Apexcoderz

Group Members:-

1. Om Santoki - 202301019
2. Parva Raval - 202301055
3. Diyen Pambhar - 202301113
4. Daksh Ubhadia- 202301014

Github Link:-

https://github.com/omsantoki/Capstone-Apexcoderz

P7 Sleep Inducer:

You are to build a Sleep Inducer. There are *N* inmates in total. Each has 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. However, the music should not continue beyond *p* minutes after going to bed (assuming it takes that much time for every inmate to fall asleep). The Sleep Inducer should be able to play for all of them without disturbing anyone (i.e., playing music while someone is asleep). Therefore, the correct number of inmates for each dorm must be assigned before starting to operate.

Pseudo Code

BEGIN

Include libraries:

-iostream

-fstream

-sstream

-string

-vector

-unordered_map

-ctime

-conio.h

-windows.h

-mmsystem.h

-iomanip

define global variables:

-filename: string

define class Inmates:

-private members: name (string),ID (string),dorm (int)

-public methods:

-constructor: Inmates()

-setter methods: setname(string), setID(string), setdorm(int)

-getter methods: getname(), getID(), getdorm()

define function ifexist(ID: string) -> bool:

-open file specified by Filename in read mode.

-iterate through each line in the file:

-check if the ID exists in the line.

-if found, return true.

-if not found, return false.

define function insert(Inmates i):

-initialize variables: close = false, filex = true

-while not close:

-clear screen

-prompt user to enter input file name if filex is true

-prompt user to enter inmate's ID

-if the ID already exists:

-print error message and wait for 2 seconds

-else:

-prompt user to enter inmate's name

-open file specified by Filename in append mode

-iff file cannot be opened:

-print error message and return

-else:

-prompt user to enter sleep time for 7 days

-calculate average sleep time

-determine dorm number based on average sleep time

-write inmate's data to the file

-close the file

-print success message

-prompt user for further actions (add more data, change file, close)

define function search(Inmates i):

-clear screen

-get current time

-prompt user to enter inmate's ID

-open file specified by filename in read mode

-if file cannot be opened:

-print error message and return

-else:

-initialize an unordered_map to store inmate data

-iterate through each line in the file:

-parse the line and extract dorm, ID, name, and sleep time data

-store inmate's data in the unordered_map

-close the file

-search for the given ID in the unordered_map

-if ID is found:

-print inmates's details and wait for Enter key press

- ese:

-print error message

define function show():

-open file specified by Filename in read mode

-if file cannot be opened:

-print error message and return

-else:

-get current day

-print header for inmate details

-iterate through each line in the file:

-parse the line and extract dorm, ID, name, and sleep time data

-print inmate's details

-wait for Enter key press

Define function delete_inmate():

-initialize close = false

-while not close:

-clear screen

-prompt user to enter input file name

-prompt user to enter inmate's ID to delete

-open file specified by Filename in read mode

-initialize found = false

-initialize a vector to store lines of the file

-iterate through each line in the file:

-check if the inmate's ID exists in the line

-if found, mark found as true, else add the line to the vector

-close the file

-open file specified by Filename in write mode

-write lines from the vector to the file

-close the file

-if inmate is found:

-print success message

-else:

-print error message and prompt user for further actions

define function set_dorm():

-clear screen

-prompt user to enter input file name

-prompt user to enter day

-open file specified by Filename in read mode

-initialize an array with day names

-reset dorm numbers based on day and sleep time

-close the file

-print wait message and wait for 3 seconds

define function inducer():

-prompt user to enter input file name

-print message to exit

-while no key is pressed:

-get current time

-determine dorm to induce sleep based on time

-print dorm number and Earpod IDs of inmates in that dorm

-play corresponding music

-print current time and refresh every second

define main function:

-initialize object of Inmates class

-initialize exit = false

-while not exit:

-clear screen

-print menu options

-prompt user for choice

-based on choice, call respective functions or exit program

END

Use of Data Structures in the Code:-

Following is the list of the data structures that we have used in our project.

1. Arrays

   Arrays are one of the fundamental data structures in programming languages. In arrays we store the data of the same data type in contiguous memory locations.

   In our code, we used two arrays H[] and M[] to store the hours and minutes of sleep for each day. This helps in accessing the sleep time data efficiently for each day of the week, and thus helps in calculating mean sleep time and other sleep pattern calculations.

2. Vectors

   Vectors are simply dynamic arrays that can resize whenever new elements are added or existing elements are deleted.

   In our code, we have used vectors in the delete_inmate function while reading lines from the input text file.Thus using the vectors helps in managing the data flexibly, as vectors give suitable methods to add,remove and access elements.

3. Pairs

   Pairs are a composite data structure that allows the collection of multiple elements of different data types into a single entity.

   We have used this data structure to store the information of each inmates, i.e. their names, dorm and sleep time, as shown:-

   unordered_map<string, pair<pair<string, string>, pair<string,string>>> inmates;

   Here variable inmates are initialized with data type unordered maps. The key is a string that represents the ids of the inmates.The value of this pair itself is a pair of nested 2 inner pairs.The first inner pair stores the name and dorm of the inmates.While the second inner pair stores the hours and minutes of sleep time of the inmates.

4. Hash Tables(Unordered Maps)

   The hashing data structure is used in the code as it efficiently stores the information of inmates using their IDs. Here, constant-time complexity is observed for insertion and searching operations.

Time and Space Complexities of the code:-

Following are the time and space complexities of each function of our project

Note that n is the number of lines in the input file

1) ifexist() Function

Time Complexity- $O(n)$

Because it iterates through each line in the text file to check if given id exists

Space Complexity- $O(1)$

Because constant amount of extra memory is used irrespective of the input done before that

2) insert Function

Time Complexity-$O(1)$ for each insertion

Space Complexity: $O(1)$ for each insertion

3) search Function:

Time Complexity-$O(k)$

where k is the max number of collision encountered during the probing process.Thus making the use of hash table efficient in the project

Space Complexity-$O(n)$

Because this function stores information of each inmate in the inmates map.

4) show Function:

Time Complexity-$O(n)$

Because the code needs to iterate through each line to display the information of each inmate.

Space Complexity-O(1)

Because no new data structure is used that grows with the input size.

5) delete Function:

Time Complexity-O(n)

As code iterates through each line of the file to find and delete the specified inmate.

Space Complexity-O(n)

Because the function may need to temporarily store the contents of the file in memory while removing the specified inmate.

6) Set Dorm Function:

Time Complexity-O(n)

Because the code iterates through each line of the file to update dorm information.

Space Complexity: O(n)

Because it may need to temporarily store the file data in memory while updating dorm information.

7) Inducer Function:

Time Complexity-O(n)

Because the code iterates through each line of the file to find and display information about inmates belonging to specific dorms.

Space Complexity-O(1)

Because the function does not use any new data structures that grows with the input size.

Code Explanation:-

The code chiefly focuses on allotting the inmates their respective dorms depending on the respective day of the week. The prime feature of our project is the simulation of the sleep inducer application(playing of songs in respective dorms) in a terminal based environment.Note that we have taken p as 30 mins in our project.

Firstly, we are taking the inputs from the user in form of the text file as well as the user has the availability to add new inmates at the end of the file.This is done in the insert function. We are also checking if the inmate by the same ID is already existing or not.Along with the input of sleep time of each inmate, we are also calculating the average sleep time.At the end,we ask if user wants to add more data in same file or whether the user has some other list of inputs from other file or user finally wants to exit the insertion process.

Next, the user can search for any user from the data that is currently existing. For this, we have used the hash table data structure for efficiency as discussed above.If the inmate is found, then their ID, Name, Dorm No and Sleep Time of that particular day are shown.

The Show function, as the name suggests, shows the details, viz. Name, ID, Sleep Time and Dorm No. of all the inmates that were stored in the input file are shown.

Just as we allow the user to insert new inmates, we are providing a Delete function to delete the existing inmates.Then comes one of the main function of the project named set_dorm. In this function first we take input of the file of record of all the inmates and then we input the day we want as the current day. For example if the day entered is Monday, then the first sleeping time of all the inmates is observed and then we allocate them their dorms accordingly.

Chiefly dorm allocation is on the following basis:-

1)Dorm 1 for those inmates whose sleep time for that day is between 9:00PM to 9:30PM

2)Dorm 2 for those inmates whose sleep time for that day is between 9:30PM to 10:00PM

3)Dorm 3 for those inmates whose sleep time for that day is between 10:00PM to 10:30PM

4)Dorm 4 for those inmates whose sleep time for that day is between 10:30PM to 11:00PM

5)Dorm 5 for those inmates whose sleep time for that day is between 11:00PM to 11:30PM

6)Dorm 6 for those inmates whose sleep time for that day is between 11:30PM to 12:00AM

Next the final function is the pivotal aspect of the project, i.e. the inducer function. Here the conditions of current time are matched and accordingly we enter the active dorm. On entering the active dorm, the music plays continuously for the respective half hour(as p is assumed 30 for all in our project).

Photos and video of the output:-

https://drive.google.com/drive/folders/1H3lKQw9vbW8MKKgJzrho3a83l-qZNhfc?usp=share_link