P6 Employee Data II

- Due Nov 29, 2017 by 10am
- Points 100
- Available Nov 10, 2017 at 12am - Dec 2, 2017 at 11:59pm 23 days

This assignment was locked Dec 2, 2017 at 11:59pm.

Using file I/O, structures, pointers, vectors and functions

This program is basically Program 5, but now you need to use an Employee structure and vector of Employee pointers. Why would we use a vector of Employee pointers, you may ask. Well, this way, when we are sorting the employees we don't need to make copies of the employees in the swapping process, we can instead just change our pointers so that the pointers will be pointing to the data in the right order (ie the first pointer will end up pointing to the employee with the highest number of hours). In the whole process of sorting the employees, none of the employees will need to be moved or copied.

Here is the structure I want you to use for the program:

```
struct Employee {
  string name;
  vector<int> hours;
  int totalHrs;
};
```

And here is some sample code of using a vector<Employee * >

```
vector<Employee * > workers;  // needs to be resized later
// in read loop
 workers[row] = new Employee;
// NOTE:this employee has not been initialized yet
//    (ie has garbage and empty vector)

// to read in the name string
  fin >> workers[row]->name;
// after reading in all the employees data, this code could output
//  it for verification
for (Employee* emp : workers) {
   cout << emp->name << ": ";
   for (int hr : emp->hours) {
      cout << hr << " ";
   }
   cout << endl;
}
```

Program Description: In this program you will need to read from a file employee data for how many hours each employee worked for a week. Then for each of the employees, a total of the hours worked is made and then the employees are sorted into descending order based on their total hours.

Task Details:

- Since we are using vectors, you can wait until you read the number of employee and then you can create your vector to be of the right size. Create a vector<Employee *> that can hold pointers to the Employee structure
- Create a new Employee from dynamic memory
- Read all the employee data into that new Employee
- Add that new Employee to your vector
- Total number of hours worked in the week and store it in the Employee struct in the totalHrs member variable
- Sort the employees by the total number of hours they worked for the week. The sort should be in descending order (most hours to least)
- **Write out the information to the console in the tabular format that I show below.**
- **IMPORTANT NOTE: Remember to delete all the Employees when you are done with the program.**

Miscellaneous:

You should create and use a function to do each of the following task:

- open and read the file into the vectors and add the total hours worked for each employee
- sort the vector based on the total hours
- write out the output

You want main to be like an outline of the program. Each function should do one thing well, and be able to fit on one screen. If a function is taking two screens, think of how you can break it down. You can have additional functions if you think it will help make your program clearer.

Make sure you display the data in a nice looking format.

The data is in a text file called empdata.txt

Your program does not require any interaction with the user, other than potentially asking which file they would like to open (this is not required). If the input file is found no messages will be sent to the user; instead the program will read and fill the array, make the calculations and write the output to the screen**.** Otherwise, if you can't open the file, then you should send a message to the screen.

The input file will be in the following format:

```
5
Jones,Frank 2 3 8 3 6 3 5
Smith,Tiny  8 8 3 0 8 2 0
Turk,Tom    9 10 4 7 0 0 0
Jackson,Jim 5 6 5 6 5 6 5
Catan,Doug  7 3 8 7 2 5 7
```

It is a text file. The first line of the input file is the number of employees. You can use this number to control the loops that will do your reading. Then each line after that will include the employees name and seven integers for the number of hours that the employee worked for the week. The employee's name will not include any spaces and so it can be read as a string. Each of the numbers will have a space between them. Since an vectors can only hold only one type of data, you will need to use two different vectors; one for the names and one for the work hours.

The input files are found in the Files Section of "Canvas\Course Resource\Program 5" folder. Right now there are two files that can be used for testing. They are called empdata.txt and empdata2.txt. Next week I will add two more files empdata3.txt and empdata4.txt. **For turn in, you need to use empdata3.txt and empdata4.txt.**

To sorting the array, you must use **bubble sort** (passing through the vector comparing two elements at a time, and swapping if needed; and then making as many passes as needed to sort all the numbers). Here is an example of the Bubble Sort algorithm (Links to an external site.). Remember as you are sorting, you will need to also be moving the names in their array.

Your output should look like this:

```
Employee Weekly Hours:
Name:          S  M  T  W  T  F  S  TTL
Kirk,James    10 10  0 10 12  0  0   42
McCoy,Lenard   0  4  0  8  4  0  2   18
Scott,Annis    1  6  2  0  0  1  0   10
.....
```

Programming Note: Write the program a little piece at a time. Test each piece as your are going.

- can you open the file and read one number
- can you read all the numbers
- ….and so on.

Note: Your program may open the input file **only once** to do all the work.

Due: The last class in the last week depending on which days your class meets.

Ways to lose points:

- If your file does not contain the program header with a program description and short function description to accompany the function prototypes.
- Your code should also be consistently indented as talked about in class, and shown in the book
- You can't use global variables unless it is a const
- You should use good variable names (descriptive, and start with lower case letter)
- Proper placement of { }'s. A } should not be placed at the end of a line that contains other code.

- Open the input file more than once while reading the information
- If you use a sort other than Bubble Sort to do the sorting
- Not closing the files that you open
- No staple to keep your papers together (folding a corner or using a paper clip are not good enough)
- Did not hand in a run for both the input files **(originally named empdata3.txt and empdata4.txt)**

Turn in: A paper copy of your cpp file, and the output of a sample run of the program using the correct two input files **(originally named empdata3.txt and empdata4.txt)**. **Note:** You will have to rename your input file so that it is consistent with the filename you use within your program.

A Note about Strings:

We have not talked much about the string class yet, but using strings are pretty easy, and I think you can understand how to use them from the examples given below. What is a string? It is 0 or more char's. For example these could all be stored as strings: "hello", "123", "F-16" "hello there". Strings are defined in the string library.

```
string str;
cin >> str;      // user types: hello there
cout << str;     // hello is displayed since it stops reading at the space
```