IT-205: Capstone Project

Sleep Inducer

Project ID
07

Team Name
Team Hummingbird

Team Members:
Parv Rana (202301112)
Bhumsar Boro (202301002)
Anshuman Bhagat (202301170)
Mohit Tigga (202301111)

Github Repository
https://github.com/BHUMSAR123/new_dsa_project_repo

Contents

Introduction

What the Problem states -

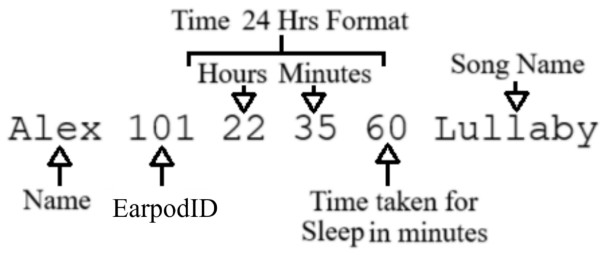
- We have to build a Sleep inducer.
- There are N inmates in total each having his/her own sleep time (randomly initialized).
- The Sleep Inducer is installed in a hostel with M sleeping dorms. It notes every inmate's daily sleeping habits and starts sleep-inducing music at bedtime. It works for multiple inmates.
- The music should not continue beyond p minutes after going to bed (assuming it takes that much time for every inmate to fall off to sleep).
- The Sleep Inducer should be able to play for all of them without disturbing anyone (i.e., playing music while someone is asleep). Therefore, it has to assign the correct number of inmates for each dorm before starting to operate.

Project Objectives -

- Building an efficient sleep inducer that assigns inmates to the dorm rooms on the basis of their sleeping habits.
- It plays music according to their sleeping habits for p minutes in order to make him/her sleep in a way that the other person in the dorm does not get disturbed.

Parameters Explained

The parameters N is the number of inmates
M is the number of sleeping dorms.
P is the time one takes to sleep
The input text should be of the beside format



NOTE: Name should not have spaces

Algorithm

The algorithm of the code is as follows

1. Data Structures:

- 'Clock': Represents time with hours and minutes.
- 'Record': Contains details of a student's name, AirPod ID, sleep time, time taken to play music, and song name. It also overloads the '<' operator for sorting based on sleep time.
- 'music_channel': Represents a music channel with an ID and AirPods assigned.
- 'Dorm': Extends 'music_channel' and includes dormitory-specific details like the dorm name and records of inmates.

2. Main Functionality:

- Inserting Details ('insert_details'):
  - Open the input file.
  - Read each line from the file.
  - Parse the line to extract student details and create 'Record' objects.
  - Add each 'Record' object to the 'records' vector.

- Assigning to Dorms ('assign_to_dorms'):
  - Sort the 'records' vector based on sleep time.
  - Iterate through each record.
  - For each record, iterate through each dormitory.
  - If a dormitory has space and doesn't conflict with existing inmates' sleep schedules, assign the record to that dormitory.
  - If no suitable dormitory is found, create a new dormitory and assign the record to it.

- Sleep Inducer ('sleep_inducer'):
  - Simulate time from 20:00 to 02:59.
  - For each minute:
    - Check if any inmate is scheduled to sleep at that time in any dormitory.
    - If an inmate is found, print their details and play the assigned music.

- Printing Dorm Details ('print_dorm_details'):
  - Iterate through each dormitory.
  - Print dormitory details including assigned inmates and music channel.

3. User Interaction:
- Prompt the user for a choice:
  - If the user chooses 1, print dormitory details.
  - If the user chooses 2, execute the sleep inducer.

4. Memory Management:
- Deallocate memory for inmate records after use.

5. Execution:
- Read input file, assign inmates to dorms, and execute user-selected action (print details or execute sleep inducer).

6. Cleanup:
- Deallocate dynamically allocated memory for inmate records.

This algorithm outlines the steps performed by the code to read input, assign dorms, simulate sleep induction, interact with the user, and clean up memory.

Pseudocode

```
// Define class for Clock
class Clock:
    int hours
    int minutes

// Define class for Record
class Record:
    string name
    int airpod_id
    Clock sleep_time
    int time_taken
    string song_name

    // Overload less than operator for sorting
    bool operator<(Record& mate):
        if sleep_time.hours != mate.sleep_time.hours:
            return sleep_time.hours < mate.sleep_time.hours
        return sleep_time.minutes < mate.sleep_time.minutes

// Define class for music_channel
class music_channel:
    int channel_id = 0
    int airpods_assigned[2]

// Define class for Dorm which extends music_channel
class Dorm extends music_channel:
    string dorm_name = ""
    Record* inmate1 = nullptr
    Record* inmate2 = nullptr

// Function to read student records from file and populate vector
function insert_details(records: vector<Record>):
    open input file
    if file not opened:
        print error message
        return
    for each line in file:
        parse line to extract student details
        create new Record object with extracted details
        add new Record object to records vector
    close input file

// Function to assign students to dorms based on their sleep schedules
function assign_to_dorms(records: vector<Record>, dorms: vector<Dorm>):
    sort records vector based on sleep time
    initialize count = 1000, channel_id_count = 202301
    create empty set assigned_names
    for each record in records:
        initialize assigned = false
        for each dorm in dorms:
            if dorm has space and doesn't conflict with existing inmates' sleep
schedules:
                assign record to dorm
                mark record as assigned
                update assigned_names set
                update dorm's airpods_assigned array
                break loop
        if record not assigned:
            create new dormitory
            assign record to new dormitory
            update assigned_names set
            update new dormitory's details
            increment count
    end for

// Function to simulate playing sleep-inducing music in dorms
function sleep_inducer(dorms: vector<Dorm>):
    print "Playing sleep-inducing music in all dorms:"
    for each hour from 20 to 23:
        for each minute from 0 to 59:
            for each dorm in dorms:
                if inmate1 of dorm scheduled to sleep at current time:
                    print details of inmate1 and play assigned music
                if inmate2 of dorm scheduled to sleep at current time:
                    print details of inmate2 and play assigned music
    end for
    for each hour from 0 to 2:
        for each minute from 0 to 59:
            for each dorm in dorms:
                if inmate1 of dorm scheduled to sleep at current time:
                    print details of inmate1 and play assigned music
                if inmate2 of dorm scheduled to sleep at current time:
                    print details of inmate2 and play assigned music
    end for
```

```
// Function to print dormitory details
function print_dorm_details(dorms: vector<Dorm>):
    for each dorm in dorms:
        print dorm details including assigned inmates and music channel
    end for

// Main function
function main():
    initialize empty vectors records and dorms
    insert_details(records) // Read student records from file
    assign_to_dorms(records, dorms) // Assign students to dorms
    print menu options
    read user choice
    if choice is 1:
        print_dorm_details(dorms) // Print dormitory details
    else if choice is 2:
        sleep_inducer(dorms) // Simulate playing sleep-inducing music
    cleanup dynamically allocated memory
    return 0
```

Data Structures Used

The data structures used in the provided code along with the reasoning :-

1. Vector:

- Purpose: Vectors are used to store dynamically sized sequences of elements, such as student records (`records`) and dormitory information (`dorms`).
- Why: Vectors are chosen because they provide dynamic resizing, efficient random access, and contiguous memory storage, making them suitable for storing and accessing a variable number of student records and dormitory details.

2. String:

- Purpose: Strings are used to store textual data, such as student names, song names, and dormitory names.
- Why: Strings are appropriate for storing textual data as they offer flexible manipulation, ease of use, and support for various operations like concatenation, comparison, and substring extraction.

3. Unordered Set:

- Purpose: Unordered sets are used to store unique student names (`assigned_names`) to track already assigned students, ensuring that each student is assigned to only one dormitory.
- Why: Unordered sets offer constant-time average insertion, deletion, and lookup operations, making them suitable for efficiently maintaining a collection of unique student names without any particular order.

4. Array:

- Purpose: Arrays are used to store the AirPod IDs assigned to inmates in dormitories (`airpods_assigned`).
- Why: Arrays provide a fixed-size collection of elements with constant-time access to individual elements by index. In this case, arrays are used to store the AirPod IDs assigned to students in dormitories, facilitating easy access to the assigned IDs.

5. Classes (Clock, Record, music_channel, Dorm):

- Purpose: Classes are used to encapsulate related data and functionality into objects, providing a modular and organized way to represent entities such as time, student records, music channels, and dormitories.
- Why: Classes help in modeling real-world entities and their interactions by bundling related data and methods together. They offer encapsulation, inheritance, and polymorphism, enabling better code organization, reusability, and maintainability.

Functions And their Use

Below are the functions used in the provided code along with their purposes:

1. insert_details(vector<Record>& records):

- Purpose: Reads student records from an input file and populates the vector of records.
- Use: It allows the program to initialize the system with student details, including their names, AirPod IDs, sleep times, time taken to play music, and song names.

2. assign_to_dorms(vector<Record>& records, vector<Dorm>& dorms):

- Purpose: Assigns students to dormitories based on their sleep schedules, ensuring a gap of at least 1 hour between students in the same dormitory.
- Use: This function organizes students into dormitories, optimizing their placements to minimize disturbances during sleep hours.

3. sleep_inducer(vector<Dorm>& dorms):

- Purpose: Simulates playing sleep-inducing music in dormitories during scheduled sleep times.
- Use: This function creates a conducive environment for students to relax and fall asleep by playing soothing music, tailored to their preferences, during their designated sleep hours.

4. print_dorm_details(vector<Dorm>& dorms):

- Purpose: Prints details of dormitories including assigned inmates and music channel information.
- Use: This function allows users to view the current occupancy of dormitories and the assignment of inmates, facilitating monitoring and management of dormitory resources.

These functions collectively enable the sleep inducer to initialize with student records, assign students to dormitories, simulate sleep induction, and provide user interaction for viewing dormitory details or executing the sleep inducer. Each function serves a specific purpose in the overall functionality and management of the system.

Time and Space Complexity

Here's the detailed time and space complexity analysis for the provided code:

1. Time Complexity:

- insert_details(vector<Record>& records):

- Time Complexity: O(N)
- Explanation: The time complexity of reading student records from the input file and populating the vector of records is linear with respect to the number of records in the file, denoted by N.

- assign_to_dorms(vector<Record>& records, vector<Dorm>& dorms):

- Time Complexity: O(N * M)
- Explanation: In the worst case, where each record needs to be assigned to a new dormitory, this function iterates over each record (N) and each dormitory (M), resulting in a time complexity of O(N * M).

- sleep_inducer(vector<Dorm>& dorms):

- Time Complexity: O(N * D * H * M)
- Explanation: Here, N represents the number of dormitories, D represents the average number of students per dormitory, H represents the number of hours simulated (24 in this case), and M represents the number of minutes per hour (60). The function iterates over each minute of each hour for each dormitory and checks if an inmate is scheduled to sleep, resulting in a time complexity of O(N * D * H * M).

- print_dorm_details(vector<Dorm>& dorms):

- Time Complexity: O(N * D)
- Explanation: This function iterates over each dormitory (N) and each inmate in each dormitory (D), resulting in a time complexity of O(N * D).

- Overall Time Complexity:

- Considering the worst-case scenario and summing up the time complexities of all functions:

- O(N) + O(N * M) + O(N * D * H * M) + O(N * D)
- This can be simplified to O(N * D * H * M), where N is the number of dormitories, D is the average number of students per dormitory, H is the number of hours simulated, and M is the number of minutes per hour.

2. Space Complexity:

- insert_details(vector<Record>& records:

- - Space Complexity: O(N)
- - Explanation: The space complexity of storing student records in the vector of records is linear with respect to the number of records, denoted by N.

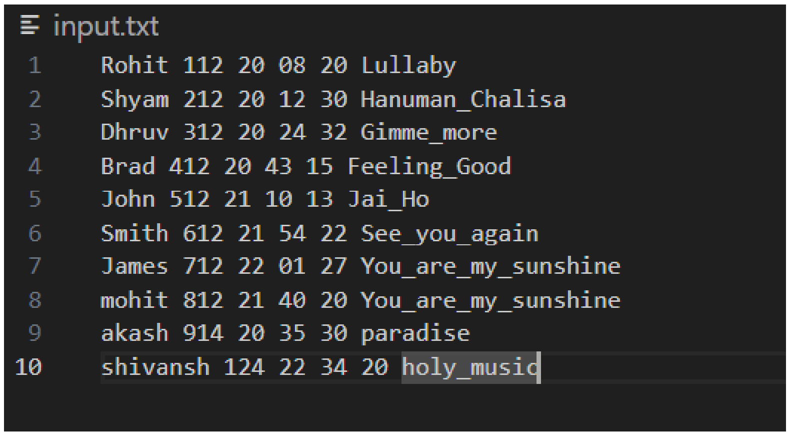
- assign_to_dorms(vector<Record>& records, vector<Dorm>& dorms:

- - Space Complexity: O(N + M)
- - Explanation: Additional space is required for storing assigned_names, which is a set containing unique student names. The space complexity of this set is O(M), where M is the total number of unique student names.

Overall Space Complexity:

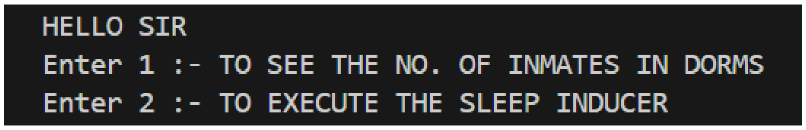
- Considering the space required for storing student records, dormitories, and additional data structures:
- O(N) + O(M)
- This can be simplified to O(N + M), where N is the number of records and M is the total number of unique student names.

Output



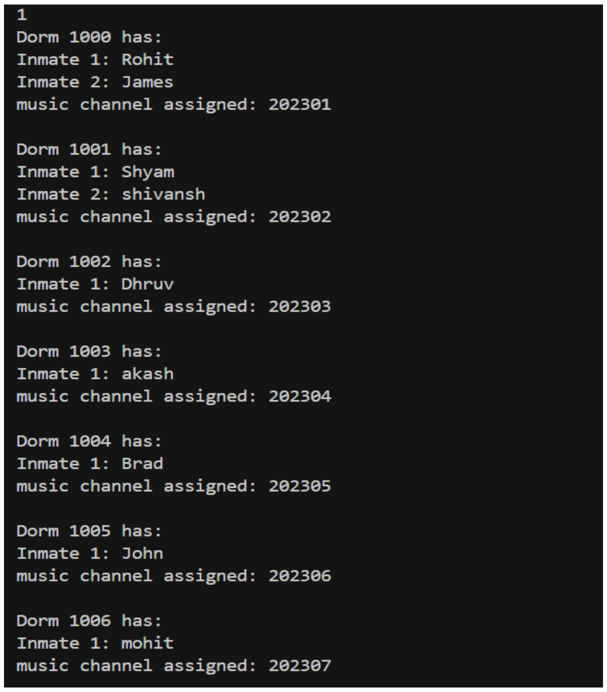
```
input.txt
Rohit 112 20 08 20 Lullaby
Shyam 212 20 12 30 Hanuman_Chalisa
Dhruv 312 20 24 32 Gimme_more
Brad 412 20 43 15 Feeling_Good
John 512 21 10 13 Jai_Ho
Smith 612 21 54 22 See_you_again
James 712 22 01 27 You_are_my_sunshine
mohit 812 21 40 20 You_are_my_sunshine
akash 914 20 35 30 paradise
shivansh 124 22 34 20 holy_music
```

These are the inputs given in the mentioned format.

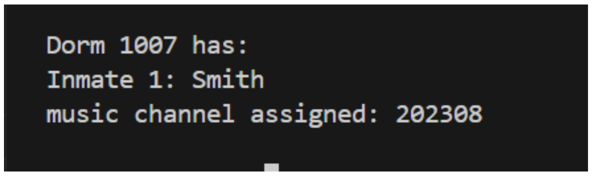


```
HELLO SIR
Enter 1 :- TO SEE THE NO. OF INMATES IN DORMS
Enter 2 :- TO EXECUTE THE SLEEP INDUCER
```

User Choice is required here.



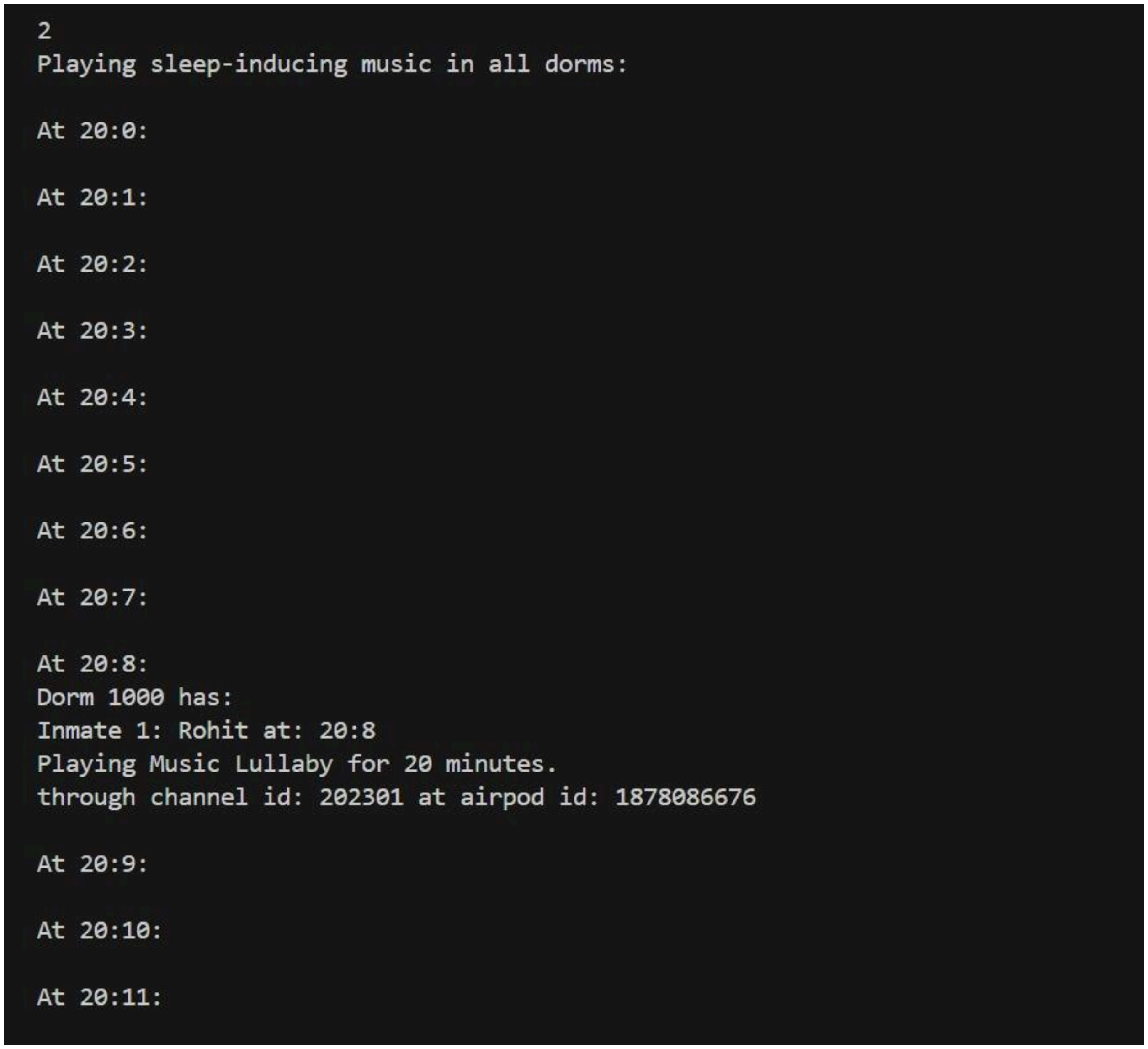
```
1
Dorm 1000 has:
Inmate 1: Rohit
Inmate 2: James
music channel assigned: 202301

Dorm 1001 has:
Inmate 1: Shyam
Inmate 2: shivansh
music channel assigned: 202302

Dorm 1002 has:
Inmate 1: Dhruv
music channel assigned: 202303

Dorm 1003 has:
Inmate 1: akash
music channel assigned: 202304

Dorm 1004 has:
Inmate 1: Brad
music channel assigned: 202305

Dorm 1005 has:
Inmate 1: John
music channel assigned: 202306

Dorm 1006 has:
Inmate 1: mohit
music channel assigned: 202307
```



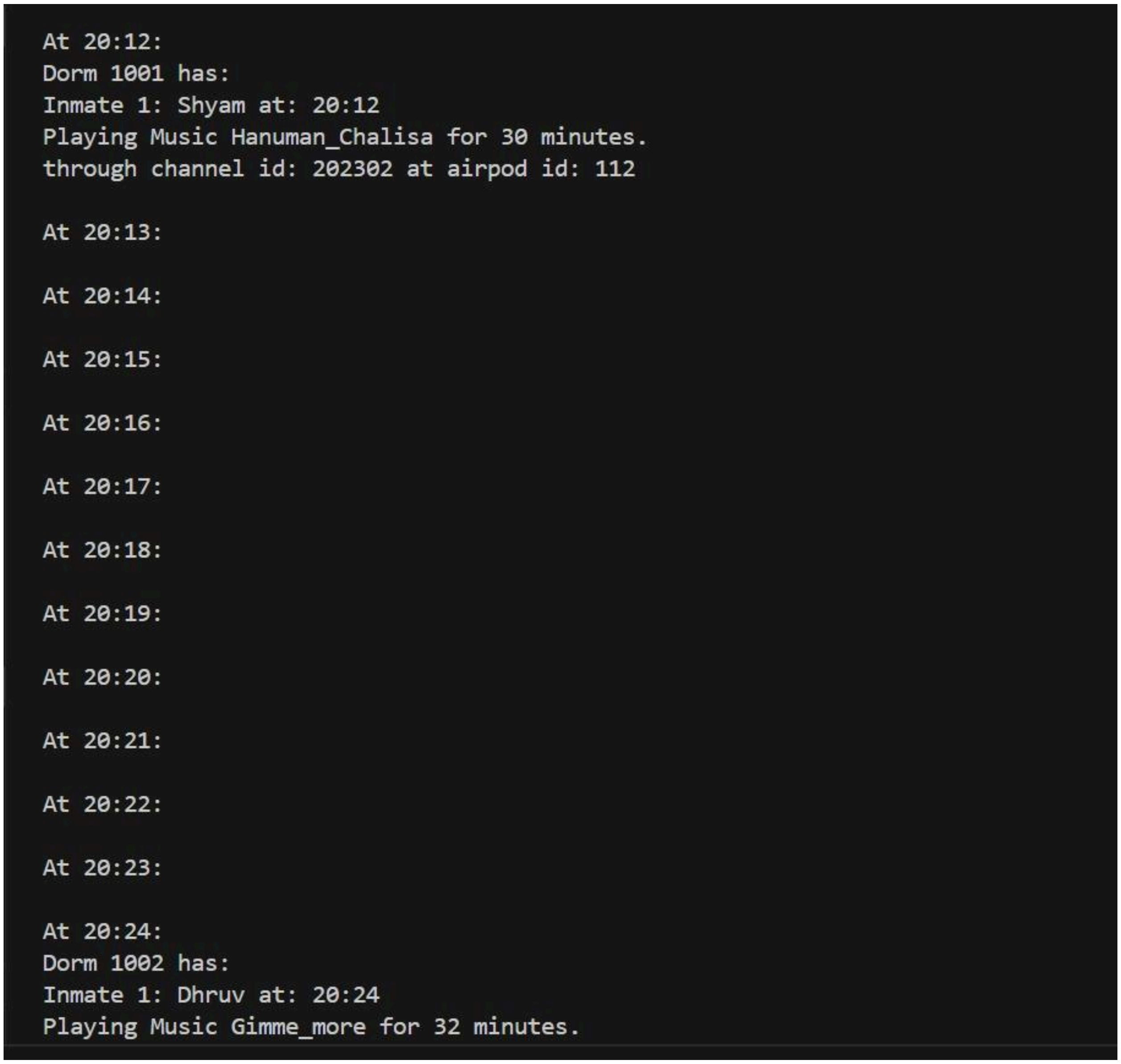
```
Dorm 1007 has:
Inmate 1: Smith
music channel assigned: 202308
```

When the 1st option is selected it gives the details of the dorm and the people in it.

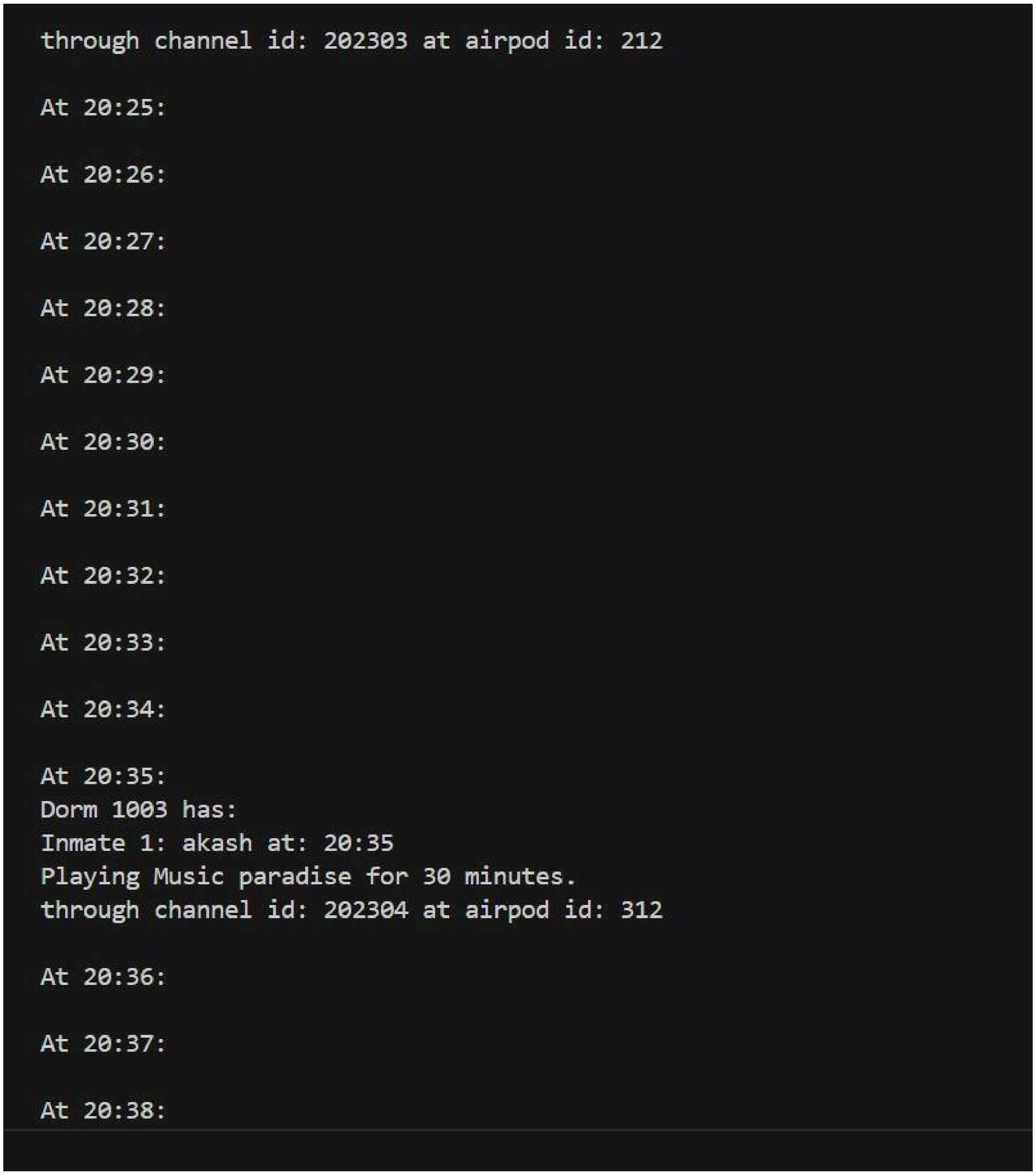
When the 2nd option is selected, the sleep inducer starts from 20:00 to 2:59 hrs.



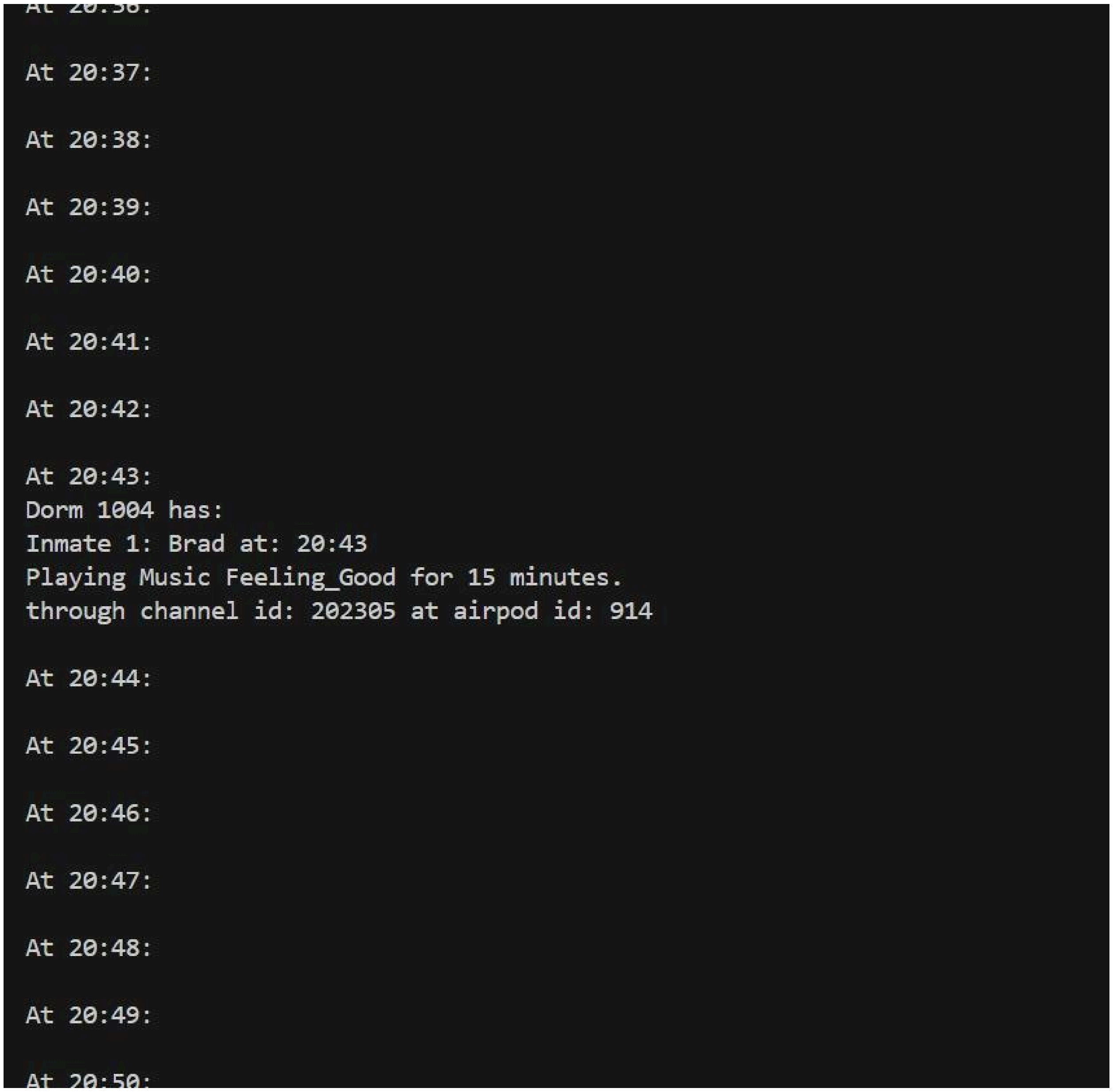
```
2
Playing sleep-inducing music in all dorms:

At 20:0:

At 20:1:

At 20:2:

At 20:3:

At 20:4:

At 20:5:

At 20:6:

At 20:7:

At 20:8:
Dorm 1000 has:
Inmate 1: Rohit at: 20:8
Playing Music Lullaby for 20 minutes.
through channel id: 202301 at airpod id: 1878086676

At 20:9:

At 20:10:

At 20:11:
```



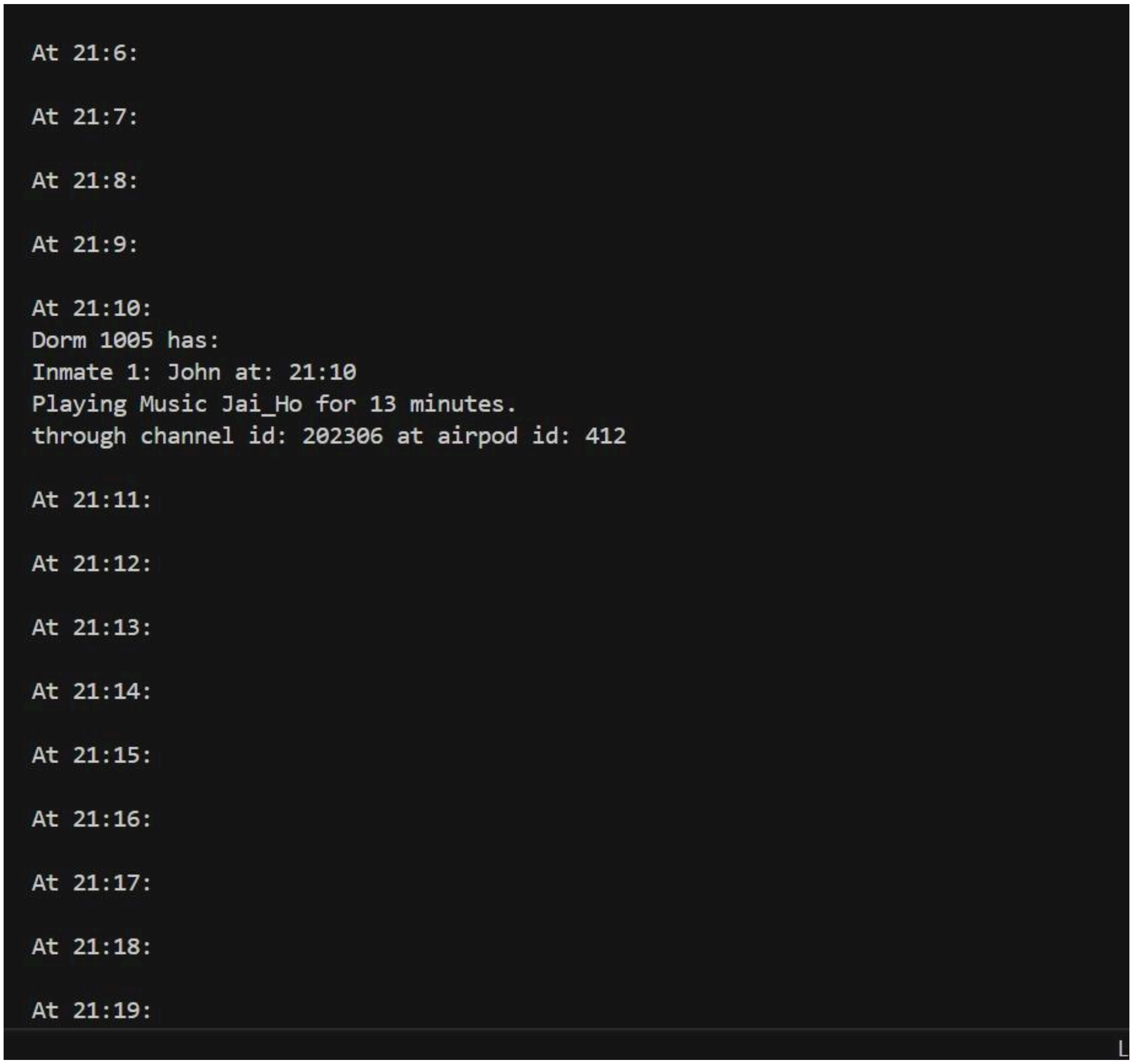
```
At 20:12:
Dorm 1001 has:
Inmate 1: Shyam at: 20:12
Playing Music Hanuman_Chalisa for 30 minutes.
through channel id: 202302 at airpod id: 112

At 20:13:

At 20:14:

At 20:15:

At 20:16:

At 20:17:

At 20:18:

At 20:19:

At 20:20:

At 20:21:

At 20:22:

At 20:23:

At 20:24:
Dorm 1002 has:
Inmate 1: Dhruv at: 20:24
Playing Music Gimme_more for 32 minutes.
```



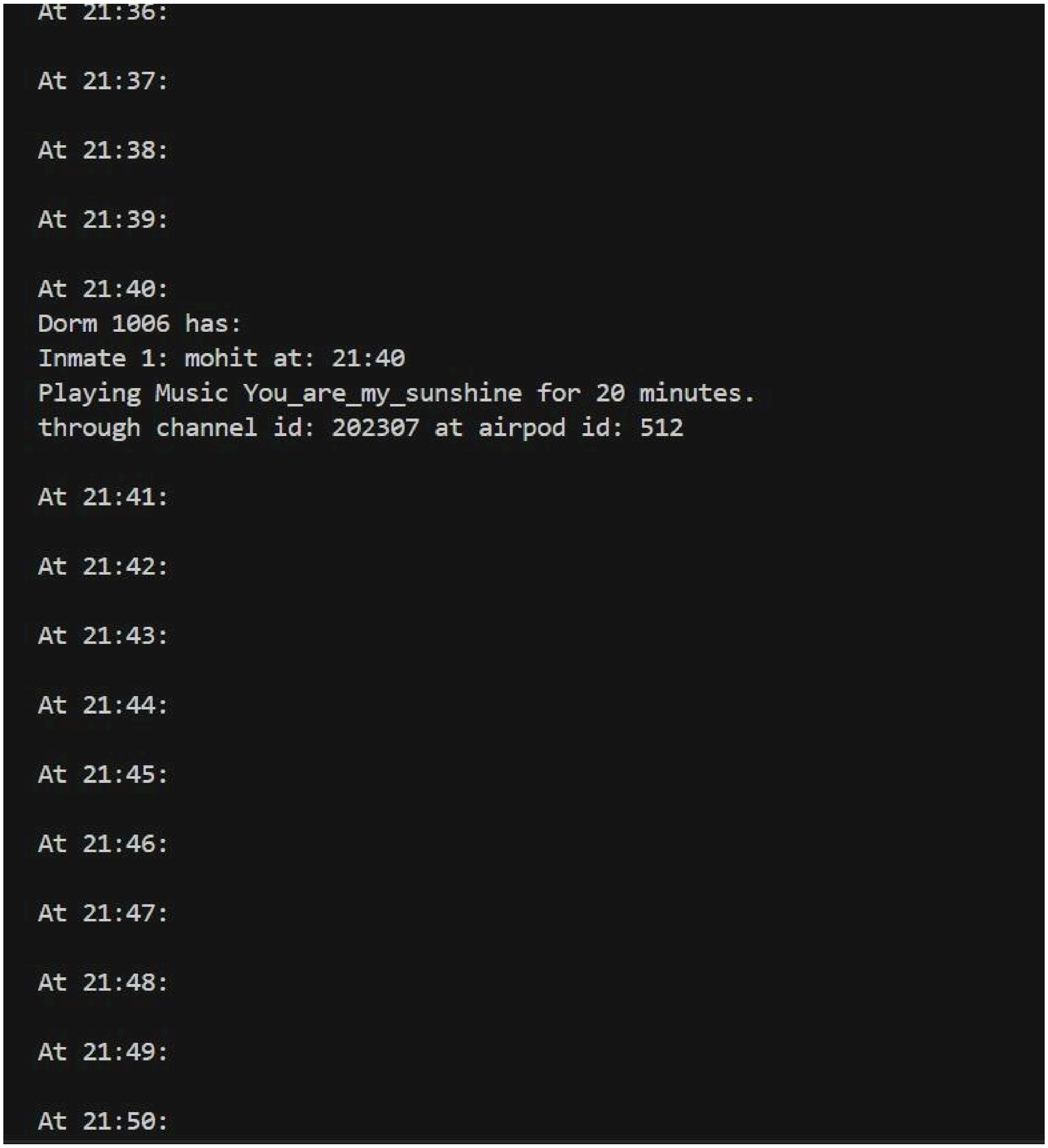
```
through channel id: 202303 at airpod id: 212

At 20:25:

At 20:26:

At 20:27:

At 20:28:

At 20:29:

At 20:30:

At 20:31:

At 20:32:

At 20:33:

At 20:34:

At 20:35:
Dorm 1003 has:
Inmate 1: akash at: 20:35
Playing Music paradise for 30 minutes.
through channel id: 202304 at airpod id: 312

At 20:36:

At 20:37:

At 20:38:
```



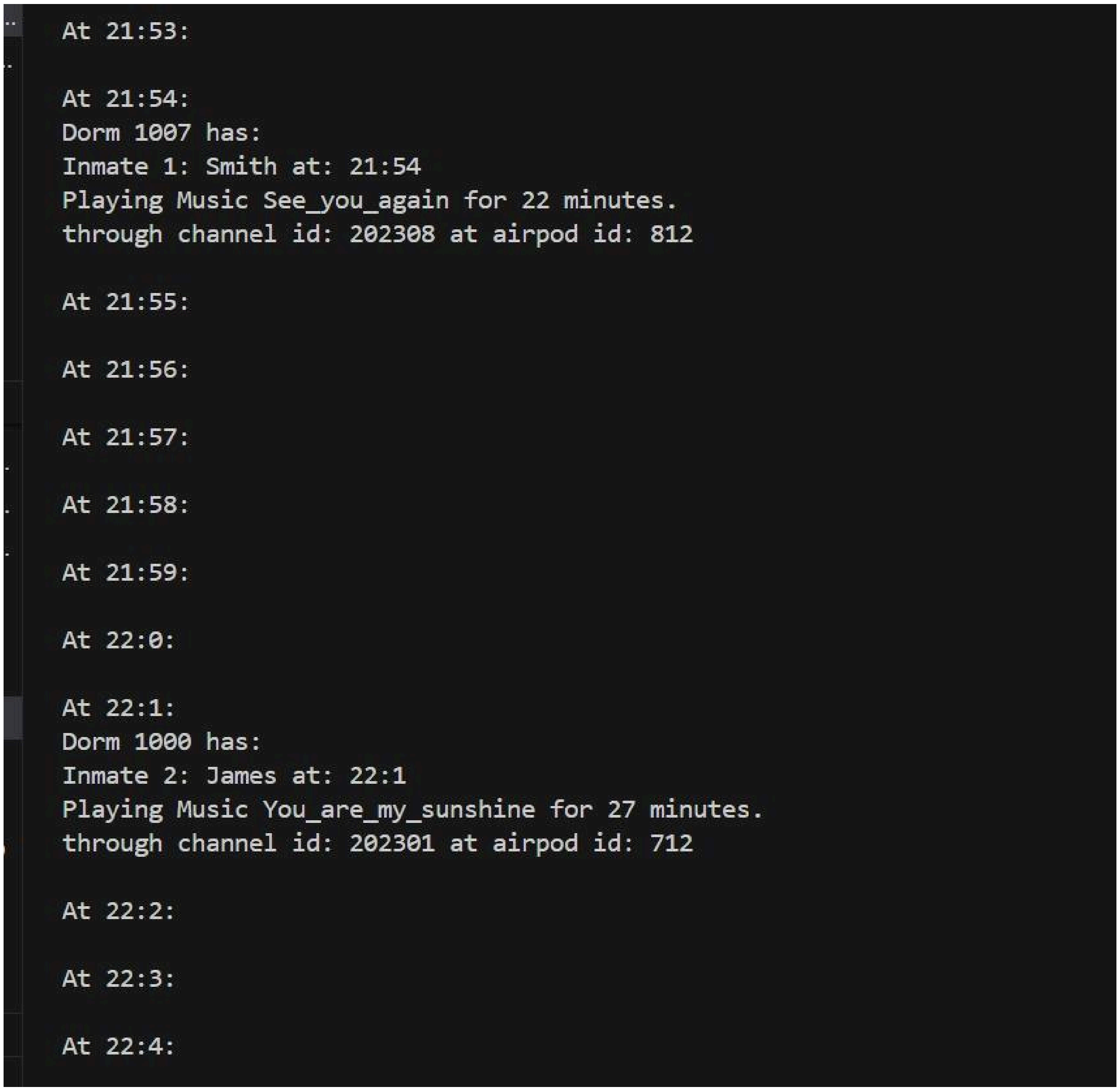
```
At 20:37:

At 20:38:

At 20:39:

At 20:40:

At 20:41:

At 20:42:

At 20:43:
Dorm 1004 has:
Inmate 1: Brad at: 20:43
Playing Music Feeling_Good for 15 minutes.
through channel id: 202305 at airpod id: 914

At 20:44:

At 20:45:

At 20:46:

At 20:47:

At 20:48:

At 20:49:
```



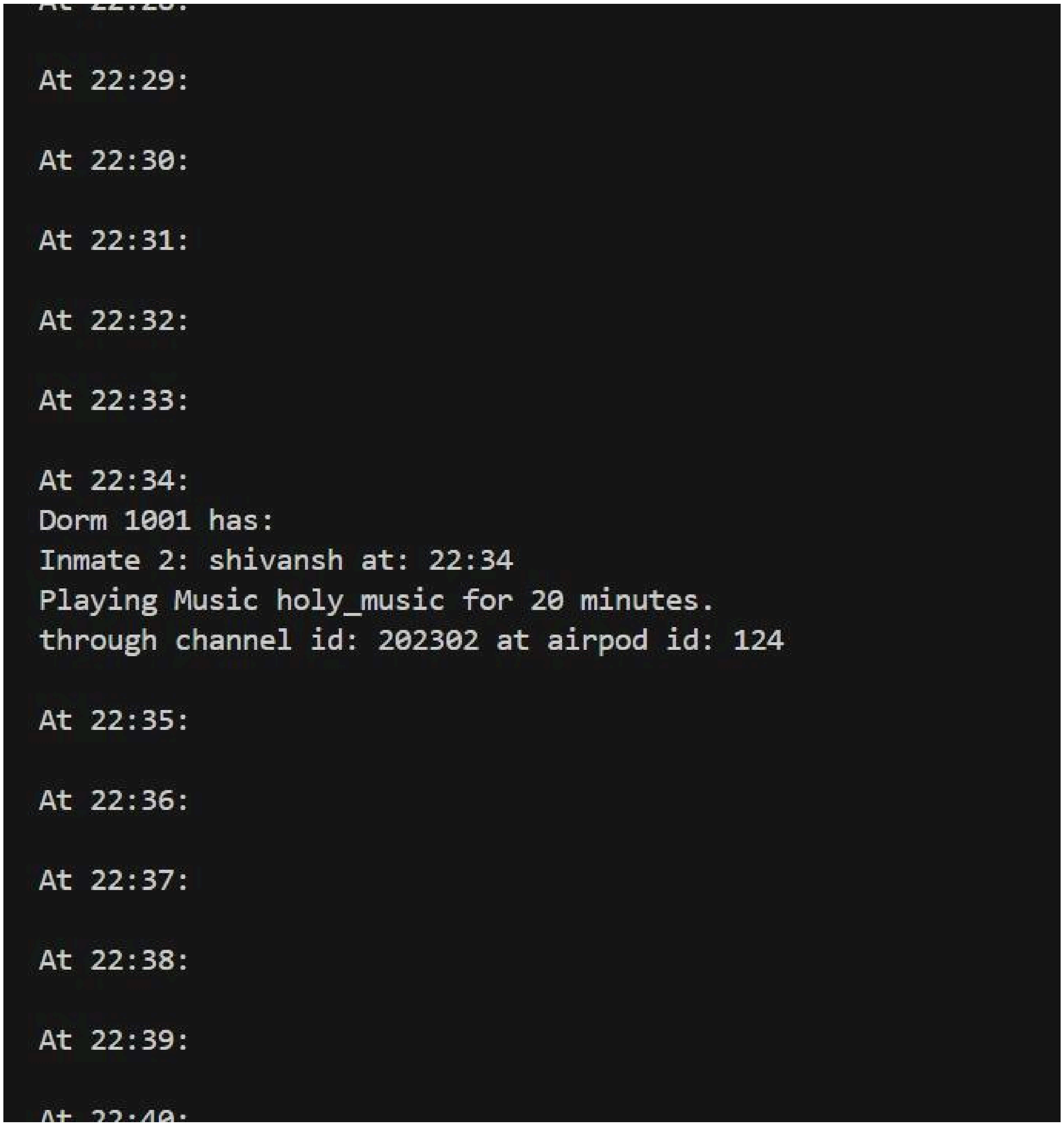
```
At 21:6:

At 21:7:

At 21:8:

At 21:9:

At 21:10:
Dorm 1005 has:
Inmate 1: John at: 21:10
Playing Music Jai_Ho for 13 minutes.
through channel id: 202306 at airpod id: 412

At 21:11:

At 21:12:

At 21:13:

At 21:14:

At 21:15:

At 21:16:

At 21:17:

At 21:18:

At 21:19:
```



```
At 21:36:

At 21:37:

At 21:38:

At 21:39:

At 21:40:
Dorm 1006 has:
Inmate 1: mohit at: 21:40
Playing Music You_are_my_sunshine for 20 minutes.
through channel id: 202307 at airpod id: 512

At 21:41:

At 21:42:

At 21:43:

At 21:44:

At 21:45:

At 21:46:

At 21:47:

At 21:48:

At 21:49:

At 21:50:
```



```
At 21:53:

At 21:54:
Dorm 1007 has:
Inmate 1: Smith at: 21:54
Playing Music See_you_again for 22 minutes.
through channel id: 202308 at airpod id: 812

At 21:55:

At 21:56:

At 21:57:

At 21:58:

At 21:59:

At 22:0:

At 22:1:
Dorm 1000 has:
Inmate 2: James at: 22:1
Playing Music You_are_my_sunshine for 27 minutes.
through channel id: 202301 at airpod id: 712

At 22:2:

At 22:3:

At 22:4:
```



```
At 22:29:

At 22:30:

At 22:31:

At 22:32:

At 22:33:

At 22:34:
Dorm 1001 has:
Inmate 2: shivansh at: 22:34
Playing Music holy_music for 20 minutes.
through channel id: 202302 at airpod id: 124

At 22:35:

At 22:36:

At 22:37:

At 22:38:

At 22:39:
```

References

[C++ Arrays - GeeksforGeeks]

C++ Classes and Objects - GeeksforGeeks
Unordered Sets in C++ Standard Template Library - GeeksforGeeks
Strings in C++ - GeeksforGeeks
Vector in C++ STL - GeeksforGeeks