CSC 236 October 7, 2004
St George Campus — University of Toronto

Assignment #2
Due: October 21, 2004, by 6:00 PM
(in the CSC 236 drop box in BA 2220)

Question 1. (15 marks) The point of this question is to prove the correctness of an interesting and efficient algorithm for finding the *majority* element of an array segment, if one exists. We first need some definitions.

Let A be an integer array, let i be a positive integer, let j be an integer such that $j \geq i-1$, and let u be an integer.

We define $FREQ(A,i,j,u)$ to be the size of the set $\{\ell \in \mathbb{Z} \mid i \leq \ell \leq j \text{ and } A[\ell] = u\}$. That is, $FREQ(A,i,j,u)$ is the number of times that u appears in the array segment $A[i..j]$.

We say that u is the *majority element* of $A[i..j]$ if $FREQ(A,i,j,u) > (j-i+1)/2$, that is, if u appears in more than half of the places of $A[i..j]$. Note that $A[i..j]$ might not have a majority element, but it cannot have more than *one* majority element.

a. (5 marks) Let A be an integer array, let i be a positive integer, and let k, j be integers such that $k > j \geq i-1$. Prove the following:

- If there exists a number u such that $FREQ(A,i,j,u) = (j-i+1)/2$, then $A[i..j]$ does not have a majority element.
- If u is the majority element of $A[i..k]$, and $A[i..j]$ does not have a majority element, then u is the majority element of $A[j+1..k]$.

b. (10 marks) Prove that the program $\text{MAJ}(A,m)$ below is correct with respect to the given specification. (A hint is given on the next page; you should probably read this before looking at the program.)

Precondition: A is an integer array, $m \in \mathbb{N}$, $m \geq 1$, and $A[1..m]$ has a majority element.
Postcondition: $\text{MAJ}(A,m)$ returns the majority element of $A[1..m]$.

```
MAJ(A, m)
j := 0
x := 1
u := A[1]
c := 1
while x ≠ m do
    x := x + 1
    if A[x] = u then
        c := c + 1
    elsif c * 2 = x - j then
        j := x
        x := x + 1
        u := A[x]
        c := 1
    end if
end while
return u
```

continued

Hint: The idea of this program is as follows. We use u to keep track of the conjectured majority element of $A[1..m]$. Initially, u is assigned $A[1]$. We go through the array, recording in variable c a count of how many times u has occurred. If and when we discover that u has occurred in exactly half the array so far, we "throw away" that part of the array and start over. In general, we have "thrown away" the part of the array from 1 through j, and we have examined elements $A[1]$ through $A[x]$.

Part of your loop invariant should be:

- $0 \leq j < x \leq m$
- $A[1..j]$ does not have a majority element.
- $c = FREQ(A, j+1, x, u)$

Remark: Note that this program runs in time *linear* in m. Can you think of a simpler, correct algorithm that is as fast as this one?

Question 2. (10 marks) Consider the programs $\textsc{Sqrt}(k)$ and $\textsc{helper}(k, F, L)$ below; $\textsc{Sqrt}$ uses $\textsc{helper}$ to accomplish its goal. We wish to prove that $\textsc{Sqrt}(k)$ is correct with respect to the following specification.

Precondition for $\textsc{Sqrt}(k)$: $k \in \mathbb{N}$.

Postcondition for $\textsc{Sqrt}(k)$: $\textsc{Sqrt}(k)$ returns $\lfloor\sqrt{k}\rfloor$. (Recall that $\lfloor x \rfloor$ is the largest integer that is less than or equal to x. In particular, if $k \in \mathbb{N}$, then $\lfloor\sqrt{k}\rfloor$ is the integer u such that $u^2 \leq k < (u+1)^2$.)

————————-

$\textsc{Sqrt}(k)$
return $\textsc{helper}(k, 0, k+1)$

————————-

$\textsc{helper}(k, F, L)$
if $L = F + 1$ **then**
    return F
end if
$m := (F + L)$ div 2
if $m^2 \leq k$ **then**
    return $\textsc{helper}(k, m, L)$
else
    return $\textsc{helper}(k, F, m)$
end if

a. (8 marks) State a Precondition/Postcondition specification for $\textsc{helper}(k, F, L)$, and prove that $\textsc{helper}$ is correct with respect to your specification. (Keep in mind Part **b** below.)

b. (2 marks) Use Part **a** to prove that $\textsc{Sqrt}$ is correct with respect to its specification.

continued

Question 3. (10 marks) Joe's boss asked him to write a program $\textsc{Mergesort}(A, f, \ell)$ as in Chapter 2 of the text. However, the boss was careless in stating the postcondition. Specifically he gave as the specification:

Precondition: A is an integer array, $1 \leq f \leq \ell \leq \mathit{length}(A)$
Postcondition: $A[f..\ell]$ has the same elements as before the invocation, in sorted order.

That is, the boss neglected to state that all other elements of the array shouldn't change. In order to teach his boss a lesson, Joe wrote the same program as in the text, except that he added the following line at the end:
if $f > 1$ **then** $A(1) := 0$ **end if**

Joe figured that the program was correct according to the specification he was given, but clearly did not do what the boss intended, and that this was therefore a good trick to play on his boss.

Let's ignore the question of whether or not it is a good idea to play a trick on one's boss, and merely ask whether or not Joe is right that the new program satisfies the specification given him by his boss.

Either prove that the program does meet the specification, or prove (by giving a counterexample) that it does not.

Question 4. (20 marks)

Let $f : \mathbb{N} \to \mathbb{N}$ be the function defined as follows:

$$f(n) = \begin{cases} 1, & \text{if } n = 0 \\ 8 \cdot f(\lfloor \frac{n}{2} \rfloor) + (n-1)^2, & \text{if } n \geq 1 \end{cases}$$

a. (10 marks)
Present a constant c and prove that for all integers $n \geq 1$, $f(n) \leq c \cdot n^3$.
You may not use the general theorem about divide-and-conquer recurrences from the text.
You may use the inequalities $\frac{n-1}{2} \leq \lfloor \frac{n}{2} \rfloor \leq \frac{n}{2}$.
Hint: Prove, for an appropriate c, that for all integers $n \geq 1$, $f(n) \leq c \cdot n^3 - 5 \cdot n^2$.

b. (5 marks)
Prove that for infinitely many $n \in \mathbb{N}$, $f(n) > 8 \cdot n^3$.
Hint: Consider powers of 2.

c. (5 marks)
Prove that for infinitely many $n \in \mathbb{N}$, $f(n) < 3 \cdot n^3$.
Hint: For numbers n of the form $3 \cdot 2^k$, prove that $f(n) < 3 \cdot n^3 - n^2$.