**Solution**
a)

**Algorithm 1** given $S$ and $d$ finds a $d$-net of $S$ of minimum size

**Require:** An array of integers $S$ sorted in ascending order and a positive integer $d$
where $S = \{s_1 < s_2 < ... < s_n\}$
Let $R$ be an empty set
$r \leftarrow 0$
**for** $s_i \leftarrow s_2$ to $n$ **do**
  **if** $s_i > s_1 + d$ **then**
    add $s_{i-1}$ to $R$ {finds the first element of $R$}
    $break$
  **end if**
**end for**
**for** $s_i \leftarrow s_2$ to $n$ **do**
  **if** $abs(R[r] - s_i) > d$ **then**
    add $s_i$ to $R$
    $r \leftarrow r + 1$
  **end if**
**end for**
**return** $R$

Time Complexity: Find the first element of $R$ : $O(n)$ + build $R : O(n) = O(n)$
Space Complexity: $R = O(n)$

*Proof.* We claim that this greedy method will find a $d$-net of $S$ of minimum size. Let $W = \{w_1, w_2, ..., w_m\}$ be any $d$-net of $S$ of minimum size where $m \leq n$. Assume that the greedy algorithm produces $R = \{r_1, r_2, ..., r_k\}$ where $k \leq n$. If $k = m$ then our greedy algorithm found a $d$-net of $S$ of minimum size. We will show that $k = m$.
We know that the first item in the $d$-net of $S$ must be within $d$ distance away from $s_1$. By finding the largest element in $S$ that falls within this distance, we make sure that $r_1$ maximizes the distance to $s_1$ which minimizes $|R|$. It must be the case that $w_1 - s_1$ is also close to zero. Now, assume inductively that or some $j \geq 1$, we have $r_t \geq w_t$, for each $t \leq j$. If $j = m$, we are done. So, suppose $j < m$ and consider $w_{j+1}$ and $r_{j+1}$. We know that $w_j \leq r_j$, and our greedy algorithm will pick the next integer that is more than $d$ away from $r_j$, so it must be the case that $r_{j+1} \geq w_{j+1}$. Thus, our greedy algorithm maximizes the distance between elements of $R$ and $S$. □

b) The first element $a_1$ of our $d-$ approximation for $S$ $A$ will be $s_1 + d$ and the last element $a_k$ will be $s_n - d$ if $s_n$ is not within $d$ units away from $s_1 + d$. If $s_n$ is within $d$ units away from $a_1$ then we simply return $A = \{a_1\}$. Otherwise, we walk through $S$ and see if $s_i$ falls within distance of either $a_1$ or $a_k$. If it does, we move on to $s_{i+1}$. If it does not, we add $s_i + d$ to $A$ and set it as the lower bound for comparison. Meaning that we will check if $s_{i+1}$ falls within $d$ units of $s_i + d$ or $a_k$ and so on.
Time Complexity: Determine $a_1$ and potential $a_k$: $O(n)$ + Walk through $S : O(n)$ * add to $A : O(n) = O(n)$
Space complexity: $A : O(n)$

*Proof.* We claim that this greedy method will find a $d-$approximation of $S$ of minimum size. Let $B = \{b_1, b_2, ..., b_m\}$ be a minimum size $d-$approximation of $S$ where $m \leq n$. Assume our greedy algorithm produces $A = \{a_1, a_2, ..., a_k\}$ where $k \leq n$. If $k = m$ then our greedy algorithm found a $d$-approximation of $S$ of minimum size. We will show that $k = m$.
Clearly, $a_1 \geq b_1$ since our algorithm makes sure $a_1$ is furthest away from $s_1$. It is crucial that $a_i \geq b_i$ for some $i \leq k$ as it shows that we are maximizing the distance between $a_i$ and the corresponding element of $S$, the greater distance will require less elements in $A$. Assume, inductively, that for some $j \geq 1$, we have $a_j \geq b_j$. If $j = m$, we are done as our greedy solution is of same size as an optimal solution. Now, suppose $j < m$ and consider $a_{j+1}$ and $b_{j+1}$. We know $a_j \geq b_j$ and our greedy algorithm will pick an $a_{j+1}$ so that it is as far away from the next $s_i$ that does not satisfy $|s_i - a_j| \leq d$. Thus, $a_{j+1} \geq b_{j+1}$ so our greedy solution is optimal. □

c) Let $r = |R|$ and $a = |A|$.
Claim: $a \leq r \leq 2a$.

*Proof.* First, we will prove that $a \leq r$. $a$ could be smaller than $r$, because $A \not\subset S$ meaning that there is more freedom when picking elements of $A$. Restricting $R$ to being a subset of $S$ limits the possible elements of $R$ and some choices are not maximizing distance. Also, $a$ could be equal $r$ as you can just find a $d-$approximation of $S$ of minimum size by running the algorithm to find a $d-$net of $S$ of minimum size.
Now, we will show that $r \leq 2a$. If $R \cap A = \{\}$, then no element $r \in R$ is able to maximize distance the same way all elements $a \in A$ do. This means that for all $a$, $R$ must contain an element $r_i < a$ and an element $r_j > a$ such that $|r_i - s_l| \leq d$, $|r_i - s_{l+1}| > d$, and $|r_j - s_{l+1}| \leq d$ for some integer $l$. Also if $R \cap A \neq \{\}$, then it follows that an element of $S$ is optimal for maximizing the distance for some elements in $S$.
Therefore, since $a \leq r$ and $r \leq 2a$, then $a \leq r \leq 2a$. □