1.

A state diagram for the given DFA is provided below:

Step 1: Create the associated GNFA, depicted below:

Step 2: Remove $q_0$ to obtain the below state diagram:
Step 3: Remove $q_1$ to obtain the below state diagram, where:

1. $a = 0^*2 \cup (0^*1)(0 \cup 20^*1)^*(1 \cup 20^*2)$
2. $b = 0^* \cup (0^*1)(0 \cup 20^*1)^*(20^* \cup \varepsilon)$
3. $c = (10^*2 \cup 0)(10^*1 \cup 2)(0 \cup 20^*1)^*(1 \cup 20^*2)$
4. $d = (10)^* \cup (10^*1 \cup 2)(0 \cup 20^*1)^*(20^* \cup \varepsilon)$

Step 4: Remove $q_2$ to obtain:

Thus, the desired regular expression is given by $ac^*d \cup b$, or equivalently

$$0^*2 \cup (0^*1)(0 \cup 20^*1)^*(1 \cup 20^*2))((10^*2 \cup 0)(10^*1 \cup 2)(0 \cup 20^*1)^*(1 \cup 20^*2))^*((10)^* \cup (10^*1 \cup 2)(0 \cup 20^*1)^*(20^* \cup \varepsilon))$$

A state diagram of the given NFA is depicted below:

The equivalent DFA is depicted below. Note that my states have simplified names. For example state ‘135’ is just shorthand for $\{q_1, q_3, q_5\}$, state ‘35’ is short for $\{q_3, q_5\}$, etc.
3.

Step 1: Create the associated GNFA, depicted below:
Step 2: Remove $q_0$ to obtain:

Step 3: Remove $q_1$ to obtain:

Step 4: Remove $q_2$ to obtain:
Step 5: Remove $q_3$ to obtain the following state diagram, where
\[ a = (00 \cup \varepsilon)(100 \cup 1)^* (1 \cup \varepsilon) \quad \text{and} \quad b = (00 \cup \varepsilon)(100 \cup 1)^* (1) \cup \varepsilon: \]

Step 6: Remove $q_4$ to obtain:

Step 7: Remove $q_5$ to obtain:

Hence, the regular expression associated with the given NFA is
\[
(a11 \cup a)(011 \cup 0)^* b = ((00 \cup \varepsilon)(100 \cup 1)^* (1 \cup \varepsilon))(11 \cup (00 \cup \varepsilon)(100 \cup 1)^* (1 \cup \varepsilon)) (111 \cup 0)^* \cup ((00 \cup \varepsilon)(100 \cup 1)^* (1) \cup \varepsilon)
\]

4(a)

The problem statement is true. $L$ is regular, and thus it is recognized by a DFA. Call this DFA $D = (Q, \Sigma, \delta, q_0, F)$. I will build an NFA, $N$, to recognize half($L$). Let $N = (Q', \Sigma, \delta', q'_0, F')$. I define each of the components of $N$ below.

1. $Q' - \{q'_0\} = \{(q, \text{parity}) \mid q \in Q, \text{parity} \in \{\text{even}, \text{odd}, \text{oddLast}\}\}$.

2.1 $\delta'(q'_0, c) = \{(q_0, \text{EorO}) \mid \text{EorO} \in \{\text{even}, \text{odd}\}\}$.

2.1. $\delta'((q, \text{odd}), a) = \{(q^*, \text{odd}) \mid q^* = \delta(\delta(q, c), a) \text{ for arbitrary } a, c \in \Sigma\} \cup \{(q^{**}, \text{oddLast}) \mid q^{**} = \delta(\delta(q, b), a), c) \text{ for arbitrary } a, b, c \in \Sigma\}.

2.2. $\delta'((q, \text{even}), a) = \{(q^*, \text{even}) \mid q^* = \delta(\delta(q, c), a) \text{ for arbitrary } a, c \in \Sigma\}.$
2.3. $\delta'(q, \text{oddLast}, a) = \emptyset$ for all $a \in \Sigma$.

3. $F' = \{(q_F, \text{parity}) \mid q_F \in F, \text{parity} \in \{\text{odd, even, oddLast}\}\}$.

The idea behind $N$ is that it presupposes that the input string $w$ that it is fed is the ‘half’ of some string in $L$. As such, $N$ attempts to nondeterministically ‘fill in the gaps’, as it were, between the characters of $w$ such that the resulting ‘filled out’ string is a string in $L$. Now, the ‘filled in’ conjugate of the input given to $N$ may either be of odd or even length. That is, if $w = w_1 \ldots w_n$, with $n \geq 1$, then $N$ guesses whether the ‘filled out’ conjugate of $w$ in $L$ is either of the form ‘? $w_1$? $w_2$?…? $w_n$?’ or of the form ‘? $w_1$? $w_2$?…? $w_n$’, and it attempts to replace the ‘?’s with distinct characters in $\Sigma$ such that the resulting string is in $L$. If the conjugate is of the former type, then it is of odd length, and so an accepting computation of input $w$ will consist in $N$ being driven through states of the form $(q, \text{odd})$, during which $N$ simulates the computation of all possible conjugates of $w$ on the DFA $D$. At every point when $N$ consumes a character of such an input, it either guesses that the character is the very last one in $w$ or it guesses that it is not the last character in $w$. If $N$ guesses the former, then it must also guess the state to which the simulation of $D$ is driven when an arbitrary odd-position character immediately following the current input character in the hypothetical conjugate string is also consumed; that is, the character ‘guessed’ by $N$ to be the very last character in the conjugate of $w$ in $L$. If the guess is incorrect; viz., that the current input character is not the last character of $w$, then that particular computation branch dies, since there are no transition arrows exiting any state of the form $(q, \text{oddLast})$.

However, if the conjugate is of even length, then an accepting computation of $w$ will consist in $N$ being driven through states of the form $(q, \text{even})$, during which $N$ simulates the computations of all possible conjugates of $w$ on the DFA $D$. Given that the conjugate string in $L$ is even, it follows that the last character of $w$ is also the last character of this conjugate, and thus we don’t require $N$ to do any last-character guessing of the sort done in the event that the conjugate is of odd length.

If any such conjugate exists for the input, then $N$ will accept its input, and if no such conjugate exists, then $N$ will reject its input. That is, if there exists some string $x$ in $L$ satisfying half($x$) = $w$, then $N$ accepts the string. On the other hand, if there does not exist a string $x$ in $L$ satisfying half($x$) = $w$, then $w$ is rejected by $N$. (The latter holds simply because $N$ has attempted every single possible ‘filling out’ of $w$, and if every such attempt yields a conjugate which isn’t in $L$, then $w$ can’t possibly be the ‘half’ of any string in $L$). Combining these two statements, it follows that $N$ accepts $w$ if and only if $w$ is ‘half’ of some string $x$ in $L$. Thus, $N$ recognizes the language half($L$) and we conclude that half($L$) is regular.
4(b)

Consider the language $L = (10)^n(00)^n$. Then $\text{half}(L) = \{x \in 0^+ \mid \#_0(x) \mod 2 = 0\}$. Then $\text{half}(L) = (00)^+$. Since it is described by a regular expression, it must be regular.

Next, I prove that $L$ is non-regular. Supposing that it is regular, the string $s = (10)^p(00)^p$, with $p$ the associated pumping length, satisfies $|s| \geq p$. So there must exist a splitting of $s$ into three components $s = xyz$, with $|xy| \leq p$, $|y| > 0$, and $xy^iz \in L$ with $i \in \mathbb{N}$. This implies that $xy^2z \in L$. Let's consider $y$'s general forms:

1. $y$ contains 01 as a prefix and 1 as a suffix.
2. $y$ contains 01 as a prefix and 0 as a suffix.
3. $y$ contains 10 as a prefix and 1 as a suffix.
4. $y$ contains 10 as a prefix and 0 as a suffix.
5. $y = 0$.
6. $y = 1$.

If case 1 holds, then the number of initial 10 substrings comprising $xy^2z$ exceeds the number of double 0’s following the last occurrence of substring 10 in $xy^2z$. If case 2 holds, then $xy^2z = \alpha 001\beta$, and thus cannot possibly be of the form $(10)^n(00)^n$. If case 3 holds, then $xy^2z = \alpha' 110\beta'$, and thus cannot possibly be of the form $(10)^n(00)^n$. If case 4 holds, then the number of initial 10's comprising $xy^2z$ is strictly greater than the number of double 0's following the last occurrence of substring 10 in $xy^2z$. If case 5 holds, then either $xy^2z = \alpha^* 1001\beta^*$ (which cannot possibly be of the form $(10)^n(00)^n$) or $y$ is the suffix of the last 10 in $xyz$. If the latter holds, then the number of 0's following the last appearance of 10 in $xy^2z$ is of odd parity. If case 6 holds, then $xy^2z = \alpha'' 11\beta''$, which isn’t even of the correct format for strings in $L$. 