Homework #9 Solution

(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[])
```

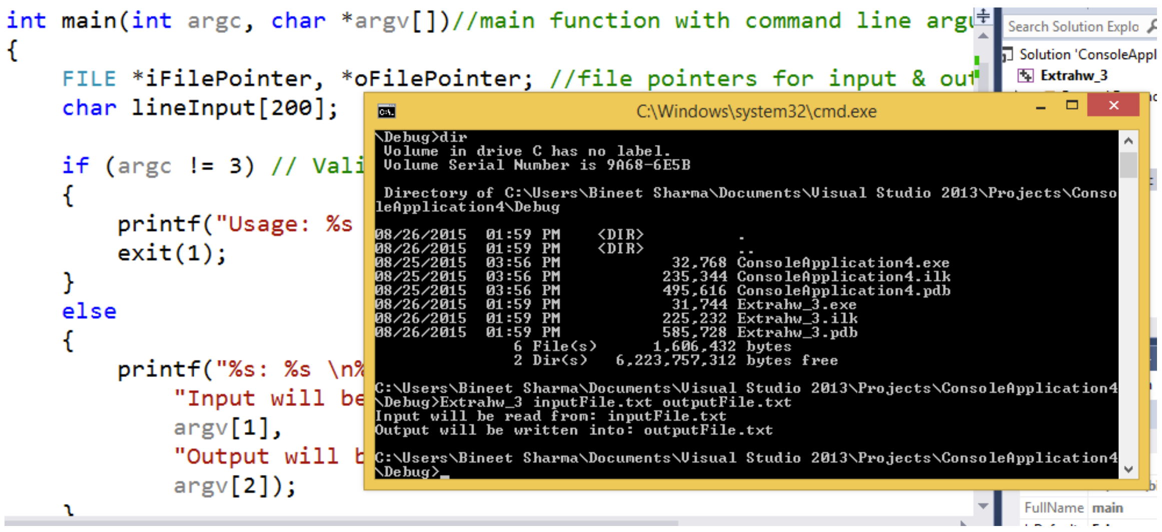
where, argv is the array of character strings. This parameter will hold the command line arguments passed by operating system when your program starts first time. This array is filled by the operating system depending upon how did you start the program.

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 outpufFinal.txt



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

    if (argc != 3) // Vali
    {
        printf("Usage: %s
        exit(1);
    }
    else
    {
        printf("%s: %s \n%
            "Input will be
            argv[1],
            "Output will b
            argv[2]);
```

```
\Debug>dir
 Volume in drive C has no label.
 Volume Serial Number is 9A68-6E5B

 Directory of C:\Users\Bineet Sharma\Documents\Visual Studio 2013\Projects\Conso
leApplication4\Debug

08/26/2015  01:59 PM    <DIR>          .
08/26/2015  01:59 PM    <DIR>          ..
08/25/2015  03:56 PM            32,768 ConsoleApplication4.exe
08/25/2015  03:56 PM           235,344 ConsoleApplication4.ilk
08/25/2015  03:56 PM           495,616 ConsoleApplication4.pdb
08/26/2015  01:59 PM            31,744 Extrahw_3.exe
08/26/2015  01:59 PM           225,232 Extrahw_3.ilk
08/26/2015  01:59 PM           585,728 Extrahw_3.pdb
               6 File(s)      1,606,432 bytes
               2 Dir(s)   6,223,757,312 bytes free

C:\Users\Bineet Sharma\Documents\Visual Studio 2013\Projects\ConsoleApplication4
\Debug>Extrahw_3 inputFile.txt outputFile.txt
Input will be read from: inputFile.txt
Output will be written into: outputFile.txt

C:\Users\Bineet Sharma\Documents\Visual Studio 2013\Projects\ConsoleApplication4
\Debug>_
```

Or, in mac, it will look like this:

```
Bineets-MBP:~ bineetsharma$Extrahw_3 inpuFile.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 (`inpuFile.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 inpuFile.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

Solution:

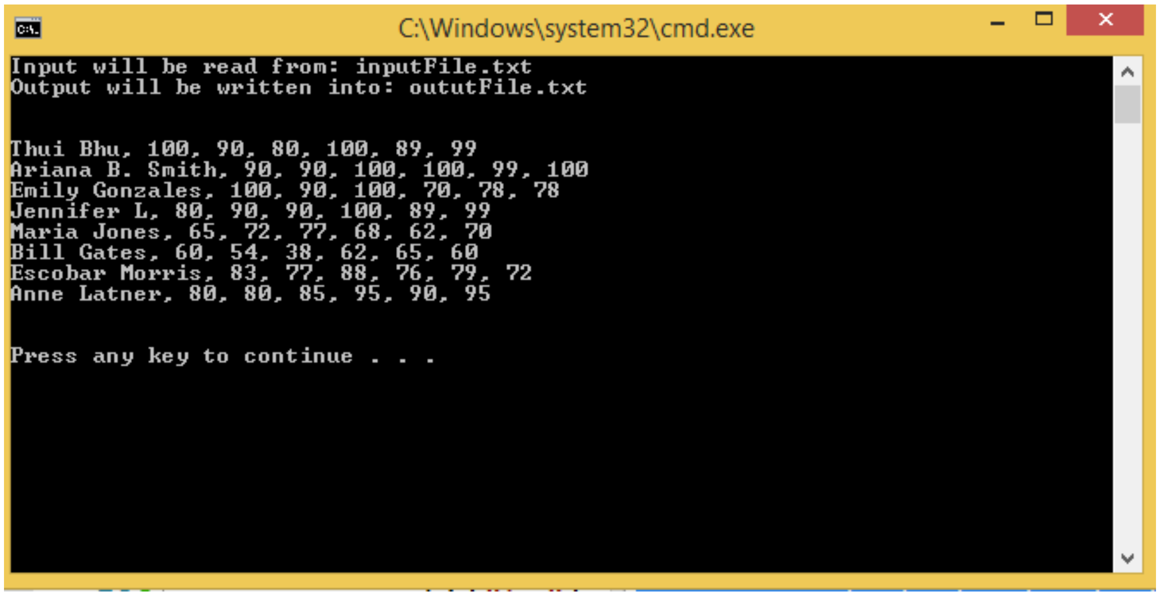
```
#include <stdio.h>

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

      if (argc != 3) //Validate that argument count not equal to 3
      {
            printf("Usage: %s inputFileName outputFileName\n",
                   argv[0]);
            exit(1);
      }
      else
      {
            printf("%s: %s \n%s: %s\n",
                  "Input will be read from",
                  argv[1],
                  "Output will be written into",
                  argv[2]);
      }
}
```

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:

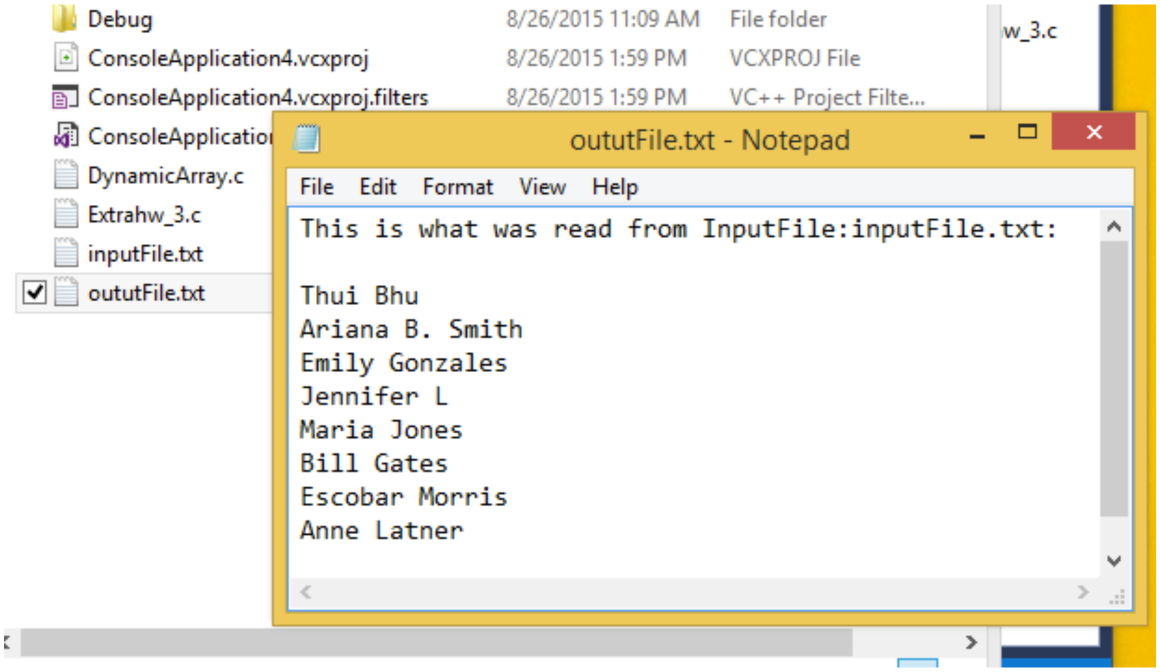
```
Thui Bhu, 100, 90, 80, 100, 89, 99, 88
Ariana B. Smith, 90, 90, 100, 100, 99, 100, 95
Emily Gonzales, 100, 90, 100, 70, 78, 78, 80
Jennifer L, 80, 90, 90, 100, 89, 99, 85
Maria Jones, 65, 72, 77, 68, 62, 70, 65
Bill Gates, 60, 54, 38, 62, 65, 60, 50
Escobar Morris, 83, 77, 88, 76, 79, 72, 76
Anne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

Your screen should look like this when the program runs:



```
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:


Debug 8/26/2015 11:09 AM File folder
ConsoleApplication4.vcxproj 8/26/2015 1:59 PM VCXPROJ File
ConsoleApplication4.vcxproj.filters 8/26/2015 1:59 PM VC++ Project Filte...
ConsoleApplicatio
DynamicArray.c
Extrahw_3.c
inputFile.txt
oututFile.txt
w_3.c
oututFile.txt - Notepad
File Edit Format View Help

```
This is what was read from InputFile:inputFile.txt:

Thui Bhu
Ariana B. Smith
Emily Gonzales
Jennifer L
Maria Jones
Bill Gates
Escobar Morris
Anne Latner
```

Solution:

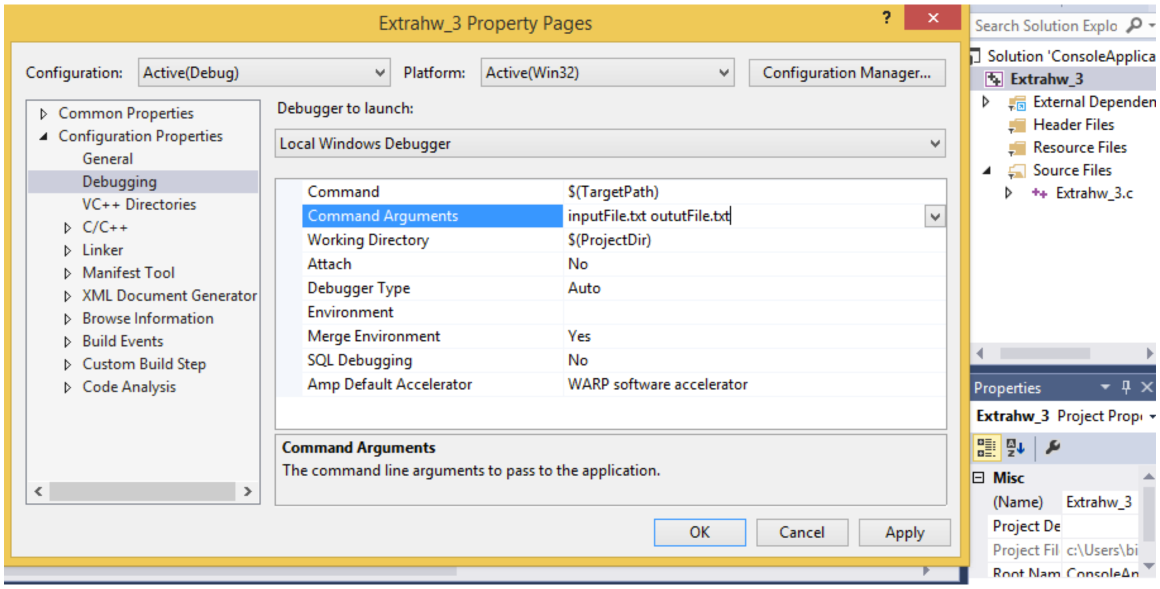
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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

	char name[30];
	char buf[100];
	char *token;
	int q1, q2, q3, q4, midi, midii, final;

	if (argc != 3) // Validate that argument count not equal to 3
	{
		printf("Usage: %s inputFileName outputFileName\n", argv[0]);
		exit(1);
	}
	else
	{
		printf("%s: %s \n%s: %s\n",
			"Input will be read from",
			argv[1],
			"Output will be written into",
			argv[2]);
```

```
                printf("\n\n");

                //properly open the file and give the error messages
                if ((iFilePointer = fopen(argv[1], "r")) == NULL)             {
                      printf("An error has been generated while \
                                 attempting to open the input \
                                 file %s\n", argv[1]);
                             exit(1);
                }
                // Validate that output file can be openned in 'write' mode
                if ((oFilePointer = fopen(argv[2], "w")) == NULL)             {
                      printf("An error has been generated while \
                                 attempting to open the output \
                                 file  %s\n", argv[2]);
                      fclose(iFilePointer);
                      //Make sure to close the previously opened file
                      exit(1);
                }

                //write the header in the output file
                fprintf(oFilePointer, "This is what was read from InputFile:%s: \
                                       \n\n",
                      argv[1]);

                while (fgets(buf, sizeof(buf), iFilePointer) != NULL)
                {
                      token = strtok(buf, ",");
                      strcpy(name, token);
                      token = strtok(NULL, ",");
                      q1 = atoi(token);
                      token = strtok(NULL, ",");
                      q2 = atoi(token);
                      token = strtok(NULL, ",");
                      q3 = atoi(token);
                      token = strtok(NULL, ",");
                      q4 = atoi(token);
                      token = strtok(NULL, ",");
                      midi = atoi(token);
                      token = strtok(NULL, ",");
                      midii = atoi(token);
                      token = strtok(NULL, ",");
                      final = atoi(token);
                      //display in screen
                      printf("%s, %d, %d, %d, %d, %d, %d, %d\n",
                             name, q1, q2, q3, q4, midi,midii, final);
                      //print the name in the output file
                      fprintf(oFilePointer, "%s\n", name);
                }
                printf("\n");
                printf("\n");
                fclose(iFilePointer);
                fclose(oFilePointer);
         }
  }
```

9.3 *Set the command line arguments in VC++*

Solution:

Press Alt+F7 to open the Project Property dialog. Or, right click on your project name from *solution explorer* window on your top right, and then chose properties. Click on **Debugging** under **Configuration Properties** section on the left hand side. Add your command line arguments next to **Command Arguments** in the right hand side. Separate the arguments with a space. E.g. inputFile.txt outputFile.txt . Press Ok to close



Extrahw_3 Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

- Common Properties
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Code Analysis

Debugger to launch:

Local Windows Debugger

Command	$(TargetPath)
Command Arguments	inputFile.txt oututFile.txt
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command Arguments
The command line arguments to pass to the application.

OK Cancel Apply

Search Solution Explo

Solution 'ConsoleApplica
- **Extrahw_3**
  - External Dependen
  - Header Files
  - Resource Files
  - Source Files
    - Extrahw_3.c

Properties

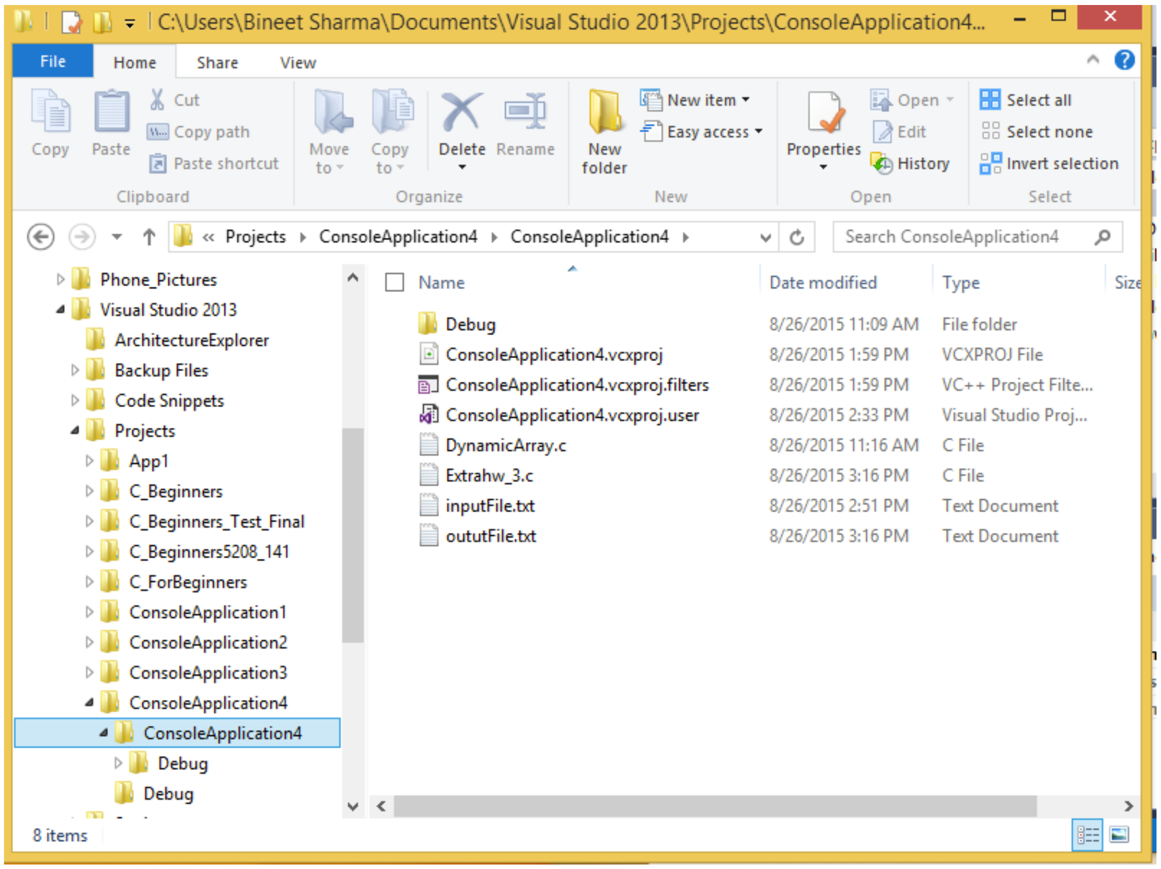
Extrahw_3 Project Prop

Misc

(Name)	Extrahw_3
Project De	
Project Fil	c:\Users\bi
Root Nam	ConsoleAp

Input File and Output file location: Now you need to put your input file in the right location. First, find out where is your VS project files are located. There will be a folder for Solution and under that will be another subfolder for Project. For example my solution and project are named same: ConsoleApplication4 and they are located in this location:

C:\Users\Bineet Sharma\Documents\Visual Studio 2013\Projects\ConsoleApplication4\ConsoleApplication4\



C:\Users\Bineet Sharma\Documents\Visual Studio 2013\Projects\ConsoleApplication4...

« Projects › ConsoleApplication4 › ConsoleApplication4 ›

Name	Date modified	Type
Debug	8/26/2015 11:09 AM	File folder
ConsoleApplication4.vcxproj	8/26/2015 1:59 PM	VCXPROJ File
ConsoleApplication4.vcxproj.filters	8/26/2015 1:59 PM	VC++ Project Filte...
ConsoleApplication4.vcxproj.user	8/26/2015 2:33 PM	Visual Studio Proj...
DynamicArray.c	8/26/2015 11:16 AM	C File
Extrahw_3.c	8/26/2015 3:16 PM	C File
inputFile.txt	8/26/2015 2:51 PM	Text Document
oututFile.txt	8/26/2015 3:16 PM	Text Document

Put your inputFile.txt in this folder.

If you have difficulty finding this folder, in Visual Studio, you can select one of the source files, and try Save As from the File menu, it will open the default folder location and you will know where this folder is.

Or, you can simply create inputFile.txt from same VS project it will automatically store the file in the right location.

Caution, the windows explorer does not show file extension automatically (that is why you may only see inputFile the .txt might be hidden in your case). If you are not careful about file extension, the VS may not find the right file.

BTW: If you can keep your files in different location, however, in that case, you can simply provide the full path of the file where you want it to be.

Run the program: Run the application as usual in Visual studio. It will find the right input file and read that file, and also will create the output file in the same folder.