1. (a) *Precondition*: A is a non-empty array (*i.e.*, $\text{len}(A) \geqslant 1$).

   Postcondition: the elements of A are unchanged and, if A contains a majority element e, then the call MAJORITY(A) returns e; otherwise, MAJORITY(A) returns some element of A.

   (There is undoubtedly a way to describe more precisely which element gets returned in the second case, but it would be needlessly complicated and not add anything useful to the postcondition.)

   (b) To make the proof easier to write, we introduce some notation: let $\#m(i)$ denote the number of occurrences of value m in $A[0 \ldots (i-1)]$.

   Assume that the precondition holds.

   We prove that the following statement is a loop invariant, by induction on the number of iterations of the loop:

   LI = "$1 \leqslant i \leqslant \text{len}(A)$ and $0 \leqslant c \leqslant \#m(i) \leqslant (i+c)/2$ and $\forall x \in A[0 \ldots (i-1)], x \neq m \Rightarrow \#x(i) \leqslant (i-c)/2$."

   Base Case: $c_0 = 1$ and $m_0 = A[0]$ and $i_0 = 1$.

   Hence, $1 \leqslant i_0 \leqslant \text{len}(A)$ (since A is non-empty by the precondition) and $\#m_0(i_0) = 1$ and $0 \leqslant c_0 \leqslant \#m_0(i_0) \leqslant (i_0 + c_0)/2$ and there are no other values in $A[0 \ldots (i_0 - 1)]$. So LI_0 holds.

   Ind. Hyp.: Assume $k \geqslant 0$ and LI_k holds.

   Ind. Step: Assume $i_k \leqslant \text{len}(A) - 1$ so there is one more iteration.

   Either $c_k = 0$ or $c_k > 0$.

   Case 1: If $c_k = 0$.

   Then $c_{k+1} = 1$ and $m_{k+1} = A[i_k]$ and $i_{k+1} = i_k + 1$.

   Then $0 \leqslant i_{k+1} \leqslant \text{len}(A)$ (because $i_k \leqslant \text{len}(A) - 1$).

   Also, $0 \leqslant c_{k+1} \leqslant \#m_{k+1}(i_{k+1})$.

   Now, either $m_{k+1} = m_k$ or $m_{k+1} \neq m_k$.

   Subcase 1(a): If $m_{k+1} = m_k$.

   Then we know $\#m_k(i_k) \leqslant (i_k + c_k)/2$ (from LI_k), so

$$\begin{aligned} \#m_{k+1}(i_{k+1}) &= \#m_k(i_k) + 1 && \text{(since } m_k = m_{k+1} = A[i_k]\text{)}, \\ &\leqslant (i_k + 0)/2 + 1 && \text{(since } c_k = 0\text{)}, \\ &= (i_k + 2)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

   Also, $\#x(i_{k+1}) = \#x(i_k)$ for all $x \in A[0 \ldots (i_{k+1} - 1)]$ such that $x \neq m$, and we know $\#x(i_k) \leqslant (i_k - c_k)/2$ from LI_k and $(i_{k+1} - c_{k+1})/2 = (i_k + 1 - 1)/2 = (i_k + 0)/2 = (i_k + c_k)/2$. Hence, LI_{k+1} holds.

   Subcase 1(b): If $m_{k+1} \neq m_k$.

   Then we know

$$\begin{aligned} \#m_{k+1}(i_{k+1}) &= \#m_{k+1}(i_k) + 1 \\ &\leqslant (i_k - c_k)/2 + 1 && \text{(by } LI_k\text{)}, \\ &= (i_k + 2)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

   Also,

$$\begin{aligned} \#m_k(i_{k+1}) &= \#m_k(i_k) \\ &\leqslant (i_k + c_k)/2 && \text{(by } LI_k\text{)}, \\ &= (i_k + 0)/2 = (i_{k+1} - c_{k+1})/2 \end{aligned}$$

   And for all elements x such that $x \neq m_k$ and $x \neq m_{k+1}$,

$$\begin{aligned} \#x(i_{k+1}) &= \#x(i_k) \\ &\leqslant (i_k - c_k)/2 && \text{(by } LI_k\text{)}, \\ &= (i_k - 0)/2 = (i_{k+1} - c_{k+1})/2 \end{aligned}$$

Hence, LI_{k+1} holds.

Then LI_{k+1} holds in every subcase.

Case 2: If $c_k > 0$.

Either $A[i_k] = m_k$ or $A[i_k] \neq m_k$.

Subcase 2(a): If $A[i_k] = m_k$.

Then $c_{k+1} = c_k + 1$, $m_{k+1} = m_k$ and $i_{k+1} = i_k + 1$.

Then $0 \leqslant i_{k+1} \leqslant \text{len}(A)$ (because $i_k \leqslant \text{len}(A) - 1$).

Also, $0 \leqslant c_{k+1}$ (because $0 \leqslant c_k$ by LI_k).

Also,

$$\begin{aligned} c_{k+1} &= c_k + 1 \\ &\leqslant \#m_k(i_k) + 1 \qquad (\text{by } LI_k), \\ &= \#m_k(i_{k+1}) \qquad (\text{since } A[i_k] = m_k), \\ &= \#m_{k+1}(i_{k+1}) \qquad (\text{since } m_{k+1} = m_k), \end{aligned}$$

and

$$\begin{aligned} \#m_{k+1}(i_{k+1}) &= \#m_k(i_k) + 1 \\ &\leqslant (i_k + c_k)/2 + 1 \qquad (\text{by } LI_k), \\ &= (i_k + 1 + c_k + 1)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

Finally, for all elements x such that $x \neq m_k$,

$$\begin{aligned} \#x(i_{k+1}) &= \#x(i_k) \qquad (\text{since } A[i_k] = m_k \neq x), \\ &\leqslant (i_k - c_k)/2 \qquad (\text{by } LI_k), \\ &= (i_k + 1 - c_k - 1)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

Hence, LI_{k+1} holds.

Subcase 2(b): If $A[i_k] \neq m_k$.

Then $c_{k+1} = c_k - 1$, $m_{k+1} = m_k$ and $i_{k+1} = i_k + 1$.

Then $0 \leqslant i_{k+1} \leqslant \text{len}(A)$ (because $i_k \leqslant \text{len}(A) - 1$).

Also, $0 \leqslant c_{k+1}$ (because $c_k > 0$ in this case).

Also,

$$\begin{aligned} c_{k+1} &= c_k - 1 \\ &\leqslant \#m_k(i_k) - 1 \qquad (\text{by } LI_k), \\ &= \#m_k(i_{k+1}) - 1 \qquad (\text{since } A[i_k] \neq m_k), \\ &< \#m_{k+1}(i_{k+1}) \qquad (\text{since } m_{k+1} = m_k), \end{aligned}$$

and

$$\begin{aligned} \#m_{k+1}(i_{k+1}) &= \#m_k(i_k) \\ &\leqslant (i_k + c_k)/2 \qquad (\text{by } LI_k), \\ &= (i_k + 1 + c_k - 1)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

Also,

$$\begin{aligned} \#A[i_k](i_{k+1}) &= \#A[i_k](i_k) + 1 \\ &\leqslant (i_k - c_k)/2 + 1 \qquad (\text{by } LI_k \text{ since } A[i_k] \neq m_k), \\ &= (i_k + 1 - c_k + 1)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

Finally, for all elements x such that $x \neq m_k$ and $x \neq A[i_k]$,

$$\begin{aligned} \#x(i_{k+1}) &= \#x(i_k) \\ &\leqslant (i_k - c_k)/2 \qquad (\text{by } LI_k), \\ &< (i_k + 1 - c_k + 1)/2 = (i_{k+1} + c_{k+1})/2 \end{aligned}$$

Hence, LI_{k+1} holds.

Then LI_{k+1} holds in every subcase.

Then LI_{k+1} holds in every case.

Having proved the loop invariant, we now use it to prove termination and partial correctness.

Termination: By LI, we know that the expression "$\text{len}(A) - i$" is always a natural number. Moreover, since $i_{k+1} = i_k + 1$ for every iteration, this expression is strictly decreasing. Hence, by well ordering, the loop must terminate.

Partial Correctness: When the loop terminates, we know that $i = \text{len}(A)$ and LI holds.

Either A contains some majority element e, or it does not.

Case 1: If A contains a majority element e.

Then $e = m$.

Otherwise (if $e \neq m$), LI would imply that $\#e(\text{len}(A)) \leqslant (\text{len}(A) - c)/2 \leqslant \text{len}(A)/2$. But this contradicts the definition of "majority".

Hence, the algorithm returns $m = e$, *i.e.*, the postcondition is satisfied.

Case 2: If A does not contain a majority element.

Then the algorithm returns m, which is equal to some element of A.

Hence, the postcondition is satisfied.

The postcondition is satisfied in every case.

We have proved that the algorithm terminates and that the postcondition is satisfied, for every input that satisfies the precondition. By definition, the algorithm is correct.