Capstone (IT205: Data Structures)

- **GitHub repository Link :**
  GitHub(Team-Code_Sorcerers)
- **Team Name :** Code_Sorcerers
- **Team Members :**
  1. Tushal Mendpara ID : 202301141
  2. Darshan Ramani ID : 202301161
  3. Manav Patel ID : 202301158
  4. Ved Donda ID : 202301186

- **Project title and description :**

  P6. DA-IICT Club Manager

  Problem:

  You need to build a manager for all the DA-IICT clubs. The manager ensures that a club member can be looked up in minimum time. A member can either be a faculty or a student One should be able to search by name, ID, specific club name, or club category (i.e., arts, science & technology, sports, culture). Note that the user of this manager may not be a DA-IICT-ian and, therefore, may not know the clubs names.

❖ **Password : Daiict@2023 (to use add & remove functions)**

[**Pseudocode:** given in separate file in github repository]

(i) The algorithm :

- First we have defined the ClubCategory enumeration and Member and club structures and initialized member_hash_table and club_hash_table to store member and club information respectively.

- Then we have defined functions for adding a new member (add_member), adding a member in file (add_member_in_file), converting club category to string (str_category), adding a new club (add_new_club), searching members by club name (search_by_club_name), searching members by name (search_by_member_name), searching members by ID (search_by_id), searching members by club category (search_by_club_category), removing a member by ID (remove_member_by_id), printing club categories (print_category), reading club members details from a file (read_ClubMembersDetails), and reading club categories from a file (read_ClubCategories).

- After that in the main function, we have called functions read_ClubCategories and read_ClubMembersDetails to read club categories and club members details from CSV files. Then, we have provided a menu for the user to select an operation (search by club name, member name, member ID, club category, add new club, add

new member, remove member by ID, or exit). After that it performs the selected operation and repeats until the user chooses to exit.

- More detail on every individual function and its algorithm is as given below.
- search_by_club_name() (function to search members by club name)
  - Input Parameters : club_name
  - Check if the club exists in the club_hash_table.
  - If not, display a "Club not found" message and return.
  - Otherwise, iterate over the members of the club:
    - Display each member's name and student ID.
  - If the club has no members, display a message indicating this.
  - End
- search_by_member_name() (function to search member by name)
  - Input Parameters : member_name
  - Search for the member in the member_hash_table using their name.
  - If found:
    - Display the member's name and student ID.
    - Display the clubs the member belongs to.
  - If not found, display a "Member not found" message.
  - End
- search_by_id() (function to search member by ID)
  - Input Parameters : student_id
  - Iterate over each member in the member_hash_table.

- ➢ If a member has the provided student ID:
  - ■ Display the member's name and student ID.
  - ■ Display the clubs the member belongs to.
- ➢ If no member with the provided ID is found, display a message indicating this.
- ➢ End

❖ search_by_club_category() (function to search all members associated with given club category)

- ➢ Input Parameters : category
- ➢ Print the list of available club categories.
- ➢ Prompt the user to enter the category number.
- ➢ Convert the user's input into a ClubCategory enum value.
- ➢ Check if the entered category is valid.
  - ■ If the category is invalid, print a message and return.
  - ■ If the category is valid, print the selected category.
- ➢ Set a flag to track if any clubs with members are found in the selected category.
- ➢ Iterate through each club in the club_hash_table.
- ➢ For each club:
  - ■ Check if the club belongs to the selected category.
  - ■ If it does, set the flag to true and call search_by_club_name to print the club's members.
- ➢ End

❖ remove_member_by_id() (function to remove member details from file and maps)

- ➢ Input Parameters : student_id
- ➢ Open the "ClubMembersDetails.csv" file for reading.

- ➢ Open a new file for writing.
- ➢ Read each line from the original file:
  - ■ If the student ID matches the provided ID, skip that line.
  - ■ Otherwise, write the line to the new file.
- ➢ Close both files.
- ➢ Delete the original file and rename the new file to the original file's name.
- ➢ If the member was found and removed:
  - ■ Remove the member's name from the club memberships in club_hash_table.
  - ■ Remove the member from member_hash_table.
- ➢ Display a success or failure message based on whether the member was found and removed.
- ➢ End

❖ add_member() (function to store member and clubs in map)

- ➢ Input Parameters : name, student_id, clubs
- ➢ Create a Member structure variable and set its attributes with the provided data.
- ➢ Add this member to the member_hash_table using their name as the key.
- ➢ For each club the member is part of:
  - ■ Check if the club already exists in the club_hash_table.
  - ■ If yes, add the member's name to the club's list members.
- ➢ End

❖ add_member_in_file() (function to store member details in .csv file)

- ➢ Input Parameters : name, student_id, clubs
- ➢ Open the "ClubMembersDetails.csv" file in append mode.

- ➢ If the file fails to open, display an error message and return.
- ➢ Write the member's name and ID to the file.
- ➢ For each club the member is part of:
  - ■ Write the club name to the file.
- ➢ Close the file.
- ➢ Display a success message.
- ➢ End

❖ add_new_club() (function to add new club in ClubCategoriesDetails.csv file)

- ➢ Input Parameters : club_name (name of the new club), category (category of the new club)
- ➢ Open the ClubCategoriesDetails.csv file in append mode.
- ➢ If the file fails to open, display an error message and return.
- ➢ Write the club_name and its corresponding category to the file.
- ➢ Close the file.
- ➢ Add the club to the club_hash_table with its category.
- ➢ Display a success message.
- ➢ End

❖ read_ClubMembersDetails() (function to read club member details from .csv file)

- ➢ Input Parameters : filename (ClubMembersDetails.csv)
- ➢ Open the ClubMembersDetails.csv file for reading.
- ➢ If the file fails to open, display an error message and return.
- ➢ Read each line from the file:
  - ■ Parse the line to extract member details (name, student ID, clubs).

        - Add the member to the member_hash_table using their name as the key.
    - Close the file.
    - End

- read_ClubCategories() (function to read club names and their categories from .csv file)
    - Input Parameters : filename (ClubCategoriesDetails.csv)
    - Open the ClubCategoriesDetails.csv file for reading.
    - If the file fails to open, display an error message and return.
    - Read each line from the file:
        - Parse the line to extract club category details (club name, category).
        - Determine the category enum value corresponding to the category string.
        - Add the club to the club_hash_table with its category.
    - Close the file.
    - Print the categories along with their corresponding clubs:
        - Iterate over each entry in the club_hash_table.
        - If the category has not been printed before:
            - Print the category name.
            - Print the clubs belonging to that category.
    - End
- printclubandcategories() (function to print all clubs and categories)
    - Print a header indicating the beginning of the category and club listing.
    - Initialize an unordered set to keep track of printed categories.

- ➢ Iterate through each entry in the club_hash_table.
- ➢ For each category entry
- ➢ Check if the category has been printed already.
- ➢ If not printed, print the category and its corresponding clubs.
- ➢ Mark the category as printed.
- ➢ End

❖ view_all_members_by_club_and_category() (function to print all member's details club wise)

- ➢ Iterate through each club category.
- ➢ For each category:
  - ■ Convert the category to its corresponding string representation.
  - ■ Print the category header.
  - ■ Iterate through each club in the category.
  - ■ For each club:
    - Print the members of the club using the search_by_club_name function.
    - Print a newline after printing all the clubs in the category.
- ➢ End

(ii)Explanations of why we chose the specific data structures (w.r.t time and space complexity)

- We used Hash Table [unorderd_map in c++] data structure as our main data structure to store details of members and clubs from excel files.

- Hash Table is here best choice as data structure according to us because Using an unordered map provides constant-time average complexity for insertions, deletions, and searches, making it efficient for storing and getting member information. Here searching is our main priority as given in the problem. Explanation of data structure and its complexity is given below.

- In the programme we have used two hash table (unordered_map) one to store details specified for members where member name is the key and details stored with structure including member name,Id,list of clubs & the other one for classification of clubs with details of each club i.e. category and list of members.

❖ Next : Description on Time & Space Complexity

Time Complexity

❖ **FUNCTION WISE**

- **search_by_club_name** : This function takes club name as input and give list of members with their id. Searching heapens in O(1) . And to print list of member it iterates through vector so there it would be O(m) where m is number of members. Id is given with **O(1)** time complexity because of member hash table.

- **search_by_member_name**: This function takes member name as input and with **O(1)** time complexity gives us Id and club through member hash table. Though it would take again O(m) to iterate through list of clubs.

- **search_by_id**: This function gives member name & clubs for given id by user.It iterates through member hash table until finds member with given id so it takes **O(n)** time complexity. { with help of one more hash table with Id as key we can do this in O(1) but it would require much more space.}

- **search_by_club_category**: This function gives the details of all the members who are in clubs under a given category by user. It requires to traverse through each member and look if he is in any club of that category which would require **O(n)** time complexity cause to know category of any club is work done inO(1).{again, this could be done with O(1) with more space taken}

BONUS FUNCTIONS

- **add_new_club** : O(1) to add in excel sheet & hash tables.
- **add_new_member** : O(1) to add in sheet & hash tables.
- **remove_member_by_id**: O(n) to iterate and find the member to delete and then delete.

Despite these functions programme has basic O(n) time complexity to read data from excel sheet by iterating and adding that to hash table.

Space Complexity

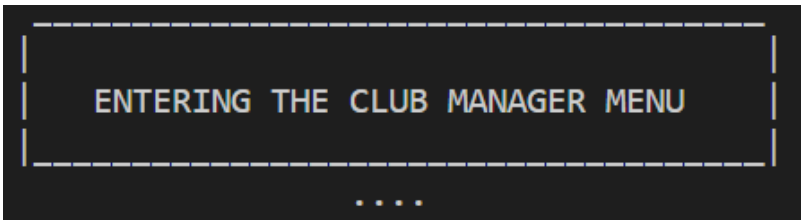
- Used Enum of categories to lessen space complexity
- **unordered_map<string, Member> member_hash_table**:

This data structure is used to store information about members. string is used as the key to store the member's name, and Member struct has details like name, student ID, and clubs. It requires **O**(**n**) space complexity where n is the number of members.

- **unordered_map<string, Member> member_hash_table**:

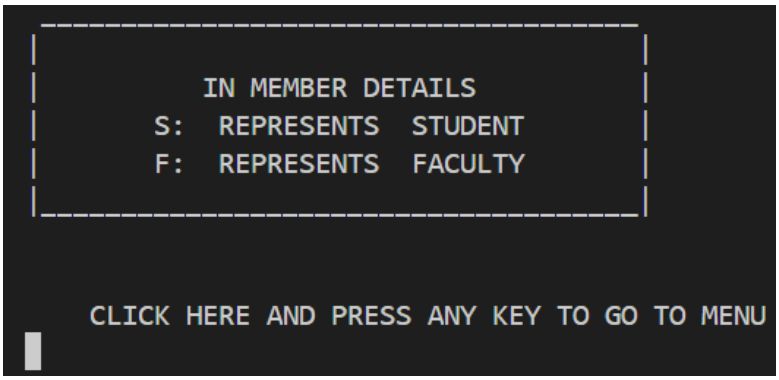
  This data structure is used to store information about clubs.here also string Club struct has details like its category,members and their id. Here **O**(**n**) space complexity where n is the number of clubs.

(iii) Example of solution :

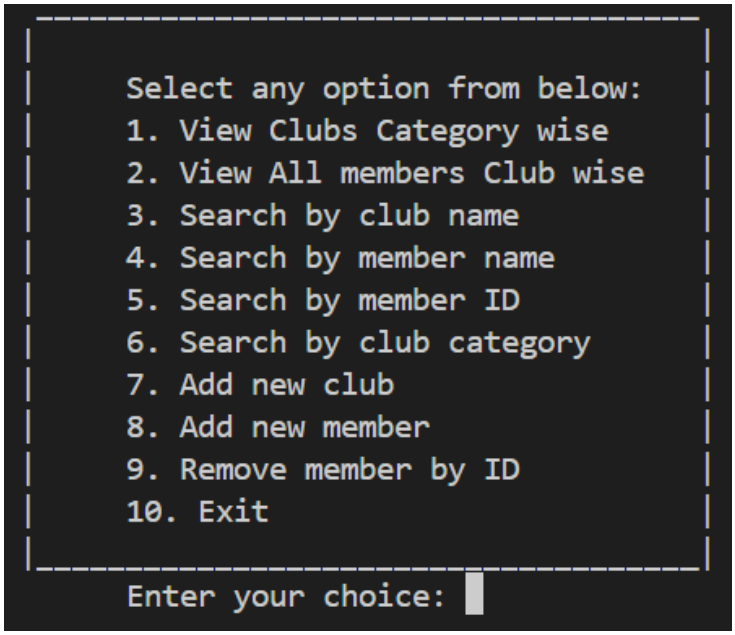


<---------->

➢ After this animation **First press any key** to enter in the program.



<---------->

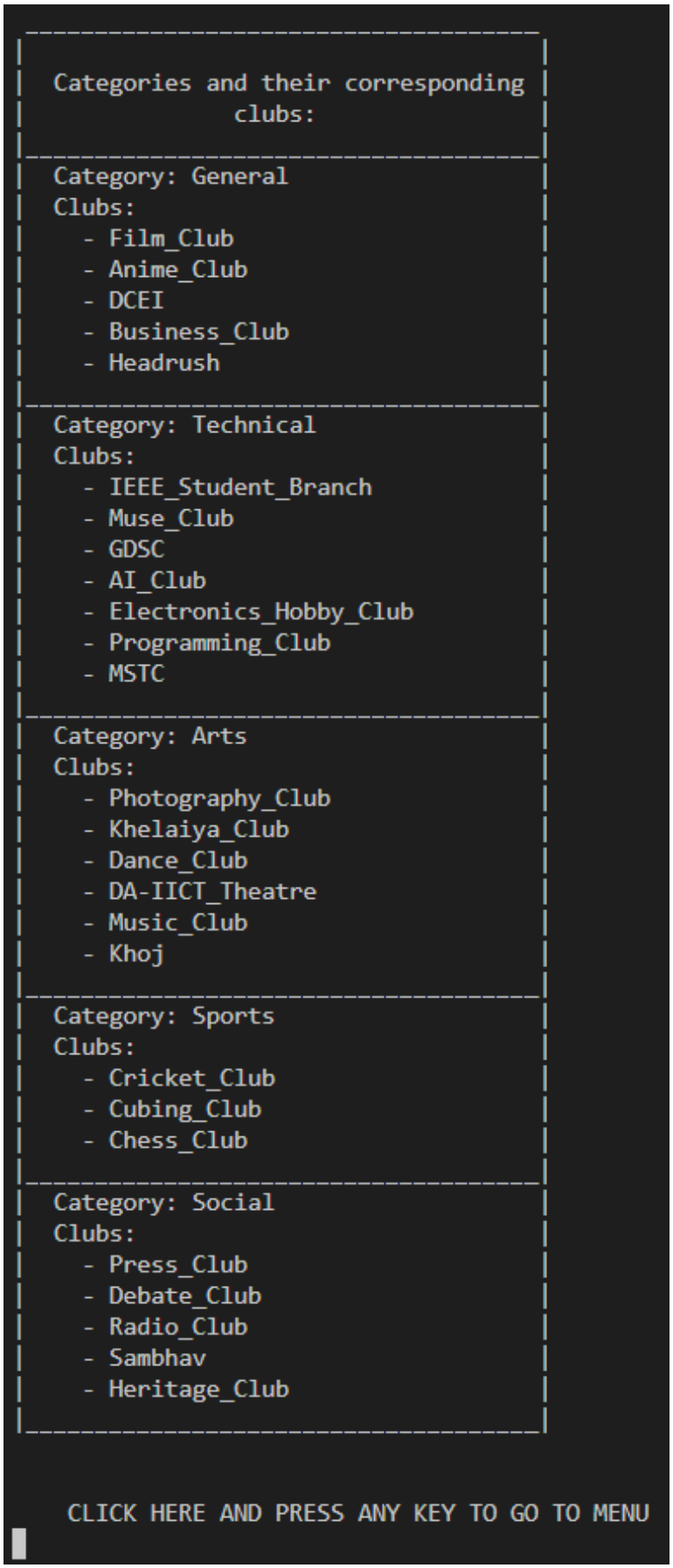


➢ As shown in the image **menu appear**.

➢ If **any person who doesn't know** anything about DA-IICT clubs or **any outside person** wants to know about all DA-IICT clubs then **enter 1** in the menu.

➢ After entering 1 to the menu all clubs appear category wise as shown in image.

➢ Then for go to the menu press **any key**.



➢ Now to know about **all members enter choice number 2** in the menu.

➢ After entering 2 to the menu all member lists appear club and category wise.

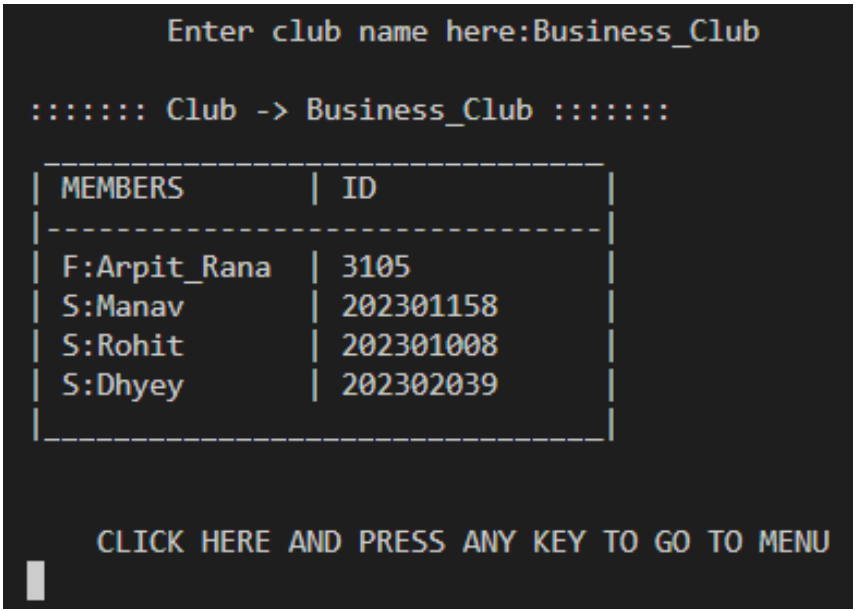
➢ Then for go to the menu press **any key**.

<---------->

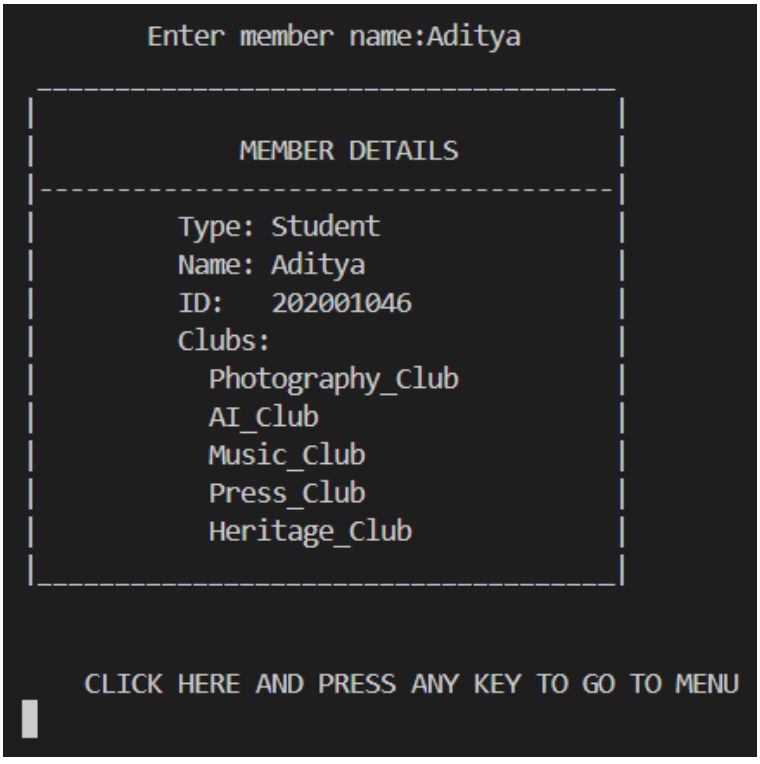
➢ For **search by club name** enter **choice number 3.**

➢ After entering any club name all member of that club shown as in image.

➢ Then for go to the menu press **any key**.



<---------->



➢ Now if you want to check any **member details by his/her name** then enter **choice number 4 in** the menu.

➢ After entering the name of a member , details of that member appear as shown in the image.

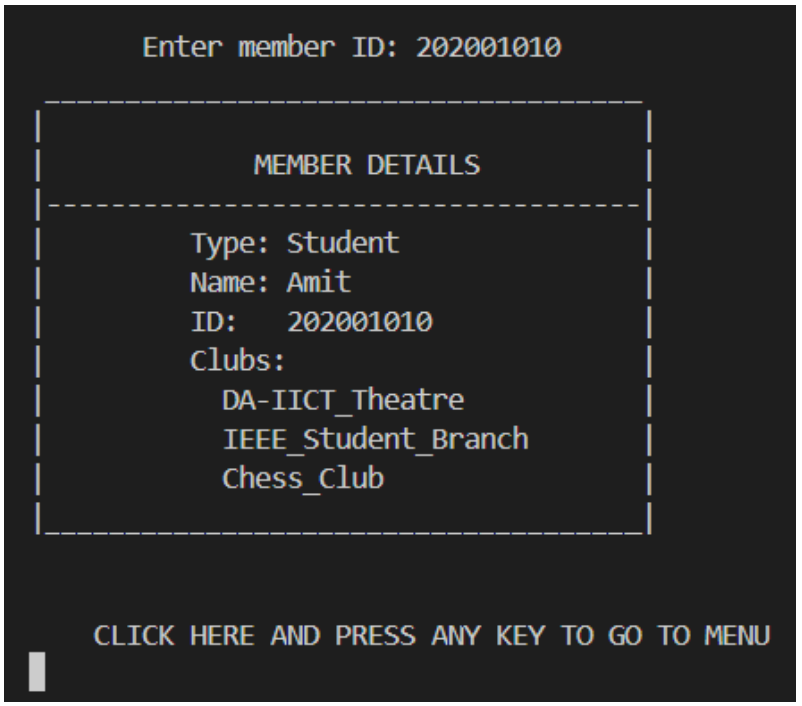
➢ Then for go to the menu press **any key**.

<--->

➢ If you want to **check any member details by his/her ID number** then enter **choice number 5** in the menu.

➢ After entering the ID number of member details of that member appear as shown in the image.

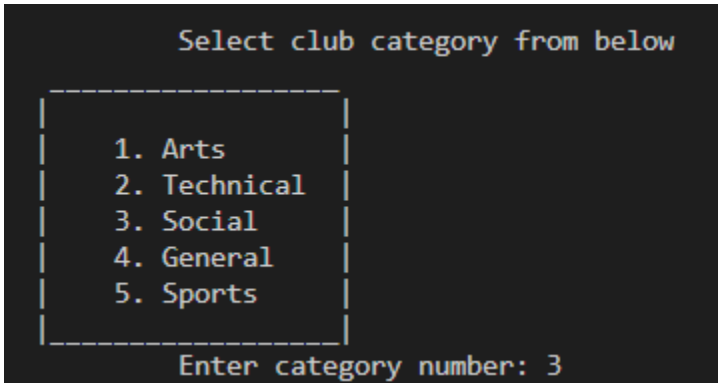
➢ Then for go to the menu press **any key.**



<--->

➢ If you want to **search any club category members** then enter **choice number 6** in the menu.

➢ Then after that enter the category number from the shown category list.



➢ After entering category number all members of that category appear club wise as shown in image.

➢ Then for go to the menu press **any key**.

```
<<===|||| Category => Social ||||===>>

::::::: Club -> Press_Club :::::::

 _________________________________
| MEMBERS        | ID             |
|________________|________________|
| F:Purbasha_Das | 4210           |
| S:Manav        | 202301158      |
| S:Aditya       | 202001046      |
| S:Bhavya       | 202303009      |
| S:Jenish       | 202301013      |
| S:Shamit       | 202112015      |
| S:Aryan        | 202121017      |
| S:Akshat       | 202101021      |
| S:Neev         | 202201031      |
| S:Jaimin       | 202201032      |
| S:Jaimin       | 202201032      |
| S:Parthiv      | 202303037      |
| S:Hemil        | 202303040      |
|________________|________________|

::::::: Club -> Debate_Club :::::::

 _________________________________
| MEMBERS        | ID             |
|________________|________________|
| F:Anil_Roy     | 1104           |
| S:Deep         | 202001011      |
| S:Tirth        | 202111014      |
| S:Maulik       | 202203027      |
|________________|________________|
```

<-->

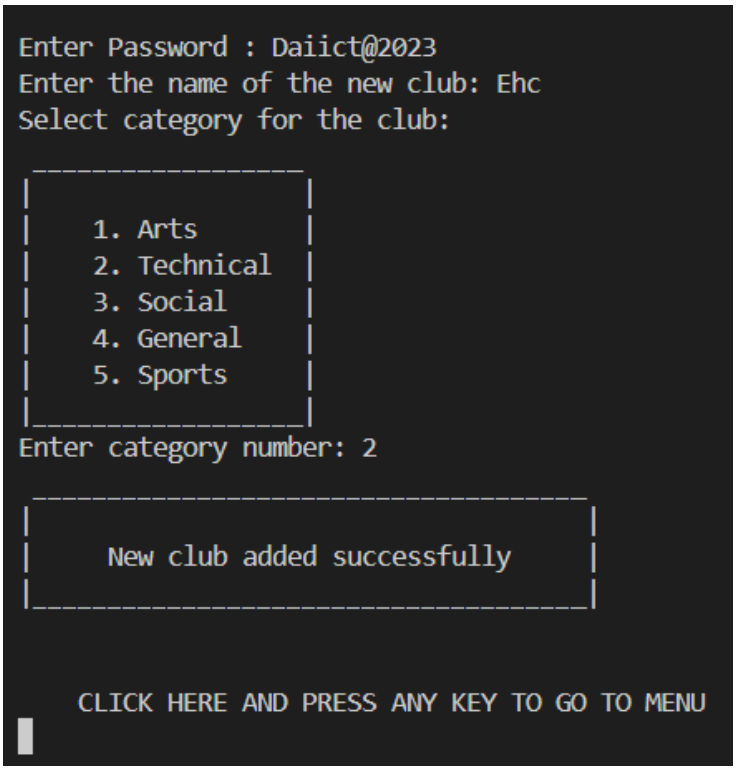
```
Enter Password : Daiict

 ____________________________________
|                                    |
|           WRONG PASSWORD           |
|  YOU DON'T HAVE ACCESS TO ADD CLUB |
|____________________________________|


    CLICK HERE AND PRESS ANY KEY TO GO TO MENU
```

➢ Now if any **official person** wants to **add a new club** then **enter 7** in the menu.

➢ After the program ask for a password and if you enter wrong password Then you don't have access to add a new club.



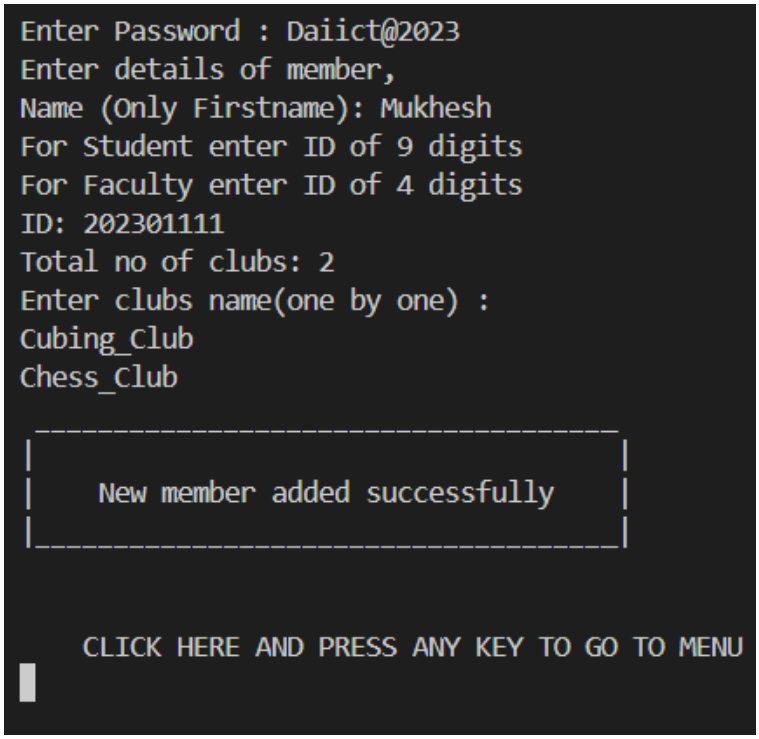
➢ After entering the **correct password** you have to enter the name of the new club and then you have to select the category of that club.

➢ Then for go to the menu press **any key**.

<--->

➢ If any **official person** wants to **add a new member** in any club then **enter 8** in the menu.

➢ After the program asks for a password and if you enter the wrong password Then you don't have access to add a new club.

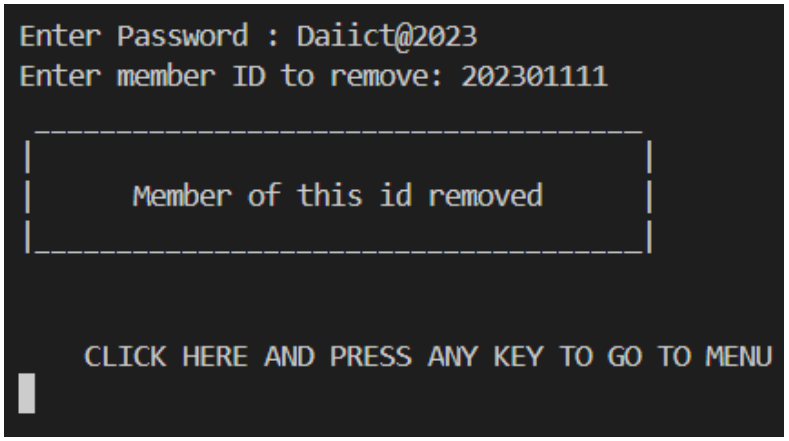


➢ After entering the **correct password** you have to enter the details of the new member and then you have to enter the club.

➢ Then for go to the menu press **any key**.

<---------->

➢ If any **official person** wants to **remove a member** from all club then **enter 9** in the menu.



➢ After entering the **correct password** you have to enter the ID of the member and then that member is removed from all related clubs.

➢ Then for go to the menu press **any key**.

<---------->

➢ For e**xit program enter choice number 10** in the menu.