CSC 238F **Term Test** **October 28/98**

Name _____ Student No. _____
Tutor's name and/or tutorial room _____

OPEN BOOK
Answer ALL questions on test paper. Use backs of sheets for scratch work.
Total Marks: 40

[10] 1. Consider the following program:

```
% Precondition : x ≥ 0, y ≥ 0, z ≥ 0
loop
    exit when x = 0 AND y = 0 AND z = 0
    if x > 0 then x := x − 1
    elsif y > 0 then y := y − 1; x := x + 5
    elsif z > 0 then z := z − 1; y := y + 5; x := x + 5
    end if
end loop
```

Assuming the precondition holds, prove that the program terminates.

****

This question is very similar to the "two digit base 14" decrement program discussed in class. In this case we have a three digit base 6 counter which is being decremented. Define a variable $k = z \cdot 6^2 + y \cdot 6 + x$. The precondition is a loop invariant and hence $k \in \mathbb{N}$. It suffices to show then that k is strictly decreasing; that is $k_{i+1} < k_i$ where k_i is the value of k after the loop has been executed i times. This is easy to do by cases corresponding to the code.

[10] 2. The correctness of the following binary search program was proved in a course handout.

Precondition: $n \geq 1$. For all u, v such that $1 \leq u \leq v \leq n$, A[u] $\leq$ A[v].

```
first:=1
last:=n
loop
        exit when first=last
        mid:=(first+last) div 2
        if x>A[mid] then
                first:=mid+1
        else
                last:=mid
        end if
end loop
if x=A[first] then found:=true else found:=false end if
```

We claim that the program satisfies the following postcondition:

Postcondition: If there is no u, $1 \leq u \leq n$, such that A[u] $= x$, then $found =$ false.

If for some u, $1 \leq u \leq n$, A[u] $= x$, then

- $found =$ true,
- $1 \leq first \leq n$,
- $A[first] = x$, and
- if $first > 1$ then $A[first - 1] < x$.

This postcondition is stronger than the postcondition stated in the handout. Here we include an additional condition for a successful search; namely, the last condition states that the variable $first$ is the smallest index u such that $A[u] = x$.

Provide an appropriate loop invariant P and show how this loop invariant (and the termination condition) can be used to establish the stronger postcondition. (You do not have to prove that P is a loop invariant.)

****

The loop invariant $P(i)$ is:

If the loop is executed at least i times, then

(i) $1 \leq first_i \leq last_i \leq n$, and

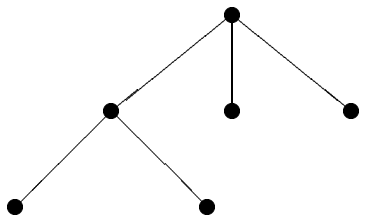
(ii) If for some $u, 1 \leq u \leq n$, A[u] $= x$, then there is some u, $first_i \leq u \leq last_i$ with A[u] $= x$, and if $first_i > 1$ then $A[first_i - 1] < x$.

Now assumming termination, we can substitute $last = first$ in the invariant to obtain:

(i) $1 \leq first \leq n$, and

(ii) If for some $u, 1 \leq u \leq n$, A[u] $= x$, then there is some u, $first \leq u \leq first$ (i.e. $u = first$) with A[u] = A[$first$] $= x$, and if $first_i > 1$ then $A[first_i - 1] < x$.

[10] 3. Similar to binary trees, we want to define various classes of ternary trees. A ternary tree T is a (directed or rooted) tree in which every internal (or interior) node has at most 3 children. Furthermore in order to distinguish these children, every edge (u, v) is labeled $LEFT$, $MIDDLE$, or $RIGHT$ and any two edges (u, v_1) and (u, v_2) (i.e. v_1, v_2 are children of a node u) cannot have the same label. Here is an example of a ternary tree of height 2 where the edge labels have been omitted.



By analogy with the respective classes of binary trees, inductively define the following classes of ternary trees:

(i) The class of all ternary trees of height h.

(ii) The class of perfectly complete ternary trees of height h.

****

In the following definitions, the left subtree $T_L = (V_L, E_L)$, the middle subtree $T_M = (V_M, E_M)$ and the right subtree $T_R = (V_R, E_R)$ are assumed to be node disjoint (i.e. $V_L \cap V_M \cap V_R = \emptyset$) with roots r_L, r_M and r_R. Moreover, $r \notin V_L \cup V_M \cup V_R$ and (in order to formally distinguish the left, middle and right subtrees) the edge (r, r_L) is labelled "LEFT", the edge (r, r_M) is labelled "MIDDLE" and the edge (r, r_R) is labelled "RIGHT". To simplify the definitions, we define the empty tree as a tree of height -1. We will define ternary trees and perfectly complete ternary trees using the same base cases.

Base cases: 1) The empty tree is a tree of height -1.

2) The directed graph $G = (V, E)$ with $V = \{r\}$ and $E = \emptyset$ is a (perfectly complete) ternary tree of height 0.

Induction step for each definition. Let $h > 0$.

(i) A ternary tree of height h is a (edge labelled) directed graph $G = (V, E)$ with left subtree T_L, middle subtree T_M, and right subtree T_R where $V = \{r\} \cup V_L \cup V_M \cup V_R$, $E = \{(r, r_L), (r, r_M), (r, r_R)\} \cup E_L \cup E_M \cup E_R$, and $T_L = (V_L, E_L)$, $T_M = (V_M, E_M)$ and $T_R = (V_R, E_R)$ are ternary trees of height i $(-1 \leq i < h)$ and at least one of T_L, T_M, or T_R has height $h - 1$.

Note: If T_L (respectively, T_M or T_R) is of height -1, then this subtree is missing.

(ii) A perfectly complete ternary tree of height h is a directed graph $G = (V, E)$ where $V = \{r\} \cup V_L \cup V_M \cup V_R$, $E = \{(r, r_L), (r, r_M), (r, r_R)\} \cup E_L \cup E_M \cup E_R$, and $T_L = (V_L, E_L)$, $T_M = (V_M, E_M)$, and $T_R = (V_R, E_R)$ are perfectly complete ternary trees of height $h - 1$.

[10] 4. Suppose we had a program (similar to the MERGE program on page 69 of the notes) called $MERGE3WAY(f, m, r, \ell)$ which has the following specifications.

Preconditions:

(a) $1 \leq f \leq m < r < \ell \leq length(A)$
(b) $A[f..m]$, $A[m+1..r]$, and $A[r+1..\ell]$ are sorted

Time Complexity: $T_{MERGE3WAY}(f, m, r, \ell) = c \cdot (\ell - f + 1)$ for some constant $c > 0$.

Postcondition: $A[f..l]$ has the same elements as before the program is invoked but these elements are now sorted ; all other elements of A are unchanged.

Consider a sorting program that divides an array into three equal (or as equal as possible) size sub-arrays, recursively sorts these sub-arrays, and then merges the three sub-arrays using $MERGE3WAY$.

(i) Describe an appropriate recurrence for the time complexity $T(n)$ of this program assuming the size of the array $n = 3^k$ for some $k \in \mathbb{N}$.

(ii) For what function $f(n)$ is the time complexity $T(n) \in \Theta(f)$?

****

(i)
- $T(1) = c_1$ for some constant c_1
- $T(n) = 3T(n/3) + cn$ for some constant c.
  Equivalently $T(3^k) = 3T(3^{k-1}) + c3^k$.

(ii) Using the general theorem concerning divide and conquer recurrences (see Theorems 2.9 and 2.11 in the course notes) we obtain: $T(n) \in \Theta(n \log n)$.