

Homework #9

(C Programming for Beginners - OnLine)

There are three distinct areas you need to research and focus on to get your final done properly:

- Process command line arguments to get the names of input and output files from the user (check below in the notes section)
- Read and write information from and into the disk files
- Parse the data read from input file, store in data structures of your choice, process it to get the required result and then print it on the screen and into the output file.

Here are couples of exercises, which will help you in writing code for the above problems.

When you write a program named `extrahw_3.c`. Your main functions header in this file should look like this:

```
int main(int argc, char *argv[])
```

where, `argv` is the array of character strings. This parameter will hold the command line arguments passed by operating system when your program starts first time. This array is filled by the operating system depending upon how did you start the program.

And `argc` holds an integer, which has a value indicating how many elements are there in the `argv` array.

There are two ways to run any program. You can run the program from a terminal/command window or through your IDE (VC++/XCode). Either way, you can provide the command line arguments when you run the program.

- When you start the program (a compiled and linked executable) from terminal/command window, you typically start this program like this in Windows (for example if the exe name is **Extrahw_3.exe**):

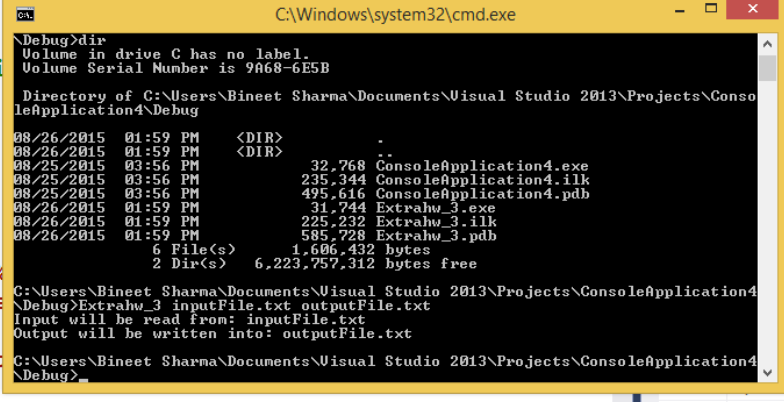
Extrahw_3 inputFile.txt outputFinal.txt

```

int main(int argc, char *argv[])//main function with command line arguments
{
    FILE *iFilePointer, *oFilePointer; //file pointers for input & output
    char lineInput[200];

    if (argc != 3) // Validity check
    {
        printf("Usage: %s inputFile.txt outputFile.txt\n", argv[0]);
        exit(1);
    }
    else
    {
        printf("%s: %s\n", argv[0], argv[1]);
        printf("Input will be read from: %s\n", argv[1]);
        printf("Output will be written into: %s\n", argv[2]);
    }
}

```



Or, in mac, it will look like this:

Bineets-MBP:~ bineetsharma\$Extrahw_3 inputFile.txt
outputFile.txt

When you start the program from VC++ (and other IDEs), you can directly set the command line arguments in the properties dialog box as described in the notes section below. In this case, you don't give the name of your executable; you only need to provide two arguments (inputFile.txt outputFile.txt).

In both cases, whether you started the program through IDE or through command-line the operating system will pick the name of executable and everything after that as array of strings and assign them to the argv array elements.

The argc parameter is assigned the number of total arguments (words in the command line in this case) that will include the name of the executable (e.g. extrahw_3 in this case, including it's absolute path) and argv array will have that many numbers of strings as there are words (three strings in this case).

So, this is what the OS will do for you:

```

argc = 3
argv[0] = "Extrahw_3.exe" (note: in fact full path of the .exe file)
argv[1] = "inputFile.txt"
argv[2] = "outputFile.txt"

```

9.1 Write the code in your main function so that your code will display a usage message like this when there are less than three arguments.

Usage: Extrahw_3 inputFile.txt outputFile.txt

However, if the input contains three arguments then display the message like this (the name of the input and output file will be exactly as what was given by the user):

Input will be read from: inputFile.txt

Output will be written into: outputFile.txt

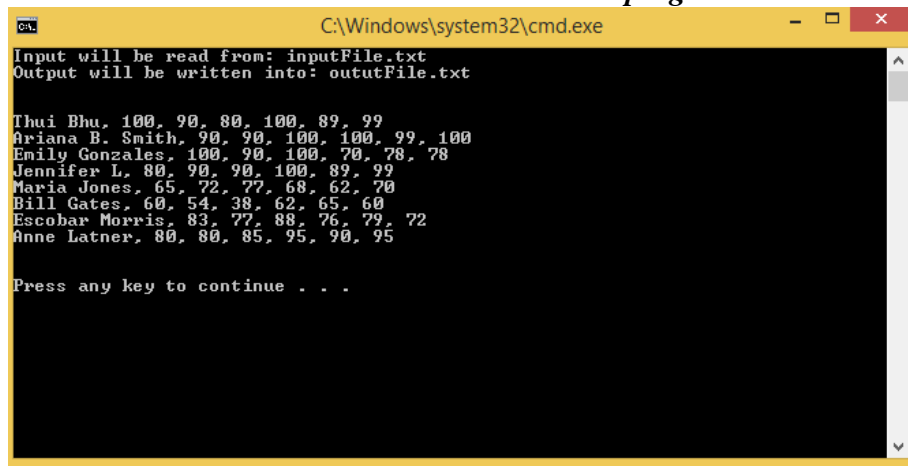
Note: You don't need to run the program from command line/ terminal. I gave this as an example only. You can keep working on VC++ or XCode as you have been doing for this homework and the final as well

9.2 Now, expand your earlier program and read the data from an input file (attached) which will have a record of one student as outlined in the final. Parse the data into individual units, e.g. name, quiz1, quiz2 etc. Then print whole record in the screen and print only the name of students in the output file.

For example: inputFile.txt contains:

```
Thui Bhu, 100, 90, 80, 100, 89, 99, 88
Ariana B. Smith, 90, 90, 100, 100, 99, 100, 95
Emily Gonzales, 100, 90, 100, 70, 78, 78, 80
Jennifer L, 80, 90, 90, 100, 89, 99, 85
Maria Jones, 65, 72, 77, 68, 62, 70, 65
Bill Gates, 60, 54, 38, 62, 65, 60, 50
Escobar Morris, 83, 77, 88, 76, 79, 72, 76
Anne Latner, 80, 80, 85, 95, 90, 95, 90
```

Your screen should look like this when the program runs:

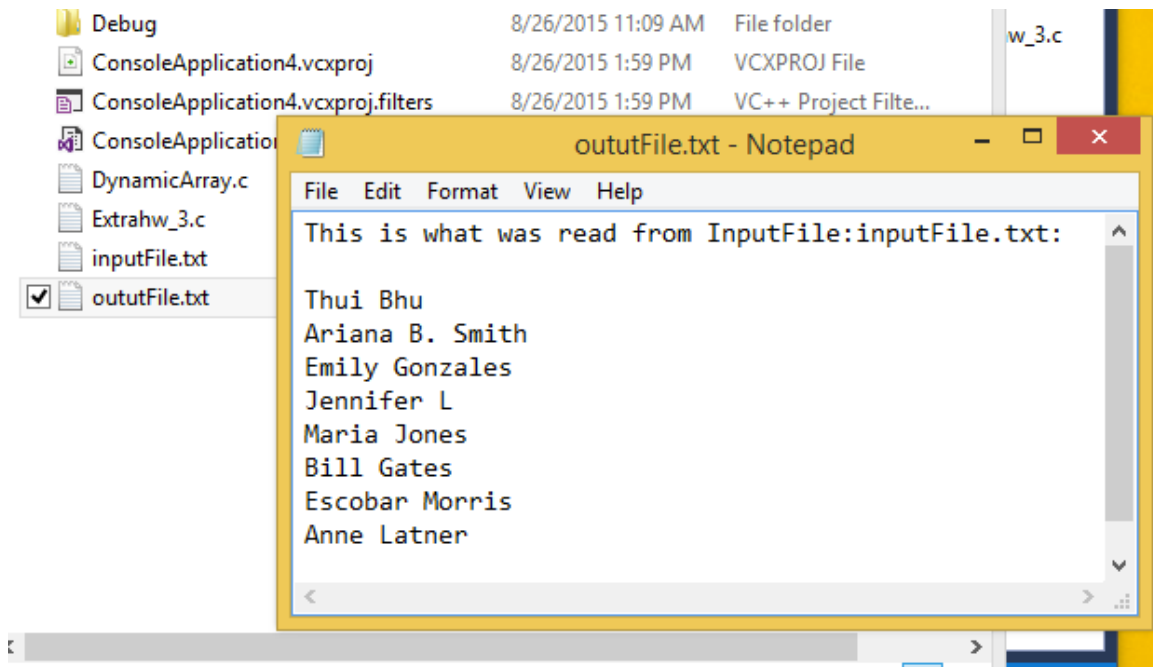
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text displayed is as follows:

```
Input will be read from: inputFile.txt
Output will be written into: oututFile.txt

Thui Bhu, 100, 90, 80, 100, 89, 99
Ariana B. Smith, 90, 90, 100, 100, 99, 100
Emily Gonzales, 100, 90, 100, 70, 78, 78
Jennifer L, 80, 90, 90, 100, 89, 99
Maria Jones, 65, 72, 77, 68, 62, 70
Bill Gates, 60, 54, 38, 62, 65, 60
Escobar Morris, 83, 77, 88, 76, 79, 72
Anne Latner, 80, 80, 85, 95, 90, 95

Press any key to continue . . .
```

Your outputFile.txt should have this:



9.3 Set the command line arguments in VC++