Computing Science 115 B4
FINAL

Date: April 27, 2000

Instructor: John Willson

NAME: _____ **ID:** _____
(Printed)

SIGNATURE: _____

120 minutes

This is a closed book examination. Calculators are fine. Although you should be concise, if you require additional space, then please use the back of the pages. Partial marks may be awarded for work so if you do not know the exact Pascal be sure to put pseudo code.

Question	**Maximum**
One	10
Two	10
Three	10
Four	10
Five	10
Six	10
Seven	10
Eight	10
Nine	10
Ten	10

Total 100

Question One (10) :

a). Given the var declarations

```
A : array [ 1..n, 1..n] of integer;
j,k : integer;
```

where n is a constant, determine what the following statement does. Rewrite the statement to accomplish the same effect in a less tricky way.

```
for j := 1 to n do
  for k := 1 to n do
    A[j,k] := ( j div k) * (k div j);
```

b). In the following program identify at least three errors causing infinite loops that are caused by program style and not by user input.

```
program loop(input,output);
uses wincrt;
{Pre:  None
 Post: The user enters a number and the
program prints out all
      integers from the number down to 0}
var sinking : integer;  {number being reduced}

procedure reduce(sinking : integer);
{Pre:  Sinking contains an integer value
 Post: The value of sinking is reduced by 1}
begin
   sinking := sinking - 1;
end;

begin  {Main}
 write ('Please enter a number:')'
 readln(sinking);
 while (sinking <> 0) do
  writeln('The next number is ', sinking);
  reduce(sinking);
 writeln('Thats all')
end. {Main}
```

Question Two (10) : Rational Number ADT

A rational number is of the form a/b, where a and b are integers with $b \neq 0$.

Write the interface section of an OBJECT-ORIENTED rational number ADT unit including the type definitions and the four basic arithmetic operations (addition, subtraction, multiplication, and division). Be sure to identify all parts of the unit and ADT. Further, follow the program style guidelines discussed in the assignment and in the lectures.

Question Three(10): Rational Number ADT

For your OBJECT-ORIENTED rational number ADT unit, provide a testing strategy for the unit and write a skeleton tester/driver program (and/or pseudo-code) that would implement your testing strategy.

Question Four(10): Stacks and Queues

Consider those standard operations for a stack {CreateStack, EmptyStack, Pop, Push} as listed in your text book with those standard operations for a queue {CreateQ, EmptyQ, RemoveQ, and AddQ} augmented with {StackFull, QFull}.

Use the above operations to write Pascal procedures (or appropriate pseudo-code) to perform the following operations. Be sure to check for empty and/or full structures as appropriate.

a). Move all entries from a stack to a queue.

b). Move all entries from a queue to a stack.

Question Five(10):

Consider the set of numbers

17	12	26	6	21	39	19	3	14	16	9	33

a. Draw a binary search tree that results from inserting these numbers as nodes into the empty tree .

b. Show Preorder traversal of your binary tree.

c. Draw the binary tree resulting from deletion of node [26]

d. In part a, show an in-order threading of the binary search tree.

e. Is the tree balanced after node [26] deletion? Why?

Question Six(10):

Write a **recursive** procedure Clone that will go through the linked list pointed to by List, leaving the list with an extra copy of each element that it contains. For example, cloning a list with elements (1,2,2,3,4) would result in a list with the elements (1,1,2,2,2,2,3,3,4,4). In your solution, you must assume that the empty list is represented by List := nil;

For this question, assume the following declarations.

```
type
  ListPtr = ^ListRec;

  ListRec =
    record
      Data : integer;
      Next : ListPtr;
    end;
```

```
procedure Clone (var List : ListPtr);
   (*
    *  pre:  List is a valid linked list.
    *  post: List is modified to have every element duplicated.
    *        The duplicated elements occur consecutively within the
list.
    *)
```

```
end; (* Clone *)
```

Question Seven(10):

Design an iterative or recursive algorithm, that when given an arrangement of digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, rearranges all the digits so that the new arrangement represents the next larger value that can be represented by using all these digits (or reports that no such rearrangement exists if no rearrangement produces a larger value). Thus the arrangement of all digits as 5647382901 would produce a rearrangement that is the larger value 5647382910.

Question Eight(10):

Consider the set of numbers

35	15	77	60	22	41

When heap sort is used to put those numbers in non-descending order, a series of "heaps" is formed.

a) Define what a heap means.

b) Draw the initial heap?

c) Draw the heap(s) as the smallest element(s) are removed to produce non-descending order.

Question Nine(10):

Here is a set of numbers to consider:

74	289	670	343	233	860	910	139

Insert the above numbers, in the order given, into the hash table of size 11 drawn below. Use chaining (open hashing) to resolve conflicts, with the following probing technique.

$H = (X \bmod 50) \bmod 11$

0	1	2	3	4	5	6	7	8	9	10

What is the largest number of keys hashing to the same address?

In the hash table above, how many table entries are unused after all the numbers have been inserted?

Question Ten(10):

The following ADTs and a complete set of routines to use them are available:

binary search tree, list, hash table, heap, queue, stack

For each of the following scenarios, choose the **Best data structure** to solve the problem. Note that several data structures may be adequate, but there should be a consideration that favours one over the other.

a. What data structure could be used to simulate a lineup of airplanes waiting to take-off from YEG?
b. You need to write a program that does infinite precision arithmetic on a Pentium. In other words, the program can manipulate numbers that have an arbitrary number of digits (some may exceed 12 digits left of the decimal point). What data structure would you choose to represent these numbers?
c. Your application requires fast insertions and deletions from a data structure - speed is all-important. Data never needs to be extracted in sorted order. You are running on a modern microcomputer with lots of memory.
d. Your problem has defined a large decision table like a soft drink vending machine. No additions or deletions occur to the decision table. The most common operations are to lookup values in the table and dispense change.
e. Your assignment is to write a "GO" program using the Fortran programming language. Unfortunately, Fortran does not allow you to write recursive programs. Therefore, you need to simulate recursive calls and backtracking in Fortran. What data structure will facilitate this?

Problem	Best Data Structure	Why? (Big O equation/value is required)
a.		
b.		
c.		
d.		
e.		