ECE 3331, Fall 2023 due Sunday 09/10 at 11:59 pm

Programming assignment 03 Turn in this program to the Canvas course website by the due date.

Your computer contains a keyboard buffer, a small amount of random access memory (RAM) that it uses to store the users' keystrokes while interacting with your program. When your program executes a scanf( ) function, your program will wait for the user to type in some characters (letters, digits, punctuation, etc) from the keyboard and then hit the Enter-key. When the Enter-key is hit, your program will then access the characters within the keyboard buffer according to the format string in the scanf( ) function. Any characters in the keyboard buffer that were not read by the scanf( ) function, will remain in the keyboard buffer waiting to be read-in by the next scanf( ) function call.

Where c is a character variable, the scanf( ) function scanf("%c",&c1) will read in one character from the keyboard.
If the user hits the q-key (for lowercase 'q') followed by the <Enter> key, that action constitutes two characters from the keyboard: 'q' and '\n'. The <Enter> key generates the '\n' character.

Another important fact, is that characters entered from the keyboard that are not yet read by a scanf( ) will sit in a keyboard queue waiting to be read by the next scanf( ) in your program.

For example:
If you have a program that contains:

```
printf("Please enter a single character from the keyboard\n");
scanf("%c",&c1);
scanf("%c",&c2);
```

and the user presses any letter key followed by the <Enter> key, then c1 will contain the character from the keyboard, and c2 will contain '\n' generated by the <Enter> key.

If you have a program that contains:

```
printf("Please enter two characters from the keyboard\n");
scanf("%c",&c1);
scanf("%c",&c2);
```

and the user presses any two letter keys followed by the <Enter> key, then c1 will contain the 1st character from the keyboard, and c2 will contain the 2nd character. The '\n' generated by the <Enter> key will remain waiting in the keyboard queue to be read by the next scanf("%c",) , if there is one, in your program.

If you have a program that contains:

```
printf("Please enter two characters from the keyboard.\n");
scanf("%c",&c1);
scanf("%c",&c2);
printf("Please enter two more characters from the keyboard\n");
scanf("%c",&c3);
scanf("%c",&c4);
```

then c3 will contain the '\n' from the first <Enter> key.

The same holds when reading a number from the keyboard. Hitting the <Enter> key to enter the number generates a '\n' character that is not part of the number that is read from the keyboard.
If you have a program that contains:

```
printf("Please enter an integer.\n");
scanf("%d",&i1);
printf("Please enter a character.\n");
scanf("%c",&c1);
```

Variable i1 will contain the integer, but character variable c1 will always contain the '\n' from the <Enter> key following the integer.

So, if you have a program that contains:

```
printf("Please enter an integer.\n");
scanf("%d",&i1);
scanf("%c",&c1);
printf("Please Enter a character.\n");
scanf("%c",&c2);
```

Variable i1 will contain the integer, character variable c1 will contain the '\n' from the <Enter> key, character variable c2 will contain the character entered by the user, and the 2nd '\n' from the 2nd <Enteer> key will be waiting in the keyboard queue to be read by the next scanf( ) in your program.

When you scanf(“%d”,&i1) an integer i1 from the keyboard, the scanf( ) will skip past any white space (blank space, tabs, newlines from <Enter>) that are sitting in the keyboard queue in front of an integer.
So, if you have a program that contains:

```
printf("Please Enter an integer.\n");
scanf("%d",&i1);
printf("Please Enter another integer.\n");

scanf("%d",&i2);
```

Variable i1 will contain the 1st integer and variable i2 will contain the 2nd integer, because scanf("%d",&i2); will skip past the ‘\n’ in the keyboard queue from the <Enter> of the 1st integer.

Using what you have learned above, write a program to input a list of names and account balances from the keyboard and write them into a file called "out.txt".

(1) Open an output file called "out.txt" for writing. The file "out.txt" will be located in the same directory from which the executable from your program will be run.
(2) Write a loop wherein you ask the user if they want to enter another complete name (lastname , firstname) and account balance. Have the user respond with a single character 'y' or 'n' (note: the user will have to hit the <Enter> key after the single character ‘y’ or ‘n’).
(3) Each time the user responds with 'y', instruct the user to enter the complete name and hit the <Enter> key.
Read the complete name (lastname, firstname) and write it to the output file followed by a semi-colon and a blank space.
To read-in the complete name (lastname, firstname) from the keyboard, use another loop that reads a single character at a time from the keyboard and writes a single character to the file. When the character that you read rom the keyboard input is exactly the newline character ‘\n’ (from the <Enter> key), then you know the user has finished entering the complete name.
Next instruct the user to enter the account balance as a floating point number followed by the <Enter> key.
Read this value and write it to the same line, but precede the value by the dollar sign '$' and print the value with exactly 2 digits past the decimal point.
Since the user might want to enter another name and account balance, don’t leave the ‘\n’ from the <Enter> key sitting in the keyboard queue, or else that is the 1st character that your next scanf(“%c”,..) will read in.
(4) If at any iteration through the loop, the user responds with 'n' to the question of inputting another name and account balance, close the output file and end the program.

For example, your file out.txt will look like

```
Doe, Jane: $211.33
Smith, John: $17.44
de Mills, Cecil: $557.44
```