MCS 360 Project One : query CTA stop names given stop identification numbers due Wednesday 20 September at 3PM

The main goal of this project is to process a comma separated values file with a C++ program. The data read from file will be stored in arrays. The user of the program can query the arrays interactively.

The file we consider is the information about the stops in the Chicago Transit Authority system. The comma separated values file `stop.txt` is contained in the archive `google_transit.zip` which can be downloaded from `http://www.transitchicago.com/downloads/sch_data`. A general description of the file formats is available at `http://www.transitchicago.com/developers/gtfs.aspx`.

In this project, we use only the identification numbers of the stops and the names of the stops, respectively stored in the fields `stop_id` and the `stop_name` in the file `stops.txt`. The first two lines of `stops.txt` start as follows:

```
stop_id,stop_code,stop_name, ...
1,1,"Jackson & Austin Terminal", ...
```

For every stop, there is one line in the file `file.txt`. The identification of the stop occurs first on every line. The name of the stop can be found between the second and the third comma.

The program makes two passes through the file `stops.txt`:

1. In the first pass, the lines are counted. The number of lines bounds the number of stops.
2. In the second pass, two arrays are filled with data on file.

   Before the second pass, the number of lines is used to allocate two arrays of size equal to the number. The first array is an array of integers to store the identification number for each stop. The second array is an array of strings, used for the names of the stops. The two arrays relate to each other in the following manner:

   If the number at position `idx` *in the first array is* `stopid`*, then the string at position* `idx` *in the second array holds the name of the stop with identification number* `stopid`*.*

After the construction of the two arrays, the interactive part of the program starts and the user is prompted for integer numbers. If the entered number matches a stop identification number, then the corresponding name of the stop is printed, otherwise an error message is shown.

If the executable is in the file `stopsnames`, then a session at the command prompt `$` could run as:

```
$ ./stopsnames
Opening stops.txt ...  Counted 11415 lines ...  Counted 11413 stops.
Give a stop id (0 to stop) : 1
Stop 1 has name "Jackson & Austin Terminal"
Give a stop id (0 to stop) : 205
Stop 205 has name "UIC-Halsted Blue Line Station"
Give a stop id (0 to stop) : 291
There is no stop with id 291.
Give a stop id (0 to stop) : -3
There is no stop with id -3.
Give a stop id (0 to stop) : 90123123
There is no stop with id 90123123.
Give a stop id (0 to stop) : 0
$
```

You may assume that the user enters integer numbers. Your program should not crash if the entered number is out of bounds.

Some important points:

1. You may (not must) collaborate in pairs.
   Both authors will receive the same number of points. A pair consists of two, not three or more.
   The formation of pairs must be done as soon as possible, before September 12, at noon.
2. The dialogue in the interactive session should run using the exact same words as shown in the example session on the previous page.
3. Your program will be tested with the `g++` compiler. Points will be deducted if you program does not compile with `g++` or if your `g++` compiled program behaves in an unexpected manner.
4. The solution consists of one file. You must use functions. Correct programs without the proper use of functions will receive only half of the points.
5. Every function in your program must have a well documented prototype, placed before the `main()`.
6. Handing in an incomplete but working program is better than handing in a program that crashes or does not run at all.
7. The first line of your C++ program must be

   ```
   // MCS 360 Project One by <Authors>
   ```

   where you replace the `<Authors>` by your names.
8. If you work in a pair, then only one author should submit.
9. Email your solution to the project to `janv@uic.edu` before 3PM on Wednesday 20 September so the date of the email is proof of an on time submission. Bring a printed version of your solution to class.
10. The subject line of your email should be MCS 360 Project One by `<Authors>` where you replace the `<Authors>` by your names.
11. There is an automatic extension of the deadline till 5PM on the same day. However, late submissions are penalized with ten points off. Submissions after 5PM will not be graded.

If you have questions or difficulties with the project, feel free to come to my office for help.