2,N) ;
   Slow := X[2]
end ;
```

   The line "Count Me" in `FindLarge` is run:

$$k_s N + d_s$$

   times during the execution of `Slow`, for some k_s and d_s. *Determine the exact value of* k_s.

6. This problem is a continuation of the previous problem.

   The following program also determines the second largest element in the array `X[1..n]`. The line "Count Me" in the procedure `FindLarge` is run $k_f N + d_f$ times, *total* for all three calls to `FindLarge`, during the execution of `Fast`. *Determine the exact value for* k_f.

```
function Fast : integer ;
var half : integer ;
begin

{Prec: N >= 2 }
{Find the largest in each of the two halves of the array}
   half := (N div 2) + 1 ;
   FindLarge( 1, half-1 ) ;
   FindLarge( half, N ) ;

{Find the second largest in only one half.}
   if X[1]>X[half] then
      FindLarge( 2, half-1 )
   else
      FindLarge( half+1, N ) ;

{Choose the correct among the three.}
   if X[1] > X[half] then
      if X[2]>X[half] then
         Fast := X[2]
      else Fast := X[half]
   else if X[1] < X[half] then
      if X[half+1]>X[1] then
         Fast := X[half+1]
      else Fast := X[1]
   else { X[1]=X[half] }
      Fast := X[1]

end ;
```