Write a program that reads a CSV file, parses each line, populates an appropriately defined struct and writes the struct to a file in binary format (over writing the file if it exists).

Create your own CSV file with at least ten lines of data.

Each CSV file line contains five fields:

- AccountNumber
- FirstName
- LastName
- AccountBalance
- LastPaymentAmount

You can assume that the file format is always correct (minimal validation required).
CSV file line example:

```
1001, Cosmo, Kramer, 5827.48, 1500.00
```

Name your input file ***accounts.csv***
Name your output file ***accounts.dat***
These filenames may be hard coded in your program (use default directory)

Program 2 – display.c

Write a program that reads the data file generated in Program 1 then displays the data to the screen.

Display column headers and format your output in an appropriate manner.

Objective

The purpose of this assignment is to create an index file for the data file created in Assignment 7. You will modify display.c to use this index file to display the data in sorted order. (Account Balance Descending)

Your Task

Program 1 – index.c

Write a program that reads the ***accounts.dat*** created in Assignment 7 and creates an index file to enable display.c to display the data in sorted order. (Account Balance Descending)

Name your index file ***accounts.idx***

The data filename and index filename will be passed to the program via command line arguments. For simplicity, have the files reside in the program's current directory.
Example:

```
exename datafilename indexfilename
```

When the program starts, validate the arguments as follows:

a) Check for the proper number of arguments. If *incorrect*, display a useful error message to the user and exit the program.

b) Ensure that the datafile exists. If the file *does not* exist, display a useful error message to the user and exit the program.

Program 2 – display.c (Revise Assignment 7 version)

Modify display.c to use the index file created in index.c.

The data filename and index filename will be passed to the program via command line arguments. For simplicity, have the files reside in the program's current directory.
Example:

```
exename datafilename indexfilename
```

When the program starts, validate the arguments as follows:

a) Check for the proper number of arguments. If *incorrect*, display a useful error message to the user and exit the program.

b) Ensure that the datafile exists. If the file *does not* exist, display a useful error message to the user and exit the program.

c) Ensure that the indexfile exists. If the file *does not* exist, display a useful error message to the user and exit the program.

Create a menu which allows the use to display the data in:

1. Natural order
2. Account Balance Descending
3. Exit

Place the menu in a loop so the user can display in different orders without having to restart the program.