Reading from a formatted File (Comma Separated File)

Working Example

Suppose, we have the following informations related to each student.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is a complete record of one student.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is a complete record of another student.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is a complete record of another student.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is a field of all records.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is another field.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

This is another field.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

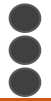
This is another field.

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1

Working Example

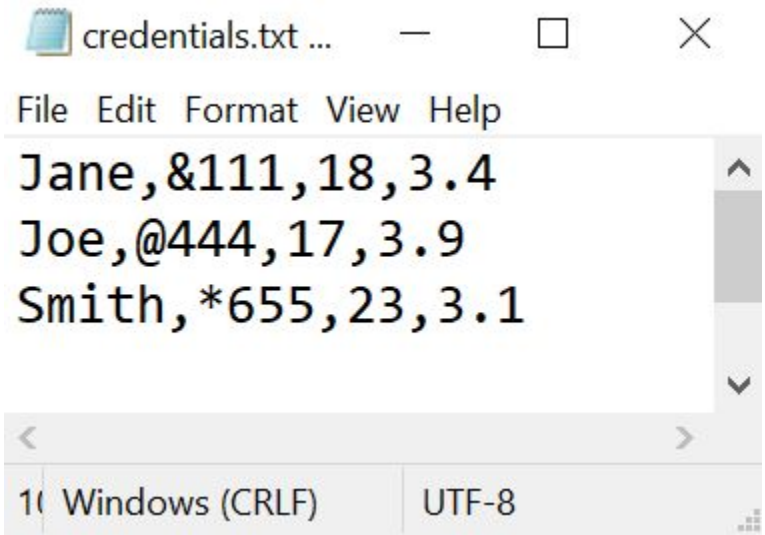
In Short, we have multiple records and each record has 4 fields (user name, password, age and Cgpa).

User Name	Password	Age	Cgpa
Jane	&111	18	3.4
Joe	@444	17	3.9
Smith	*655	23	3.1



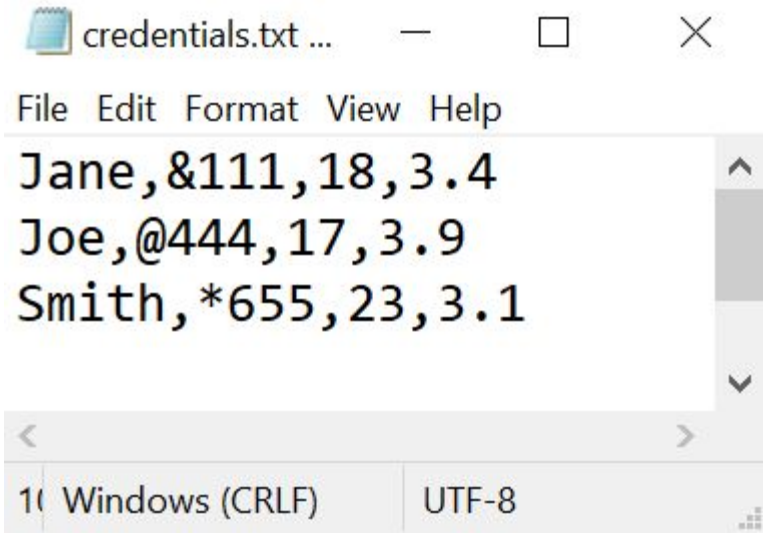
Working Example

Now, we have stored this information related to each record in a single line separated by commas in a text file.



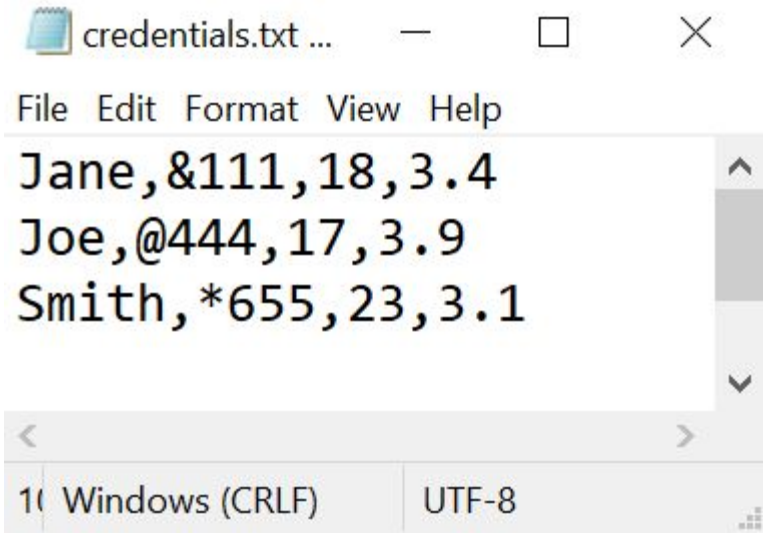
Working Example

Write a C++ Program that reads credentials.txt comma separated file till the end and then ask the user to enter a username and password and then display the information if it is a valid student.



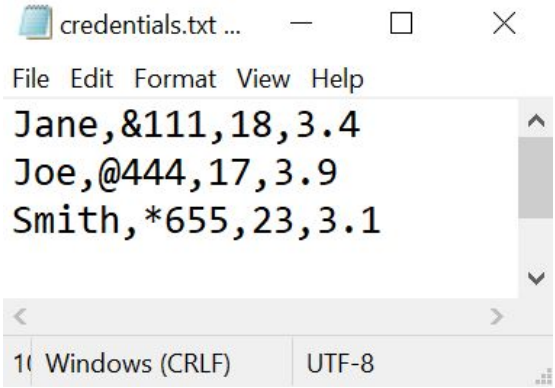
Working Example

How to approach this problem?



Working Example

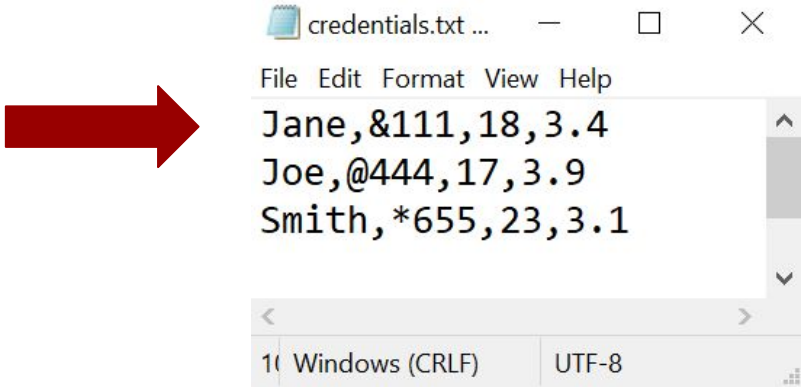
We know how to read a file till the end.
We will get information of each record in a **string**.




```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
}
}
```

Working Example

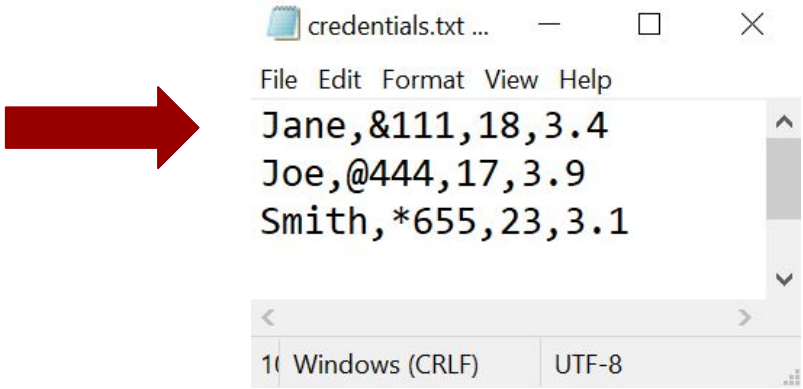
But the information is separated by commas.




```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
}
}
```

Working Example

Now, we have to separate every field of every record and store in parallel arrays.




```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
}
}
```

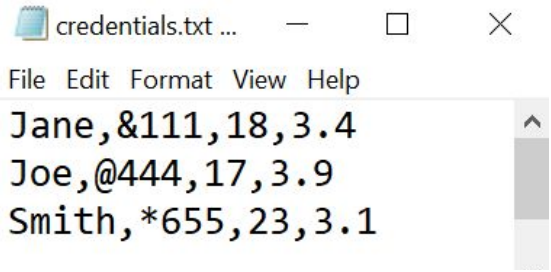
Working Example

Let's make a function that takes one record in string along with the number of the field to specify which part we want to separate. Then the function will return the separated string i.e, name if the field number is 1 and password if the field number is 2.

```
string ParseRecord(string record, int field)
{
}
```

Working Example

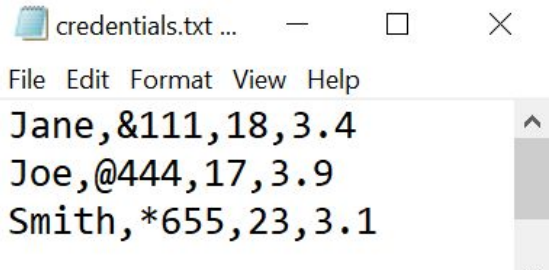
Now, it returns
Name if the field is 1
Password if the field is 2
Age if the field is 3
CGPA if the field is 4.




```
string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Working Example

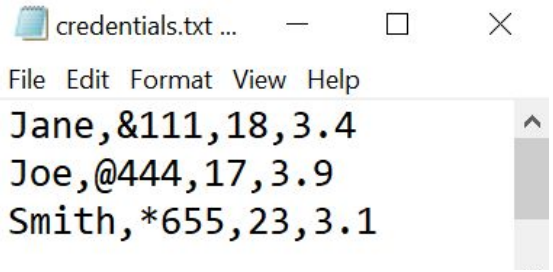
Now, we can use this function to separate every field.




```
string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Working Example

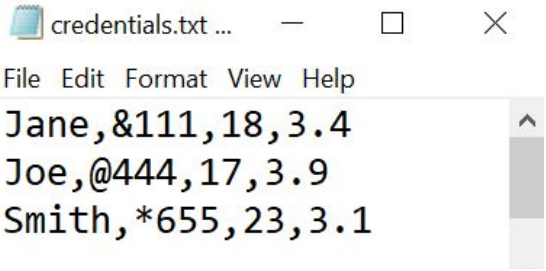
Now, we can use parseRecord() to separate every field.




```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
    names[idx] = parseRecord(word, 1);
    passwords[idx] = parseRecord(word, 2);
    ages[idx] = parseRecord(word, 3);
    cgpa[idx] = parseRecord(word, 4);
    idx = idx + 1;
}
}
```

Working Example

Do you see any problem in parseRecord() function?

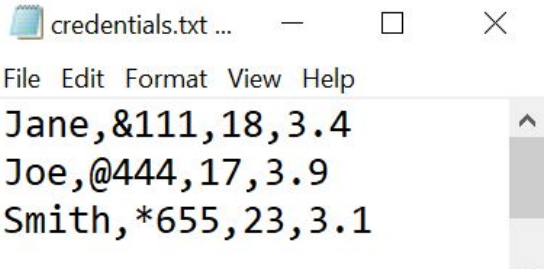



```
string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Working Example

Return type of this function is string therefore, it will always return a string.

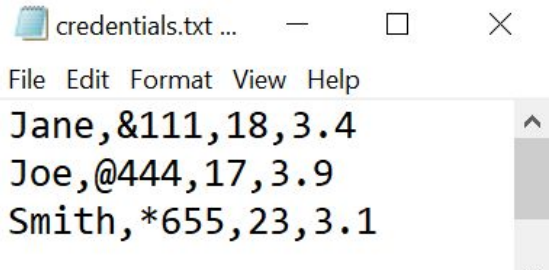
“18” = parseRecord(“Jane,&111,18,3.4”, 3)




```
string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Working Example

Although, the age and CGPA of the student is in **int** and **float** data type.




```
string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Working Example

For that we will use built-in **stoi()** function that converts a string into integer.

18 = stoi(“18”)

```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
    names[idx] = parseRecord(word, 1);
    passwords[idx] = parseRecord(word, 2);
    ages[idx] = stoi(parseRecord(word, 3));
    cgpa[idx] = parseRecord(word, 4);
    idx = idx + 1;
}
}
```

Working Example

For that we will use built-in **stof()** function that converts a string into float.

3.4 = stof(“3.4”)

```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
    names[idx] = parseRecord(word, 1);
    passwords[idx] = parseRecord(word, 2);
    ages[idx] = stoi(parseRecord(word, 3));
    cgpa[idx] = stof(parseRecord(word, 4));
    idx = idx + 1;
}
}
```

Working Example

Now, we have populated the parallel arrays correctly. The rest is now just a piece of cake.

```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
    names[idx] = parseRecord(word, 1);
    passwords[idx] = parseRecord(word, 2);
    ages[idx] = stoi(parseRecord(word, 3));
    cgpa[idx] = stof(parseRecord(word, 4));
    idx = idx + 1;
}
}
```

```
#include <iostream>
#include<fstream>
using namespace std;

string names[100], passwords[100];
int ages[100];
float cgpa[100];
int idx = 0;

string parseRecord(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

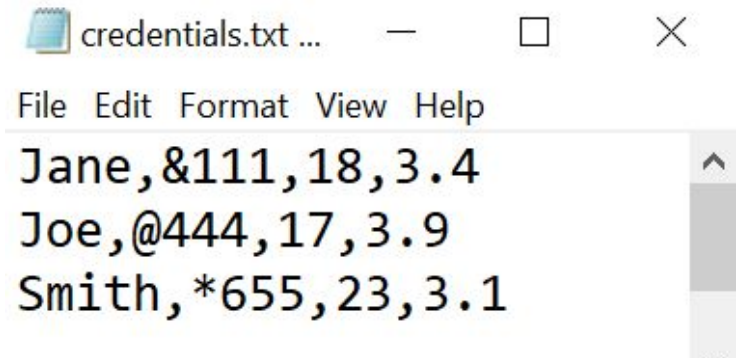
```
void inputData()
{
string word;
fstream f_variable;
f_variable.open("credentials.txt", ios::in);
while (!(f_variable.eof()))
{
    getline(f_variable, word);
    names[idx] = parseRecord(word, 1);
    passwords[idx] = parseRecord(word, 2);
    ages[idx] = stoi(parseRecord(word, 3));
    cgpa[idx] = stof(parseRecord(word, 4));
    idx = idx + 1;
}
}

void displayOutput(int index)
{
   cout << "Name \t Age \t CGPA" << endl;
   cout << names[index] << " \t " << ages[index];
   cout << " \t " << cgpa[index];
}
```

Working Example

```
main()
{
    inputData();
    string n, p;
    cout << "Enter Name: ";
    getline(cin, n);
    cout << "Enter Password: ";
    getline(cin, p);
    for(int x = 0; x < idx; x++)
    {
        if(n == names[x] && p == passwords[x])
        {
            displayOutput(x);
        }
    }
}
```

Self Assessment

1. Read a file named credentials.txt in parallel arrays



Self Assessment

Now your task is to ask for a username and password from the user and check in your parallel arrays whether the username and password is present or not. If present then display the message "Access Granted" in green colour and then display a happy message if the CGPA is greater than 3.5
otherwise "Access Denied" in red colour.
Note: For displaying the text in different colour, follow this tutorial.
https://www.geeksforgeeks.org/how-to-print-colored-text-in-c/