Zhan Yu
zyu293@wisc.edu

(a) Proof.

Claim: The following are loop invariants for indexOfMax algorithm before the while test in line (4).

i. $0 < i \le n$ ii. $m = A[b]$ iii. $m = \max A[0..(i-1)]$ iv. b is the largest index at which $\max A[0..(i-1)]$ appears

P(n): Loop invariants (i), (ii), (iii), (iv) hold when the loop condition is tested for the n-th time.

Base case: show P(1) holds, i.e., loop invariants hold when $i < n$ is tested for the 0 time.

At this point, $i = 1$, $m = A[0]$, $b = 0$

since $n > 0$, (i) $0 < i \le n$ (n is an integer)

since $b = 0$, (ii) $m = A[0] = A[b]$

since there is only one index and element in A,

(iii) $\max A[0..(1-1)] = \max A[0] = A[0]$

and b is the largest index at which $m = \max A[0..(1-1)]$ appears.

□

Inductive step:

Induction hypothesis: The loop invariants hold k-th time the loop is tested i.e. P(k)

Show the loop invariants hold the (k+1)-st time the loop is tested.

Let $i_k$, $m_k$ and $b_k$ be the values of i, m and b, respectively, after k iterations of the while loop

IH: $0 < i_k \le n$, $m_k = A[b_k]$, $m_k = \max A[0..(i_k - 1)]$

b is the largest index at which $\max A[0..(i_k - 1)]$ appears

Consider the $(k+1)^{st}$ time the loop body executes. To get into the body of the loop, the while condition must be true, so we have $i_k < n$

Zhan Yu
zyu293@wisc.edu

Case 1: line (5) $A[i_k] \geq m_k$ is true

From line (6), we get $m_{k+1} = A[i_k]$

From line (7), we get $b_{k+1} = i_k$

From line (9), we get $i_{k+1} = i_k + 1$

since $i_{k+1} = i_k + 1$ ~~(we don't change $i$ in this case)~~

and $0 < i_k \leq n$ (by IH) and $i_k < n$

~~(i) $0 < i_{k+1} \leq n$~~ $0 < i_k < n$ $\Rightarrow 1 < i_k + 1 < n+1$ $\Rightarrow 0 < i_{k+1} < n+1$ ($n \in \mathbb{Z}^+$)

so (i) $0 < i_{k+1} \leq n$ holds

since $m_{k+1} = A[i_k]$ and $b_{k+1} = i_k$

(ii) $m_{k+1} = A[b_{k+1}]$ holds

since $m_k = \max A[0..(i_k - 1)]$ (by IH), $A[i_k] \geq m_k$ and $b_{k+1} = i_k$

$\because m_{k+1} = A[b_{k+1}]$

$= A[i_k]$

$\geq \max A[0..(i_k - 1)]$

since $i_k = i_{k+1} - 1$

$m_{k+1} \geq \max A[0..(i_{k+1} - 2)]$

so, $m_{k+1}$ is the max element in $A[0..(i_{k+1} - 1)]$

$\therefore$ (iii) $m_{k+1} = \max A[0..(i_{k+1} - 1)]$ holds

$\because m_{k+1} = A[b_{k+1}]$

$\therefore$ (iv) $b_{k+1}$ is the largest index at which $\max A[0..(i_{k+1} - 1)]$ appears

Zhan Yu
zyu293@wisc.edu

Case 2: line (5) $A[i_{k+1}] \geq m_k$ is fake

so $A[i_k] < m_k$

From line (9), we get $i_{k+1} = i_k + 1$

since $i_k < n$

by IH, $0 < i_k < n$

$1 < i_k + 1 < n+1$

since $i$ is an integer

$0 < 1 < i_{k+1} < n+1$

(i) $0 < i_{k+1} \leq n$ holds

by IH, $m_k = A[b_k]$

since $b_{k+1} = b_k$, $m_{k+1} = m_k$

(ii) $m_{k+1} = A[b_{k+1}]$ holds

since $A[i_k] < m_k$, $m_k = \max A[0..(i_k - 1)]$ (by IH)

~~since $m_k = m_{k+1}$ $i_{k+1} = i_k + 1$~~

~~$m_{k+1} = \max A[0..(i_{k+1} - 1)]$~~

~~$m_k = \max A[0..i_k]$ $(m_k > A[i_k]$~~

~~$m_k = m_k$~~

$\max A[0..(i_k - 1)] > A[i_k]$

$\therefore m_k = \max A[0..i_k]$

since $m_k = m_{k+1}$ $i_{k+1} = i_k + 1 \Rightarrow i_k = i_{k+1} - 1$

(iii) $m_{k+1} = \max A[0..(i_{k+1} - 1)]$ holds

$\because m_{k+1} = A[b_{k+1}]$

$\therefore A[b_{k+1}] = \max A[0..(i_{k+1} - 1)]$

$\therefore$ (iv) $b_{k+1}$ is the largest index at which $\max A[0..(i_{k+1} - 1)]$ appears

■

Zhan Yu
zyu293@wisc.edu

(b)

Partial correctness:

Assume we have valid input and the program terminates on that input.

Since the program returns b, we need to show that when the loop exits, b is the index of largest element appear.

when the loop exists, it does so because $i<n$ evaluates to false,

This means $i \geq n$

Loop invariants (i) says $0<i\leq n$

So when loop exits $i=n$

Loop invariants (iii) says $m = \max A[0..(i-1)]$

so when loop exists, m is the largest element in A

Loop invariants (iv) says b is the largest index at which $\max A[0..(i-1)]$ appears, and $m=A[b]$

so, The program returns b, which contains the correct value. □

Termination:

Assume the input is valid, i.e. $n \in z^+$ and an array $A[0..(n-1)]$ of n integers

The only reason the program wouldn't terminate is if we had an infinite loop.

Loop exits when $i \geq n$ (when tested on line 4)

i starts as 1 (line 1)

i increase by 1 (line 9) in each iteration of the loop when if statement is false.

n doesn't change

since n is finite and unchanging at some point $i \geq n$ becomes true.

So loop cannot be infinite

So, the implementation is correct □