Arun Hari Anand CS30 HW#5

① Precondition: n is a valid index of a, such that $n \geq 1$.
Postcondition: The program terminates and returns the maximum element of $a[1 \ldots n]$.

Loop invariant: At the commencement of iteration I, m holds the greatest element of $a[1 \ldots i-1]$. Additionally, $2 \leq i \leq n+1$, and the array has not been altered.
Proof of LI: by induction
Claim: If the precondition holds for the method call, the LI is true at the start of every iteration.
Base case: We can observe that the loop invariant holds at the commencement of the first iteration. $m = a[1]$, and $i = 2$, and the LI states that m must be the largest element of $a[1 \ldots i-1]$, which in this case is $a[1]$. Since $a[1]$ must be the largest element of $a[1]$ by default, $i = 2 \leq n+1$, and the array has not been altered, LI holds for the first iteration.

Induction step: Assume that LI holds at the start of an arbitrary iteration $\hat{I}$. We now attempt to show that LI holds for the next iteration I. Assume that $\hat{i}$ is the value of i at the start of iteration $\hat{I}$.

1. m is the maximum element of $a[1 \ldots \hat{i}-1]$, by the induction hypothesis
2. Since there is an iteration I that follows $\hat{I}$, $\hat{i} \neq n$. Thus $2 \leq \hat{i} \leq n$.
3. There are 2 cases:
   i) $a[i] \leq m$, in which case m is still the largest element of $a[1 \ldots i]$, and so, according to line 4, m is unchanged. The array is also not modified.
   ii) $a[i] > m$, in which case $a[i]$ is the maximum element of $a[1 \ldots i-1]$ since m is the maximum of $a[1 \ldots \hat{i}-1]$ by IH. The value of m is thus appropriately changed to be $a[i]$. The array is also not modified.
4. Thus, LI holds for iteration I, by (2), (3).

The claim thus follows from the principle of induction.

Theorem 1: The max program is correct.
Assume that the precondition holds. It follows from LI that, at the start of the iteration when $i = n+1$, the value of m is the maximum element of $a[1 \ldots n]$. We also know that, from the for-loop, the loop terminates when $i = n+1$, and returns m. Since m is the max of $a[1 \ldots n]$, the postcondition is satisfied, and the program is correct. Theorem 1 thus follows.

max method
(2) Precondition: n is a valid index of a such that $n \geq 1$.
Postcondition: The program terminates and returns the maximum element of $a[1...n]$

helper method
precondition: p and q are valid indices of a, such that $p \leq q$.
postcondition: The program terminates and returns the maximum element of $a[p...q]$

Proof of correctness, by induction:
We first attempt to show through induction that helper is correct.

Predicate $P(n) \equiv$ helper works correctly for $|[p \cdots q]| = n$

Claim $\equiv P(n)$ holds for all $n \geq 1$.

Base case: When $n=1$, $|[p...q]| = 1$, which means that p must equal q. Thus, in the first if-statement in helper, the statement evaluates to true and $a[p]$ is returned, which is the largest element of $a[p...q]$ by default. So, the base case, (i.e) $P(1)$ holds.
Induction step: Assume that $P(k)$ holds for an arbitrary $k \geq 1$. We now attempt to show that $P(k+1)$ holds true. Since we know that $k+1 \geq 2$, we know that $p \neq q$, so the first if-statement is skipped. m is now made to equal helper$(a, p+1, q)$. Since $|[p+1...q]| = k$, we know from the induction hypothesis that $P(k)$ holds and m holds the maximum element of $a[p+1...q]$. In the final if statement, $a[p]$ is returned iff $a[p] > m$, which is valid because $a[p]$ is the maximum of $a[p...q]$ iff $a[p] > m = max(a[p+1..q])$. As such, $a[p]$ is only returned if it is the maximum element in the array, or else the maximum element in the array is m and so m is returned. $P(k+1)$ is thus shown to hold true. The claim thus follows from the principle of induction. Helper thus works for all $n \geq 1$.
Now that we have proven the correctness of helper, we attempt to show the correctness of max. Assume that all preconditions for max(a, n) hold. Since max(a,n) returns the same value as helper$(a, 1, n)$ and all the preconditions for helper are satisfied, we know that it returns the maximum element of $a[1...n]$. As such we know that the postcondition for max(a,n) is satisfied, and the program is correct.

Polysum

precondition: $n \geq 0$, $x \in \mathbb{Z}$, a is an array of real numbers.

postcondition: the program terminates and returns $\sum_{i=0}^{n} a[i]x^{i}$

helper

precondition: p and q are valid indices of array a, such that $p \leq q$, $x \in \mathbb{Z}$

postcondition: The method terminates and returns $\sum_{i=p}^{q} a[i]x^{i-p}$

Proof of correctness: Induction.

We now attempt to prove helper is correct through induction.

$P(n) \equiv$ helper works correctly for $|[p \ldots q]| - 1 = n$

Claim: $p(n)$ holds for all $n \geq 0$.

Base case: Assume that n is 0; that is $|[p \ldots q]| = 1$. This would mean that $p = q$. From the first if-statement of helper, we can observe that the program returns $a[p]$, which is equal to $\sum_{i=p}^{p} a[i]x^{i-p}$, so the base case holds, and $P(0)$ is true.

Induction step: Assume that $P(k)$ is true for an arbitrary $k \geq 0$. We now attempt to show that $P(k+1)$ is true. Since we know $k \geq 0$, $k+1 \geq 1$. Therefore, $p \neq q$, and the first if-statement is skipped. On the return statement, we know that $|a[p+1 \ldots q]| - 1 = k$, and so by the induction hypothesis, helper$(a, p+1, q)$ returns correctly $\sum_{i=p+1}^{q} a[i]x^{i-p-1}$. This figure multiplied by x gives us $\sum_{i=p+1}^{q} a[i]x^{i-p}$. Since $a[p]$ is added, and $a[p] = \sum_{i=p}^{p} a[i]x^{i-p}$, the total sum of the return statement must equal $\sum_{i=p}^{q} a[i]x^{i-p}$. As such, we know that $P(k+1)$ holds true, since the postcondition is satisfied. Thus, the claim follows from the principle of induction, and so helper is correct for all $n \geq 0$.

Proof of correctness of polysum: Assume that all preconditions hold. We thus know that since $n \geq 0$, helper$(a, 0, n, x)$ satisfies all preconditions for helper. Since helper$(a, 0, n, x)$ returns $\sum_{i=0}^{n} a[i]x^{i}$, and polysum returns the same thing as helper$(a, 0, n, x)$ we know that the postconditions for polysum are satisfied. As such, the program is correct.

(4) TOH: preconditions: $n \geq 1$; there are n disks at peg 1 in decreasing order of size and no disks at pegs 2 and 3.

Postconditions. The program terminates and moves all the pegs from peg 1 to peg 2 using only legal moves.

helper: preconditions: If α_i, α_j, and α_k are the initial peg states of pegs i, j, and k respectively:

(i) i, j and k are a permutation of $1, 2, 3$

(ii) $(\alpha_1, \alpha_2, \alpha_3)$ is a legitimate configuration

(iii) $\alpha_i = \alpha_i' \cdot d_m d_{m-1} \ldots d_1$, where $d_m, d_{m-1} \ldots d_1$ are the disks that need to be moved and α_i' is a legitimate peg state.

(iv) peg j is either empty or the topmost disk on peg j is larger than any of the top m disks of α_i

(v) α_k is either λ or the topmost disk of peg k is larger than any of the top $m-1$ disks of α_i

postconditions: If β_i, β_j and β_k are the peg-states of pegs i, j and k respectively after the method has been executed;

i) i, j, and k are still the same permutation of $1, 2$ and 3.

ii) $(\beta_1, \beta_2, \beta_3)$ is a legitimate configuration

iii) $\beta_i = \alpha_i$, $\beta_j = \alpha_j \cdot d_m d_{m-1} \ldots d_1$, $\beta_k = \alpha_k$

Proof of correctness through induction:

Predicate $P(n) \equiv$ If all the preconditions hold for a method call: helper(n, i, j, k) then the program terminates, all the postconditions for helper hold, and only legal moves have been made.

Claim: $\forall_{n \geq 0}\ P(n)$

Base case: Assume that $n=0$. and also assume that all the preconditions for helper hold. Since $n=0$, and so in the second line of helper $(m>0)$ returns false, the program will not execute any of the lines of code within the if-statement. Thus, the final peg-state is the initial peg state. As such, i, j, k will remain unchanged. Since $(\alpha_1, \alpha_2, \alpha_3) = (\beta_1, \beta_2, \beta_3)$ and $(\alpha_1, \alpha_2, \alpha_3)$ is a legitimate configuration according to the precondition ii), $(\beta_i, \beta_j, \beta_k)$ is a legitimate configuration. Additionally, since $d_m d_{m-1} \dots d_1 = \lambda$, $\alpha_i = \alpha'_i$ according to precondition (ii) and so $\beta_i = \alpha_i = \alpha'_i$; Also $\alpha_j \cdot d_m d_{m-1} \dots d_1 = \alpha_j = \beta_j$, and $\beta_k = \alpha_k$. Since all the postconditions have been met, and no moves were made so they were all legal by default, the base case i.e. $P(0)$ holds true.

Induction step:

Assume that $p(x)$ holds for some arbitrary $x \geq 0$. That is assuming that all the preconditions for helper have been met, helper(x, i, j, k) meets all postconditions and makes only valid moves : this is the induction hypothesis.

Let us now try to show that $P(x) \Rightarrow P(x+1)$.

① Case 1 : Assume that all the preconditions for helper(x+1, i, j, k) hold.

Preconditions for helper($x+1$, i, j, k) are as follows

(i) i, j, k is a permutation of 1,2,3.

(ii) $(\alpha_1, \alpha_2, \alpha_3)$ is a legitimate configuration

(iii) $\alpha_i = \alpha'_i \cdot d_{x+1} d_x \ldots d_1$, where $d_{x+1} d_x \ldots d_1$ are the disks to be moved and α'_i is a legitimate peg state.

(iv) peg j is either empty or the topmost disk on peg j is larger than any of the top $x+1$ disks α_i.

(v) α_k is either λ or the topmost disk of peg k is larger than any of the top x disks of α_i.

② If we can show that all preconditions for helper (x, i, j, k) hold, we know that the following are true:

i) i, j, k are still the same permutation of i, j, k

ii) $(\beta_1, \beta_2, \beta_3)$ is a legitimate configuration.

iii) $\beta_i = \alpha'_i$, $\beta_j = \alpha_j \cdot d_x d_{x-1} \ldots d_1$, $\beta_k = \alpha_k$

③ Since we know that $x \geq 0$, $x+1 \geq 1$. So, $x+1 > 0$ returns true and the statements inside the if-statement are executed.

④ The first call within the if-statement is helper (x, i, k, j). i) Since we have assumed that all preconditions for helper $(x+1, i, j, k)$ hold, we know that i, j, k are a permutation of 1, 2, 3. ii) We also know that $(\alpha_1, \alpha_2, \alpha_3)$ is a legitimate configuration. iii) From precondition iii) in ①, we know that if $d_x d_{x-1} \ldots d_1$ are to be moved, then for this method call, we know that $\alpha''_i = \alpha'_i \cdot d_{x+1}$ is a legitimate peg state. iv) since we know that peg k is either empty, or the topmost disk of peg k is larger than any of the top x disks of j from precondition (v) mentioned in ①, we know that precondition iv) for this call is satisfied, and

v) since $\alpha_j = \lambda$ or the topmost disk of peg j

is larger than any of the top $x+1$ disks of peg i from precondition (iv) of ①, we also know that the topmost disk of peg j is larger than any of the top $x-1$ disks of peg i. So, all 5 preconditions have been met for helper(x, i, k, j), and so from IH we know that all postconditions must be satisfied, and only legal moves have been performed.

post conditions for helper(x, i, k, j) are as follows
If β''_i, β''_j and β''_k are the peg states of pegs i, j, and k
after method call:

- i) i, j and k are still the same permutation of 1, 2, 3.
- ii) $(\beta''_1, \beta''_2, \beta''_3)$ is a legitimate configuration
- iii) $\beta''_i = \alpha''_i = \alpha'_i \cdot d_{x+1}$ from (4) (iii)

$\beta''_k = \alpha_k \cdot d_x d_{x-1} \ldots d_1$

$\beta''_j = \alpha_j$

⑤ Since we know that the peg state of i is currently $\beta_i'' = \alpha_i' \cdot d_{x+1}$, we know that it must have atleast the disk d_{x+1}, and that it is at the top of peg i. We also know from the postconditions of the method call from ④ that the current peg-state of peg j is still α_j (from postcondition (iii)) and we know from precondition (iv) for the initial method call in ① that peg j is either empty or the topmost disk of peg j is larger than the top $x+1$ disks at peg i. Thus we know that peg j is either empty or the topmost disk of peg j is larger than d_{x+1} of peg i. As such, since we know that d_{x+1} is the topmost disk at i, it is legal to move it to peg j. After this line of the program is executed, based on the postconditions mentioned in ④ and the move just made, we also know that the current peg states of i, j, and k, here represented as β_i', β_j' and β_k' are as follows:

(i) $\beta_i' = \alpha_i'$, (ii) $\beta_j' = \alpha_j \cdot d_{x+1}$ (iii) $\beta_k' = \alpha_k \cdot d_x d_{x-1} \ldots d_1$

Where, $(\beta_i', \beta_j', \beta_k')$ is a legitimate configuration from the postconditions of ⑤ and the fact that the move just executed is a legal move.

⑥ We know that

i) i, j, k remain unchanged over the course of the program, so they must still be the same permutation of i, j and k.

ii) $(\beta_1', \beta_2', \beta_3')$ is a legitimate configuration from ⑤

iii) $\beta_k' = \alpha_k \cdot d_x d_{x-1} \ldots d_1$, where $d_x d_{x-1} \ldots d_1$ are the disks that need to be moved, and α_k is a legitimate peg state from precondition (ii) in ①

(iv) $\beta_j' = \alpha_j \cdot d_{x+1}$. So peg j must either be empty, or it must atleast have d_{x+1} at the top. We know from precondition i) in ① that d_{x+1} must be larger than each of $d_x, d_{x-1} \ldots d_1$. We also know that, since β_j' is a legitimate peg state, each disk in α_k is smaller than each disk of $d_x, d_{x-1} \ldots 1$. As such we know that d_{x+1} is larger than any of the top $x+1$ disks of peg k.

v) We know from precondition i) in ① that α_i is a legitimate pegstate, and from precondition (ii) of ① that $\alpha_i = \alpha_i' \cdot d_{x+1} d_x \dots d_1$. As such, it must be the case that the topmost disk of α_i' must be larger than each of $d_{x+1}, d_x \dots d_1$. Since the top $(x+1)-1$ disks of peg k are $d_x, d_{x-1} \dots d_1$, from ⑤, we know that the topmost disk of $\beta_i' = \alpha_i'$ is larger than each of $d_x, d_{x-1} \dots d_1$. We also know from $P(x)$ that only legal moves have been performed.

It follows that all of the preconditions for the method call helper$(m-1, k, j, i)$ hold, where $m = x+1$. Since $m-1 = x$, and we know that $P(x)$ holds from IH. So, all the postconditions for this method call must hold after the call. They are as follows:

i) i, j and k are still the same permutation of 1, 2 and 3.

ii) $(\beta_1, \beta_2, \beta_3)$ is a legitimate configuration, where $(\beta_1, \beta_2, \beta_3)$ is the configuration after the method call.

iii) $\beta_k = \beta_k' = \alpha_k$ from precondition iii) in ⑥

$\beta_j = \beta_j' d_x d_{x-1} \dots d_1 = \alpha_j \cdot d_{x+1} d_x \dots d_1$, from postcondition iii) in ⑤

$\beta_i = \beta_i' = \alpha_i'$, from postcondition ii) in ⑤

As is evident, the postconditions for the method call helper$(x+1, i, j, k)$ have been satisfied, and only legal moves have been made, $P(x+1)$ holds for this case.

Case 2: Assume that the preconditions for helper$(x+1, i, j, k)$ do not hold. Since $P(x+1)$ states that:

preconditions for helper$(x+1, i, j, k)$ $\Longrightarrow$ postconditions for helper$(x+1, i, j, k)$ hold

As such, if the first part of the implication is F, $P(x+1)$ is true by default. So $P(x+1)$ holds.

As such, the claim thus follows from the principle of induction.

Proof of correctness of TOH(n)

Assume that preconditions for TOH hold. That is

i) $n \geq 1$

ii) there are n disks at peg 1 in decreasing order of size, and no disks at pegs 2 and 3.

Let us now consider the preconditions for helper(n, 1, 2, 3) for an arbitrary n.

i) i, j, k is a permutation of 1, 2, 3

ii) Since the disks at peg 1 are arranged in decreasing order of size, and pegs 2 and 3 are empty, $(\alpha_1, \alpha_2, \alpha_3)$ is a legitimate configuration.

iii) $\alpha_1 = \alpha_i = \alpha_i' \cdot d_n d_{n-1} \ldots d_1$, where $\alpha_i' = \lambda$ and $d_n d_{n-1} \ldots d_1 = \alpha_i$. As such, we know that $d_n d_{n-1} \ldots d_1$ are the disks that need to be moved, and α_i' is a legitimate peg state.

iv) peg $j = 2$ is empty from precondition (ii) for TOH

v) peg $k = 3$ is empty from precondition (ii) for TOH

As such all the preconditions for helper (n, 1, 2, 3) hold. So we know that the postconditions must be satisfied when helper terminates. Since TOH(n) returns the same result as helper (n, 1, 2, 3), and we know that P(n) states that

i) the program terminates

ii) i, j and k are the same permutation of 1, 2, 3

iii) $(\beta_1, \beta_2, \beta_3)$ is a legitimate configuration

iv) $\beta_i = \alpha_i = \lambda$, $\beta_j = \alpha_j \cdot d_n d_{n-1} \ldots d_1 = d_n d_{n-1} \ldots d_1$ and $\beta_k = \alpha_k = \lambda$.

v) Only legal moves have been made

All the postconditions for TOH(n) have been met. The program is thus correct for all $n \geq 1$.

⑤ Precondition: $n \geq 1$
Postcondition: The program returns true if there exists a majority value in the array, and returns false if not.

Loop invariant for the main loop:

Assume that $\alpha(x, i)$ = the number of times the element x appears in $a[1...i]$.

Assume that $\alpha^*(x, i)$ = the number of indices between $[1...i]$ such that x does not appear in that index of the array a. In other words

$$\alpha^*(x,i) = |\{y \in \mathbb{Z} \mid 1 \leq y \leq i \wedge a[y] \neq x\}|$$

The loop Invariant for the main loop is as follows:

(i) $\alpha(m, i-1) - \alpha^*(m, i-1) \leq c$

(ii) For all x in a such that $x \neq m$: $\alpha^*(x, i-1) - \alpha(x, i-1) \geq c$

(iii) $2 \leq i \leq n+1$

(iv) $c \geq 0$

Proof of loop invariant; induction

. Claim: Loop invariant holds at the start of every iteration of the loop.

Base case: At the start of the first loop $i=2$. $\alpha(m, i-1) - \alpha^*(m, i-1) = \alpha(a[1], 1) - \alpha^*(a[1], 1)$, due to line (2) of the program, which assigns $a[1]$ to m. Since $a[1]$ is the only element in $a[1...1]$, $\alpha(a[1], 1) = 1$, $\alpha^*(a[1], 1) = 0$, and so $\alpha(a[1], 1) - \alpha^*(a[1], 1) = 1 - 0 = 1 \leq 1 = c$. Similarly, for all $x \neq m$, $\alpha^*(x, 1) = 1$ and $\alpha(x, 1) = 0$; $\alpha^*(x,1) - \alpha(x,1) = 1 - 0 = 1 \geq 1 = c$. Since the loop terminates when it is not the case that $2 \leq i \leq n$, $2 \leq i \leq n$ for $i = 2$. And since 1 has been assigned to the variable c, $c \geq 0$. Therefore, LI holds for the base case.

Induction step:

Assume that LI holds at the start of an arbitrary iteration $\hat{I}$, where $i = \hat{i}$, and $c = \hat{c}$. We attempt to prove now that LI holds at the start of the iteration immediately following $\hat{I}$, here called I'.

① At the start of $\hat{I}$, the following holds true by induction hypothesis:

i) $\alpha(\hat{m}, \hat{i}-1) - \alpha^*(\hat{m}, \hat{i}-1) \leq \hat{c}$

ii) For all x such that $x \neq m$, $\alpha^*(x, \hat{i}-1) - \alpha(x, \hat{i}-1) \geq \hat{c}$

iii) $2 \leq \hat{i} \leq n+1$

iv) $\hat{c} \geq 0$.

② Since there is an iteration following $\hat{I}$, we know further that $2 \leq \hat{i} \leq n$. Thus, at the start of iteration I', $2 \leq \hat{i}+1 \leq n+1$.

③ There are three cases inside the for loop:

• Case 1: $\hat{c} = 0$.

We know from the program that $\hat{c}=0$ implies that $m = a[\hat{i}]$ and $c=1$ at the start of the next iteration I', so,

$$\begin{aligned}
& \alpha(m, i-1) - \alpha^*(m, i-1) \\
&= \alpha(m, \hat{i}) - \alpha^*(m, \hat{i}) \quad , \text{ since } i-1 = \hat{i} \\
&= \alpha(m, \hat{i}-1)+1 - \alpha^*(m, \hat{i}-1) , \text{ since } m = a[\hat{i}] \\
&\leq 1 \quad , \text{ since we know that } \alpha(m, \hat{i}-1) - \alpha^*(m, \hat{i}-1) \leq 0, \text{IH} \\
&\leq c \quad , \text{ since } c=1 .
\end{aligned}$$

We have thus shown that $\alpha(m, i-1) - \alpha^*(m, i-1) \leq c$ for this case

We also may show that:

For all $x \neq m$,
$$\begin{aligned}
& \alpha^*(x, i-1) - \alpha(x, i-1) \\
&= \alpha^*(x, \hat{i}) - \alpha(x, \hat{i}) , \text{ since } i-1 = \hat{i} \\
&= \alpha^*(x, \hat{i}-1)+1 - \alpha(x, \hat{i}-1) , \text{ since } m = a[\hat{i}] \neq x \\
&\geq 1 , \text{ since we know that } \alpha^*(x, \hat{i}-1) - \alpha(x, \hat{i}-1) \geq \hat{c}, \text{ by IH and } \hat{c}=0. \\
&\geq c , \text{ since } c=1
\end{aligned}$$

so $\alpha^*(x, i-1) - \alpha(x, i-1) \geq c$, for this case

Also, $c \geq 0$, since $c=1$

Since $c=1$, $c \geq 0$.

As such, loop invariant holds at the start of iteration I'. The claim thus follows from the principle of induction for case 1.

case 2: $a[\hat{i}] = m$

We know from the program that $a[\hat{i}] = m$ implies that c becomes $\hat{c}+1$ at the start of iteration I'. So,

$$\begin{aligned}
& \alpha(m, i-1) - \alpha^*(m, i-1) \\
&= \alpha(m, \hat{i}) - \alpha^*(m, \hat{i}) \quad , \text{ since } i-1 = \hat{i} \\
&\leq \alpha(m, \hat{i}-1) + 1 - \alpha^*(m, \hat{i}-1) , \text{ assuming that } m = \hat{i} \text{ maximizes } \alpha(m, \hat{i}) - \alpha^*(m, \hat{i}) \\
&\leq \hat{c}+1 , \text{ since we know that } \alpha(m, \hat{i}-1) - \alpha^*(m, \hat{i}-1) \leq \hat{c} \text{ from IH} \\
&= c , \text{ since we know } c = \hat{c}+1
\end{aligned}$$

Thus we have shown that $\alpha(m, i-1) - \alpha^*(m, i-1) \leq c$, for this case

We can also show that

For all $x \neq m$:
$$\begin{aligned}
& \alpha^*(x, i-1) - \alpha(x, i-1) \\
&= \alpha^*(x, \hat{i}) - \alpha(x, \hat{i}) , \text{ since } i-1 = \hat{i} \\
&= \alpha^*(x, \hat{i}-1)+1 - \alpha(x, \hat{i}-1) , \text{ since } a[\hat{i}] = m \\
&\geq \hat{c}+1 , \text{ since we know from IH that } \alpha^*(x, \hat{i}-1) - \alpha(x, \hat{i}-1) \geq \hat{c}
\end{aligned}$$

$= c$, since $\hat{c}+1 = c$ by the if statement executed at $\hat{I}$

So $\alpha^*(x, i-1) - \alpha(x, i-1) \geq c$
Also, $c = \hat{c}+1 \geq 1$, so $c \geq 0$.

Case 3: $\hat{c} \neq 0$ and $a[\hat{i}] \neq m$

We know from the program that $a[\hat{i}] \neq m$ and $\hat{c} \neq 0$ implies that $c = \hat{c}-1$ for the next iteration I'! Thus:

$$\alpha(m, i-1) - \alpha^*(m, i-1)$$
$$= \alpha(m, \hat{i}) - \alpha^*(m, \hat{i}), \text{ since } \hat{i} = i-1$$
$$= \alpha(m, \hat{i}-1) - (\alpha^*(m, \hat{i}-1)+1), \text{ since } m \neq a[\hat{i}]$$
$$= \alpha(m, \hat{i}-1) - \alpha^*(m, \hat{i}-1) - 1, \text{ distributing}$$
$$\leq \hat{c}-1, \text{ since we know that } \alpha(m, \hat{i}-1) - \alpha^*(m, \hat{i}-1) \leq \hat{c} \text{ from IH}$$
$$= c, \text{ since we know } c \geq 0, \text{ from IH}$$

So we have shown that $\alpha(m, i-1) - \alpha^*(m, i-1) \leq c$ for this case

We can also show that for all $x \neq m$:

$$\alpha^*(x, i-1) - \alpha(x, i-1)$$
$$= \alpha^*(x, \hat{i}) - \alpha(x, \hat{i}), \text{ since } \hat{i} = i-1$$
$$\geq \alpha^*(x, \hat{i}-1) - (\alpha(x, \hat{i}-1)+1), \text{ since } m \neq a[\hat{i}] \text{ and } a[\hat{i}] = x \text{ minimizes the value of } \alpha^*(x,\hat{i}) - \alpha(x,\hat{i})$$
$$= \alpha^*(x, \hat{i}-1) - \alpha(x, \hat{i}-1) - 1, \text{ distributing}$$
$$\geq \hat{c}-1, \text{ since } \alpha^*(x, \hat{i}-1) - \alpha(x, \hat{i}-1) \geq \hat{c} \text{ by IH}$$
$$= c, \text{ since } c = \hat{c}-1$$

We have thus shown that $\alpha^*(x, i-1) - \alpha(x, i-1) \geq c$ for this case
Also, $\hat{c} > 0$ and so $\hat{c}-1 \geq 0$, so $\hat{c}-1 = c \geq 0$

Since we have shown for all three possible cases that $\alpha(m, i-1) - \alpha^*(m, i-1) \leq c$, $\alpha^*(x, i-1) - \alpha(x, i-1) \geq c$ for all $x \neq m$, and that $c \geq 0$, and we know $2 \leq i \leq n+1$, we know that LI holds at the start of I' under all 3 cases. So, the claim follows from the principle of induction.

Proof of correctness of majority: Assume that preconditions for majority(a, n) hold. Consider the loop invariant at the start of the iteration where $i = n+1$:

For all $x \neq m$, $\alpha^*(x, n) - \alpha(x, n) \geq c$, and $c \geq 0$.

So, $\alpha^*(x, n) \geq \alpha(x, n) + c$, where again $c \geq 0$. If the number of times x does not appear in the array is atleast equal to the number of times it does appear (plus a nonnegative constant), then x cannot be a majority element by the definition of a majority element. This is because $\alpha(x, n) \leq \lceil \frac{n+1}{2} \rceil$.

As such, all elements of a that are not m are automatically ruled out. Since the program then $d = \alpha(m, n)$ and returns whether or not $\alpha(m, n) \geq \lceil \frac{n+1}{2} \rceil$

it returns whether or not m is a majority value, by the definition of the majority value. Since m is the only viable candidate, we know that the post condition holds, and so the program is correct.