Set A Part 1

get_inspection_time(n, D)

//for each ward, read its inspection time and store it in the array D

1. **for** *i* = 1 **to** n
   read *D[i]*

Evaluation criteria : **[1 mark]**
Division: reading inspection time of each ward and storing array D- 1 mark

visit_ward(D, C, n, t)

1. *w* = 0 // keeps track of the number of wards completed
2. *time* = 0 // keeps track of the total time taken
3. **while** *w* < *n*
   for *i* = 1 **to** *n*
   if D[*i*] > 0
   if D[*i*] > *t*
   D[*i*] = D[*i*] - *t*
   time = *time* + *t*
   else if D[i] <= *t*
   C[i] = *time* + D[i] // set completion time in C[i]
   D[i] = 0
   time = *time* + *t* // always increment by *t*
   w = *w* +1 // one more ward completed

Evaluation criteria : **[5 marks]**
Division: Finding the next ward to inspect - 2 marks Calculating the time of completion - 3 marks

display(C, n)

// Prints the contents of the array C, with the elements separated by a single space

1. **for** *i* = 1 **to** n
   print C[*i*]; print(' ');

Evaluation criteria : **[1 mark]** Division: print the time of completion of each ward separated by a space - 1 mark

Set A Part 2

get_times(R, D, n)

1. read the value of *n*
2. **for** *i* ← 1 **to** n
   do read R[i] // *ready_time*
   read D[i] // *inspection time*

Evaluation criteria : **[0.25 mark]** Division: reading ready time and inspection time of each ward and storing in array R and D respectively- 0.25 mark

visit_wards(R, D, n, t)

//Visits all the wards by following the specifications given in the question.

1. *time* ← 0 //to track total time taken
2. **while** TRUE
   do *ready_index* ← -1 // index of the ward that is selected for inspection
   min_time ← 9999 // initialize to a value greater than all possible
   // inspection times
   for *i* ← 1 **to** *n*
   //select *w_id* s with *ready_time* <= *time* and *inspection_time* greater than 0
   do if R[*i*] <= time **and** D[*i*] > 0
   //select *w_id* s with *inspection_time* less than *min_time*
   then if D[*i*] < *min_time*
   then *min_time* ← D[*i*]
   ready_index ← *i*
   //select the *w_id* s with *inspection_time* equal to *min_time*

```
                    else if D[i] = min_time
                            then if R[ready_index] > R[i]
                                    then ready_index ← i
            if ready_index ≠ -1
                    then if D[ready_index] > t
                            then D[ready_index] ← D[ready_index] - t
                                 time ← time + t
                            else C[ready_index] ← time + D[ready_index]
                                 time ← time + D[ready_index]
                                 D[ready_index] ← 0
            else break  // break out of while
```

Evaluation criteria : **[2 marks]** Division: finding the next ward to inspect- 1 mark Calculating the time of completion - 1 mark

ward_list(C, n)

// Print the *w_id* and time of completion of inspections for each of the n wards

//Arrange C[1 … n] in non-decreasing order using any sorting algorithm

```
1. for i ← 1 to n
        do for j ← n downto i + 1
              do if C[j] < C[j - 1]
              then exchange (C[j] , C[j - 1])
2. for i ← 1 to n
        do print i; print ‘ ‘; print C[i]; C
```

Evaluation criteria : **[0.75 mark]** Division: sorting the wards based on completion time - 0.75 mark