We use the following variables:

1. \( T \): an array of \( n \) numbers s.t. \( T[i] \) is the time of 
   student \( i \)'s exam \( (1 \leq i \leq n) \) (input to algorithm)

2. \( K \): integer giving number of rooms currently 
   assigned

3. \( L \): list of indices of assigned rooms \( 1, \ldots, K \) 
   in order of start time of last exam scheduled 
   so far

4. \( S \): an array s.t. \( S[i,j] \), if defined, is the 
   start time of the last exam so far scheduled 
   in room \( j \).

5. \( A \): an array s.t. the room assigned to 
   student \( i \) is given by \( A[i,j] \). (Output)

6. \( O \): an array of students \( 1, \ldots, n \) ordered 
   by the start times of their examinations, 
   as given in \( T \).

Basic idea of algorithm is to order the 
students by the start times of the exams 
and assign successive students to the room which 
first became free or, if there is no such 
room, to a new room.

We may assume \( n > 0 \).
Allocate \(( T )\)

1. \(D \leftarrow \text{list } [i_1, \ldots, i_n] \text{ ordered by } T[i]\)
2. \(k \leftarrow 1\)
3. \(L \leftarrow [i_1]\) \hspace{1cm} \{ \text{Allocate room } r_i \text{ to first student} \}
4. \(S[i] = T[C[i]]\)
5. \(A[C[i]] = 1\)
6. \(\text{for } i \text{ from 2 to } n \text{ consider in order of start time} \)
7. \(\text{let } j \text{ be first element of } L\)
8. \(i \leftarrow (S[i] + 36 \geq T[C[j]]) \text{ if clash} \)
9. \(k \leftarrow k + 1\)
10. \(\text{add } k \text{ to end of } L\) \hspace{1cm} \{ \text{Get a new room; it will have latest start time so far} \}
11. \(S[k] \leftarrow T[C[j]]\)
12. \(A[C[j]] \leftarrow k\)
13. \(\text{else} \)
14. \(\text{remove } j \text{ from start of } L\) \hspace{1cm} \{ \text{Assign } r_j \text{ to current student; it will now have latest start time so far} \}
15. \(\text{add } j \text{ to end of } L\)
16. \(S[i] \leftarrow T[C[i]]\)
17. \(A[C[i]] \leftarrow j\)
18. \(\text{Return } \langle i_1, A[C[i_1]] \rangle, \ldots, \langle n, A[C[n]] \rangle\)

\textbf{Note:} The sort algorithm used in line 1 can be bucket sort, with 4,224 buckets (no of minute slots) each having a capacity of \(n\). The output is a list of pairs \(\langle i, j \rangle\), meaning that student \(s_i\) is assigned to room \(r_j\). The variable \(k\) is the number of rooms used.
3. The algorithm maintains the following invariants:

(1) $S_{ij}$, if defined, is the start time of the last exam currently scheduled for room $j$

(2) $L = \left[ l_1, \ldots, l_k \right]$ is the list of numbers $\left[ l_1, \ldots, l_k \right]$ ordered by $S_{ij}$ ($1 \leq i \leq k$)

These invariants trivially hold at the start of the for-loop.

When the next student, $s_{ij}$, is considered, the algorithm checks (line 8) to see if there is a room (used so far) that has become free. If not, the if-statement is true, and a new room $k+1$ is used. It will have the latest start time considered, so it is added to the end of $L$. $S$ is updated and (1) and (2) are maintained. If a room has become free, $r_j$ will certainly be a candidate since $r_j$ has the earliest start time of rooms considered so far, and the student can be assigned. Notice that $r_j$ will now have the latest start time, so it goes to the end of the list; (1) and (2) are maintained.

At the end of the algorithm, all students will be legally assigned (no clashes).

To see that the number $k$ of rooms is
minimal, suppose k is incremented (line 9).
For all i (1 ≤ i ≤ k)

(i) $s[i] + 36 ≥ t[0][i]$ (otherwise condition in line 8 would fail)

(ii) $s[i] ≤ t[0][i]$ start-times are ordered

From (i) and (ii) all of the examinations in $r_1, \ldots, r_k$ (incremented k) clash with each other, at the time $t[0][i]$. So there is no possibility of having fewer than k rooms.

2. Bucket sort takes linear time. The rest of Allocate(1) has a single loop carried out n times, with a fixed maximum number of instructions. So time complexity is $O(n)$

4. At the point where room $r_k$ is brought into operation (line 9), all rooms $r_1, \ldots, r_k$ are in use at some point $t[0][i]$, so you certainly need at least k examiners.