Final

19 December, 1991

This is an open book exam. You are to turn in this exam at 5:35 PM.
Spread out! At least two chairs to the next person.

Name: _____

Problem 1: 30 points

Starting with the following type definitions and variable declarations:

```
TYPE
     CPARTS = (Intense, Red, Green, Blue) ;
     COLORS = SET OF CPARTS ;
     STR9 = ARRAY[1..9] OF CHAR ;
     CMAP = RECORD
              Name: STR9;
              Value: COLORS;
            END ;
VAR
     P: CPARTS ;
     C: COLORS ;
     S: STR9 ;
     M: CMAP ;
     B: BOOLEAN ;
     I, J: INTEGER ;
     X, Y: REAL ;
```

Which of the following 30 ``phrases" is a valid Pascal statement. *Circle* your choices.

```
C := Red ;                B := C in M ;
I := ORD(C) ;             I := ORD(B) ;
WRITE(OUTPUT, S[3]) ;     M.Name := S[3] ;
C := C + Intense ;        I := ORD(X) ;
X + Y ;                   M.Value := [ Red ] ;
P := M.Value ;            IF B THEN X := Y ;
X := I/J ;                C := C AND [ Blue ] ;
M.Name := 'C' ;           B := I in C ;
B := B AND NOT B ;        B := I <> J ;
J := ORD(S[7]) ;          C := [ P ] ;
J := SUCC(I) ;            B := B ;
B := EOLN(I);             I := M.Name[7] ;
X := Y * 2.0 ;            X := Ord(S) * 2.0 ;
C.Value := Green ;        B := READLN(INPUT, I);
Y := I ;                  Y := EXP(X) ;
```

Problem 2: 10 points

Suppose the following function

```
TYPE
   Tptr = ^ Tnode ;
   Tnode = RECORD
          Left:  Tptr ;
          Right: Tptr ;
          Value: INTEGER ;
        END ;

FUNCTION TryAgain(N: INTEGER): Tptr;
VAR
   R: Tptr ;
BEGIN
   NEW(R);
   R^.Value := N ;
   IF (N > 100) THEN
     BEGIN
      R^.Left  := TryAgain(N MOD 100);
      R^.Right := TryAgain(N MOD  10);
      R^.Left^.Left := R
     END
   ELSE
     BEGIN
      R^.Left  := NIL ;
      R^.Right := R
     END
END { TryAgain } ;
```

is called with the argument 567 (that is, `TryAgain(567)` is executed). Draw the linked data structure returned by that call.

Use your own paper for the remaining questions.

Problem 3: 10 points

Write a function that converts pounds and ounces into grams. The two arguments, pounds and ounces, to the procedure are passed as Pascal `INTEGER`s and the result is returned as an `INTEGER`. Use the following constant definitions in your program:

```
CONST
     OuncesInPounds = 16;
     GramsInPounds  = 453.592 ;
```

Problem 4: 10 points.

Write a `WHILE` loop *and* associated initialization that reads all the characters of the `INPUT` file and the sets the character variable `C` to the smallest *upper-case letter* in the file. For example, if the `INPUT` file contained:

```
I went to see the doctor of philosophy
With a poster of Rasputin and a beard down to his knee
He never did marry or see a B-grade movie
```

Your code should set `the variable C` to `'B'`.

Problem 5: 10 points

(A) Draw the binary insertion tree that results from inserting, in order, the numbers 3, 2, 5, 6, 1, 4, 7 into an empty tree.

(B) Give *both* the preorder and postorder traversals of your binary insertion tree.

Problem 6: 10 points

Consider a program that is using a hash table to store Social Security numbers. Collisions are resolved using *chaining* as described on pages 492-493 of the text or in the 10 December lecture. The following data structures and hash functions are being used.

```
TYPE
   ChainPtr = ^ChainNode ;
   ChainNode =
     RECORD
        SSNum: INTEGER;
        Next: ChainPtr;
     END ;
   HashTable = ARRAY[0..4] OF ChainPtr;

FUNCTION HashNum(SS: INTEGER): INTEGER;
  BEGIN
   HashNum := SS MOD 5
  END
```

Draw the hash table and chains after the following six numbers are placed into an empty table:

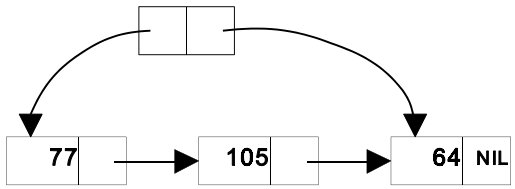
```
111111111 222222222 333333333
555555555 777777777 999999999
```

Problem 7: 10 points

Assume a FIFO queue of integers is represented using a linked list to hold the items of the FIFO and a separate record to hold pointers to the front and back of the FIFO. The following Pascal data structures are used in this implementation of the FIFO.

```
TYPE
     ItemPtr = ^ItemRec ;
     ItemRec = RECORD
                 Value: INTEGER ;
                 Next:  ItemPtr ;
               END ;
     FIFOrec = RECORD
                 Front: ItemPtr ;
                 Back:  ItemPtr ;
               END ;
```

The FIFO queue < 77, 105, 64> will look like the figure on the right when this data representation is used. [77 is the front and 64 the back of the queue.]



For this problem you are to write a Pascal procedure `Qjoin` that takes two FIFOs and sticks the second onto at end of the first *and* makes the second FIFO into an empty FIFO. For example, if the first FIFO is <5, 6> and the second is <7, 8> before `Qjoin` is called, then the first will be <5, 6, 7, 8> and the second will be the empty FIFO < > after the call.

Start with the following header. You *should* be able to write your procedure in the space remaining on this page.

```
PROCEDURE Qjoin(VAR P, Q: FIFOrec);
```