Computer Science Foundation Exam

December 15, 1999

Section I A

No Calculators!

Name: ______________________________

SSN: ______________________________

In this section of the exam, there are three (3) problems.

You must do all of them.

The weight of each problem in this section is indicated with the problem.

The algorithms in this exam are written in a combination of pseudocode, and programming language notation.

Partial credit can not be given unless all work is shown.

If you need extra room to do work to be graded then do so on the last page attached to this test. Make sure to clearly label the problem you are working on.

As always, be complete, yet concise, and above all be neat, credit can not be given when your results are unreadable.

(1, 20%) Given the following array of numbers and algorithm, answer the questions below. Assume that the global array **X[1..n]** is correctly declared and contains the values shown. Also, assume that n is the length of the array, the / operator performs an integer division, and that the array will always contain distinct values, in general.

Array X	3	7	2	6	9	5	8	1
Position	1	2	3	4	5	6	7	8

```
procedure ArrayOperation()
   i, j, k, s, t: integer;
   i ← 0;
   j ← 0;
   s ← 0;
   k ← (1+n)/2;
   while ((s <> k) AND (i <= n))do   // This was a mistake on the actual
        i ← i + 1;                    // exam. It should have read
        s ← 1;                        // ((s <> k) AND (i < n))
        j ← 1;                        // The way it is written there
        while (j <= n) do             // would be an array out of bounds
             if (X[i] > X[j]) then    // error. I'm sorry about that!
                  s ← s + 1;
             endif
             j ← j + 1;
        endwhile
        t ← X[i];
        X[i] ← X[s];
        X[s] ← t;
   endwhile
endprocedure
```

a) (8 pts) Show the array **X** after the procedure was called?

Array X	1	5	3	9	6	7	8	2
Position	1	2	3	4	5	6	7	8

b) (6 pts) Assuming that n is odd and that the value of i is 1 at the end of a call to the procedure above, what is true about the original value stored in X[1] before the procedure call?

X[1] was the median value of all the values stored in the array.

c) (6 pts) What is the minimum running time of this algorithm in terms of n? What is the maximum running time of this algorithm in terms of n? (Do not give an exact answer. Leave your answer in order notation.)

Minimum: O(n)   Maximum: O(n^2)

(2, 18%) The following are Postfix expressions. All values are single decimal digits and the operations are addition "+", subtraction "–", multiplication "*" and division "/". (Note that the final answer and intermediate answers in the stack may not be single decimal digits.) In each box below the Postfix expression, show ONLY the contents of the stack at the indicated point in the Postfix string (point A, B or C). Put the final answer in the blank. If the Postfix string is invalid, carry the operations as far as possible and write "invalid" as the answer. (6 points each)

a) **7 2 * 1 9 ^{A} + - 5 3 - ^{B} / 7 4 - ^{C} * = 6**

A	B	C
9		
1	2	3
14	4	2

b) **1 9 7 8 - ^{A} * 3 * 2 ^{B} 2 3 * - ^{C} - + = -22**

A	B	C
-1	2	-4
9	-27	-27
1	1	1

Next to each Postfix expression, circle one answer to indicate if it is a valid Postfix string or not: (no extra credit for providing the answer, if it is valid) (3 points each)

c) **9 5 + 2 + * 2 1 - + 5 6 3 - +** **Invalid**

d) **6 4 2 6 * + / 3 1 - +** **Valid**

(3, 20%) Answer each of the following "timing" questions concerning an algorithm of a particular order and a data instance of a particular size. Assume that the run time is affected by the size of the data instance and not its composition.

a) (4 pts) For an **O(log$_2$n)** algorithm, an instance with **n = 64** takes **16** seconds.

How long will it take with **n = 512**? **24**

b) (4 pts) For an **O(n^3)** algorithm, an instance with **n = 100** takes **37** milliseconds.

If you used a different-sized data instance and it took **37** seconds(37000 milliseconds), how large must that instance be? **1000**

c) (4 pts) For an **O(2^n)** algorithm, a friend tells you that her instance took **160** seconds to run. You run the same program, on the same machine, and your instance with **n = 4** takes **80** seconds.

What size was her data set? **5**

Given the following pseudocode segment, answer the questions below for an arbitrary **n**:

```
x ← 0
for i ← 1 to (2*n) do
     for j ← 1 to (n-2) do
        x ← x + j
```

d) (2 pts) What is the Order of this code segment, in terms of n? **n^2**

e) (6 pts) What will be the value of **x** (in terms of n) when the `for` loops end? **n(n-1)(n-2)**

Extra Work Page - Please clearly label any work on this page that you would like graded.

Computer Science Foundation Exam

December 15, 1999

Section I B

No Calculators!

Name: ______________________________

SSN: ______________________________

In this section of the exam, there are three (3) problems.

You must do all of them.

The weight of each problem in this section is indicated with the problem.

The algorithms in this exam are written in a combination of pseudocode, and programming language notation. The algorithms that you are asked to produce should use a syntax that is clear and unambiguous.

Partial credit can not be given unless all work is shown.

If you need extra room to do work to be graded then do so on the last page attached to this test. Make sure to clearly label the problem you are working on.

As always, be complete, yet concise, and above all <u>be neat</u>, credit can not be given when your results are unreadable.

(4, 10%) Given a global array of numbers **X[1..n]**, you are to write a recursive function **MaxArray** that will return the largest array element in the range specified by the parameters to the function. Assume that the array was already initialized and that the **MaxArray** function will be called as shown below and answer will be equal to the largest element in the array X in between the indices of i and j, inclusive. (Assume that i <= j for each function call.)

```
answer ← MaxArray(i,j)
```

In the space below, write a RECURSIVE function `MaxArray`.

```
function MaxArray(i, j isoftype in Num)

    if (i = j) then
        return X[i]
    else
        temp isoftype Num
        temp <- MaxArray(i+1,j)
        if (temp > X[i]) then
            return temp
        else
            return X[i]
        endif
    endif
endfunction
```

(5, 16%) Find the closed form or exact value for the following:
(n is an arbitrary positive integer; for part b assume n>12):

a) (4 pts) $\sum_{i=1}^{n}(2ni+3)$ = $\mathbf{n^3 + n^2 + 3n}$

b) (6 pts) $\sum_{i=15}^{n+2}(4i+7)$ = $\mathbf{2n^2 + 17n - 492}$

c) (6 pts) $\sum_{i=15}^{34}(5i-70)$ = **1050**

(6, 16%)

a) (8 pts) Consider the following record type definition:

```
Card_Node definesa record
      suit isoftype Char
      card_num isoftype Num
      next isoftype Ptr toa Card_Node
   endrecord
```

For a Card_Node variable, the suit can either be 'S', 'H','D', or 'C' (for spades, hearts, diamonds, and clubs, respectively) and card_num ranges from 2 to 14, where 11 is a jack, 12 is a queen, 13 is a king and 14 is an ace.

Complete the two blanks below so that the function below returns the number of face cards(jacks, queens, kings, and aces) in a list pointed to by head, a Ptr toa Card_Node.

```
Function Face_Cards returnsa Num (head isoftype in Ptr toa Card_Node)
   if  (head = NIL) then
         Face_Cards returns 0
   else if (head^.card_num > 10) then
         Face_Cards returns 1+ Face_Cards(head^.next)
   else
         Face_Cards returns Face_Cards(head^.next)
   endif
endfunction
```

b) (4 pts) What is the minimum height of a binary tree with 128 nodes? What is the maximum height? (Note, the height of a binary is maximum number of links between any node and the root node. Hence, the minimum and maximum height of a binary tree with 2 nodes is 1.)

Minimum = **7**   Maximum = **127**

c) (2 pts) Given an unsorted array of n numbers, what is the expected amount of time spent to search for a specified value, if we use a "reasonable" algorithm? Simply answer with a Big-O bound in terms of n instead of an exact answer in terms of n.

O(n)

d) (2 pts) Now, answer the question in part c assuming that the array of n numbers is sorted. (Assume that a binary search of some sort is used.)

O(lg n)

Extra Work Page - Please clearly label any work on this page that you would like graded.