Write a program which books tickets for a theme park (a booking system). The theme park has only 40 persons capacity. The visitors may be local and international.

The program can perform following tasks:

1. Add a new visitor information which must have the following:
   (Sr No., Full Name, ID (IC or passport), Ticket No)

2. Search visitor information based on Ticket No, Name, or ID
3. Delete the visitor information based on ticket number
4. Delete the visitor information based on full name
5. Delete the visitor information based on ID
6. Display Total No of tickets sold

For local visitors the booking system should record the following

Sr_No. Ticket_number full_name ID (local)

For international visitors the following information should be collected.

Sr_No. Ticket_number full_name ID (International)

The booking system should generate an output file "Tickets.txt".

Input Specifications (Tickets.txt)

- **It is assumed that your "Tickets.txt" is empty at the beginning or you are creating it first time but after creating just append with the data**. The first line shows the Total visitors , then next line first visitor, and next line 2nd visitor till 40 visitors. **Each new visitor's information should be added in the at end of the previous visitors.**
- The first line at display screen and "Tickets.txt" should display the total number of visitors:

  For example:

```
Total visitors: 2
```

- The Sr_No. is the serial number of visitors. 01, 02, 03 …
- Ticket_Number is the ticket number generated using decimal conversion of first 6 characters the visitor's name including the space(s) and the serial number appended at the right. If visitor's name is less than 6 alphabets, append spaces at the beginning of the name (left side). If greater than 6 alphabets, then only take first 6 alphabets.
- Full name: The full name (30 characters) and replace spaces with _ (underscore).
- ID: for local [20 characters] with appended "IC" at the left side.
- ID for international [20 characters] with appended "PASS" at the left side

For example;
Local visitor:

```
01 89971151051143201 Yasir_Hafeez IC122345676
```

For international:

```
02 89971151051143202 Yasir_Hafeez PASS122345676
```

- The search operation should match the name regardless of the capitalization or small alphabets. For example; if someone search “yasir hafeez” or “yasiR haFeez” or “YASIR HAFEEZ” the search function should provide the information of “Yasir_Hafeez”, meaning both should be considered same.
  NOTE: If more than one visitors have same name then it should display all the visitors with the same name.

- If you delete any visitor’s information based on ticket no, name, or ID, the Ticket.txt file should be updated so that it should preserve the modified visitor’s information. **In case of same multiple visitors with same name, delete the visitor’s information based on ID.**
  No need to change the serial no in case of deletion of data.

- If the total number of visitor’s information is asked, the Tickets.txt file should provide the complete list of visitors.

Output Specifications:

- The output should be displayed on screen, and another file “output.txt” should be generated which contains all the commands displayed on the screen. Each command should be separated by one blank line on the output.txt and screen, for example:

- For adding visitor’s information, the following line should be displayed and also written in the output.txt file:

```
The information of the visitor Yasir_Hafeez is recorded.
```

- Similarly, for searching the visitor by name:

```
The following record of the visitor is found:

01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
```

- For searching the visitor by id:

```
The following record of the visitor is found:

01 89971151051143201 Yasir_Hafeez IC122345676
```

- For deleting the visitor’s information (by name):

```
The visitor: Yasir_Hafeez information is deleted.
```

- In case of multiple visitors with same names the following output will be displayed.
  `The following visitors have same names, Please enter the ID to deleted:`

```
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
```

- For deleting the visitor's information (by Ticket No):

```
The visitor with Ticket No.: 89971151051143201, information is deleted.
```

- For deleting the visitor's information (by ID):

```
The visitor with ID: IC122345676, information is deleted.
```

- For displaying the list of visitors:

```
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
03 89971151051143203 Yasir_Hafeez PASS142445676
04 89971151051143204 Yasir_Hafeez IC122344690
```

- The list should display (on the screen) the visitors' information in ascending order based on visitors' name (only if is asked, so the option to display in ascending order should be displayed).
- If the names are same, no change in the displayed list of visitor's information.
- The Tickets.txt file should remain same and no change in case of displaying the list in ascending order.

If Example output of the overall program:

```
Total visitors: 4
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
03 89971151051143203 Yasir_Hafeez PASS142445689
04 89971151051143204 Yasir_Hafeez IC122344690
```

If I may delete visitor's information `PASS142445689` the output would be:

```
Total visitors: 3
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
04 89971151051143204 Yasir_Hafeez IC122344690
```

If I may add one more visitor's information the output would be:

```
Total visitors: 4
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
04 89971151051143204 Yasir_Hafeez IC122344690
05 89971151051143205 Yasir_Hafeez PASS1424454440
```

Implementation Restrictions:

Must use the following struct to store visitor's information:

```
struct visitData{

char *SrNo;
char *ticketNo;
char *Name;
char *ID;

};
```

And the following struct to store the data.

```
struct ticket {

struct visitData visitor;
struct ticket *nextPtr;

};
```

NOTE: you are not restricted to use only those variables mentioned in above structures.

A good programmer follows a good practice of program design and create several functions with prototypes for each specific task to solve the problem. You are not restricted to create as much functions as needed, nor you are forced to create functions. With appropriate comments to understand the purpose of each function or expression where needed.

Sample Output:

```
Total visitors:5
01 89971151051143201 Yasir_Hafeez IC122345676
02 89971151051143202 Yasir_Hafeez PASS122345676
03 89971151051143203 Yasir_Hafeez PASS142445689
04 89971151051143204 Yasir_Hafeez IC122344690
05 89971151051143205 Yasir_Hafeez PASS1424454440
```

Sr_No Ticket Full_name ID all are separated with one space between each other.

Deliverables:

Must submit a single file on moodle with the name Student_ID.c for example if id is 2053333 then the file name will be: 2053333.c

Restrictions:
You may use any compiler but your program must run on CodeBlocks or any other compiler compatible to C11 or newer.

Program starts with your name and student id before including any header. For example:

```
/* Name: Yasir Hafeez  Student ID: 2053333 */

#include <stdio.h>
```

You must include as much comments as needed to understand clearly the purpose of expressions or functions.

Grading Details

Your program will be graded upon the following criteria:

1) Your correctness

2) At least creating functions for any one of the following task:

    i) Adding visitor's information

    ii) Searching visitor's information

    iii) deleting visitor's information

3) Your programming style and use of white space. (Even if you have a plan and your program works perfectly if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.)

4) Compatibility to standard C11. (If your program does not compile in such an environment, you will get a sizable deduction from your grade, likely to be over 50%)

Here are six situations where **plagiarism or collusion** may arise for computing students:

Situation 1: Student B has trouble with a part of the code and asks student A for help. Student A shows his/her own code to student B to demonstrate how it has been done. This is collusion and both students will lose their marks for this section.

Situation 2: Student B has trouble with a part of the code and asks student A for help. Student A types in some code for student B. This is collusion and both students will lose their marks for this section.

Situation 3: Student B looks at student A's code without student A knowing, in order to see some code that may help them with their assignment. This is plagiarism, and student B will lose marks for this section. But if both students have produced similar code and both deny copying, then both students could be penalised.

Situation 4: Student B has trouble with the code and finds some suitable code on the Internet or in a book. Student B copies it, make a few adjustments, and gets it to work in this situation. This is plagiarism and student B will lose marks for the section. (Unless student B indicated in the assignment that this code has been used, in which case marks will be awarded for the parts the student has written.)

Situation 5: Student B has trouble with the code and consults a book or the Internet for similar code. On finding some, the student studies it to understand how the author has solved the problem. Using an improved understanding of programming, student B writes his/her own code to solve the assignment and acknowledges the assistance of the source by referencing it. This is perfectly fine and a useful way to study programming.

Situation 6: Student B has trouble with the code and asks student A for help. Student A explains some of the programming principles that student B is having trouble with, possibly giving bits of code that would work in general situations. This is perfectly fine and a useful process for both students.

- You should have written every line of code yourself and should be able to explain each line fully if asked to do so.
- Do not let other people see your code. In the real world it is good to share code, but for an assignment it could lead to you being accused of collusion, which will certainly waste your time and could lead to you losing marks or retaking the assignment.
- If other students ask for help, and you wish to help, do so by improving their understanding, not by giving them the answer. If anyone copies code you have shown them, you may both be accused of collusion. Also bear in mind that if other students get qualifications they do not deserve, you may one day fly in an aircraft with flight control software written by one of them.
- If you do not have time to help them, or if they are asking questions, you feel you should not answer, tell them to ask me.
- When you have problems with your own work, ask me. Do not worry about asking questions that are too closely related to the assignment - if I am not able to answer, I will simply tell you. If I am giving you evasive answers, I am probably trying to help you without giving away too much about possible assignment solutions. (Or it may be that I don't know the answer!) 😊