Spring 2017 CMSC 140 Programming Project 7: Payroll

Concepts tested by the program:

1. Working with arrays
2. Using file operations
3. Using a selection sort to sort parallel arrays
4. Using a binary search with sorted arrays
5. Using a sequential (linear) search with unsorted arrays
6. Implementing functions besides function main()

Project Description

Write a C++ program that processes a payroll. Your program will calculate the wages for each employee, given the number of hours worked and the payrate per hour. It will search for two employees by number, one with a binary search (see page 462) and one with a linear search (see page 459). A selection sort (see page 476) will sort the arrays so that the employee numbers are in ascending order.

Project Specifications

Input for this project has two sources. The user must enter the names of the input and output files from the keyboard. The user will enter two employee numbers from the keyboard in response to prompts. The input file should have one line for each employee with the 3 required items: employeeID, hoursWorked, and payRate (separated by whitespace on the file) The last line of the file **must always have only -1**.

Output also has two sources. The program title and the programmer name should appear on both the console and the file. If the input file does not open, an error message should appear on the console and no attempt should be made to open the output file for any output or processing. The message "Processing complete" should appear on the console after the programmer name and just before the main program ends (make it the last statement before "return;"). The console output will show the prompts with the search numbers. The file output should show the unsorted table of the input items with the wages, followed by the sorted table with the same 4 columns. The column numbers with decimal amounts should line up on the decimal point under the column headings.

Processing Requirements

Your program should use the following one-dimensional arrays:

empId: an array of long integers to hold employee identification numbers. Assume there will be no more than 20 identification numbers, but your code should check that the file doesn't attempt to enter more. If an attempt is made to enter 21, process only the first 20.

hours: an array of doubles to hold the number of hours worked by each employee. Fractional hours are possible.

payRate: an array of doubles to hold each employee's hourly pay rate.

wages: an array to hold each employee's gross wages.

The program should relate the data in each array through the subscripts. For example, the number in element 0 of the hours array should be the number of hours worked by the employee whose identification number is stored in element 0 of the empId array. That same employee's pay rate should be stored in element 0 of the payRate array.

1) Calculate the wages for each employee and place into an array with the related subscript.
2) Write the unsorted arrays on the output file.
3) Prompt for and input from the keyboard an empID to use for the linear search.
4) Next sort the empID long integers using the selection sort algorithm. If an exchange is made for empID, be sure to make the corresponding exchanges in the three associated items.
5) Write the sorted arrays on the output file.
6) Prompt for and input from the keyboard an empID to use for the binary search.

The Selection sort algorithm, Binary Search and Linear Search algorithms should be three separate functions that will be called from the function main(). Modify the code in your text for this project. Do NOT use any output statements in these functions. You may use more functions in your design.

You may assume that the input file is constructed correctly, and will always contain -1 at the end. Be sure to test for an empty file (one that contains ONLY -1) and a file with more than 20 employees. Print an error message to the screen if there are no employees, and do NOT attempt to open the output file. If your file has more than 20 employees, just process the first 20 (without an error message.)

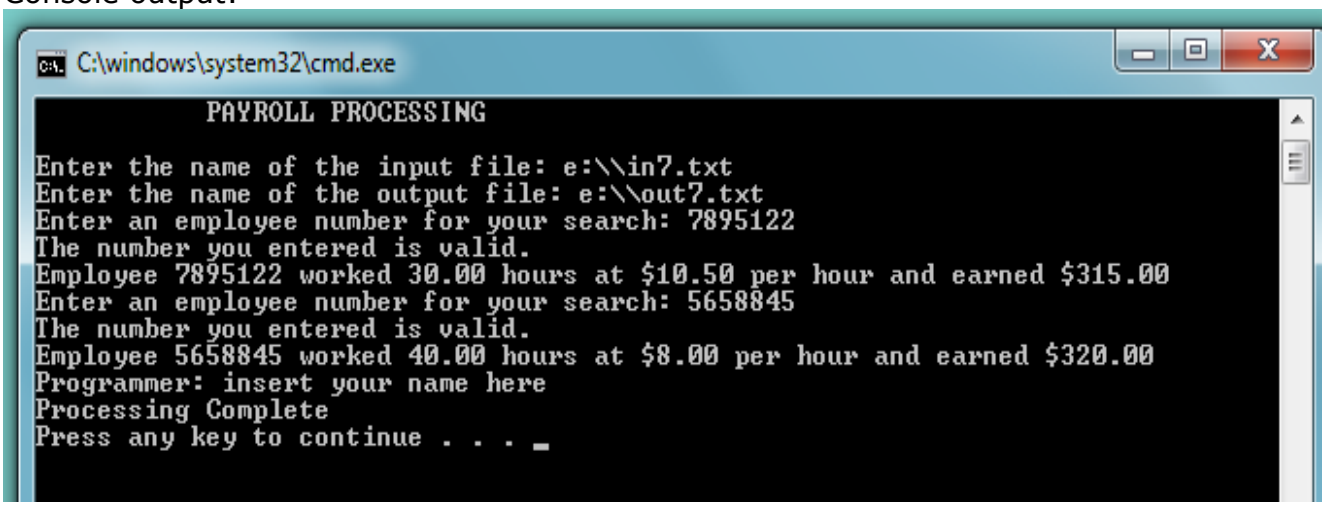
To input the file items without reading past the end of the file (-1 is the sentinel), use a while loop with two conditions: the empID in a temporary variable that is not negative and the number of employees less than 20. Inside your loop body: Store the "good" employee number, input the number of hours and the payRate in the arrays. The last item in your loop should be the next employee number from the file.

Sample input file:

in7.txt | out7.txt

```
5658845 40 8
4520125 25.25 9.25
7895122 30 10.50
8777541 10 12
8451277 50 10.20
1302850 35.20 7.50
-1
```

Console output:



Output: Your output should be in the format shown below:

in7.txt | out7.txt | in7S17.txt | payroll.cpp

```
                PAYROLL PROCESSING

Employees in order entered:

Employee Number      Hours Worked          PayRate per Hour        Wages
   5658845              40.00                    8.00           320.00
   4520125              25.25                    9.25           233.56
   7895122              30.00                   10.50           315.00
   8777541              10.00                   12.00           120.00
   8451277              50.00                   10.20           510.00
   1302850              35.20                    7.50           264.00

Employee numbers in ascending order:

Employee Number      Hours Worked          PayRate per Hour        Wages
   1302850              35.20                    7.50           264.00
   4520125              25.25                    9.25           233.56
   5658845              40.00                    8.00           320.00
   7895122              30.00                   10.50           315.00
   8451277              50.00                   10.20           510.00
   8777541              10.00                   12.00           120.00

Programer:  insert your name here
```

Project 7 Submission requirements:

1. A Word document that includes:
   - Screen shots showing sample test data. (at least 2 of each--screen and file--different from those given.
   - A flowchart showing your main function logic. Use a striped rectangle containing the name of the function for each function call.

- Your source code
- Output file
- Test plan (table) with at least 2 different data files (don't forget to test a file with only -1 in it and one with more than 20 employees.)

2. A zip file named LastNameFirstName_Project7_MOSS.zip containing ONLY payroll.cpp

Note: This format is required to check for duplicate submissions using "MOSS" Plagiarism Detection Software.

Your completed assignment should be submitted to the Blackboard assignment area no later than the due date.
You should include one block comment at the top of each program containing the course name, the project number, your name, the date and platform/compiler that you used to develop the project, for example

```
/*
 * Course: CMSC140 CRN XXXXX
 * Project 7
 * Instructor:
 * Description: (Give a brief description for Project7)
 * Due Date:
 * Platform/Compiler: (Windows 7, Microsoft Visual Studio 2013 for example)
 * I pledge that I have completed the programming assignment independently.
   I have not copied the code from a student or any source.
   I have not given my code to any student.
   Print your Name here: _____

     Pseudocode for algorithm design (show the logic in the main function)
*/
```

Grading Criteria for Project 7

The following are components on which the projects will be graded. If program does not compile, project will get grade "0". Contact your instructor prior to the project submission due date, if you have compilation issues.

Attributes	**Value** (points)
Functionality (If project does not compile, project will get grade "0")	Total 100
Displays the console input and report file output appropriately formatted	30
Calculates and displays the wages and the sorted columns correctly	30
Program executes correctly (produces expected output)	15
Meets all requirements	15
Overall Look-and-Feel	10
Total	Total 100 points

Project General Requirements (points will be deducted)

Attributes	Value(points)
Programming Style and proper naming convention: (see coding standards)	(-20 pts maximum)
Curt or unclear variable names	-5
Long variable names should use camel case or underscores to separate words	-5
Comments and internal notes	
Sparse and inadequate comments. (Missing with blocks of code, before functions, or with variable definitions)	-5
File header is not included (project description, name, etc.)	-5
Essentially no comments	-10
Indentation and white spaces should be a visual aid to understanding code structure	
Indenting is mostly okay, but sometimes inconsistent.	-5
No indenting, or very inconsistent indenting that is a barrier to understanding the code	-10
Lack of white space separating variables and operators. Lack of white spaces separating functions and major code blocks	-5
Test Plan	(-20 pts maximum)
Missing Entirely	-20
Cursory or inadequate testing (at least 2 different data files in table form)	-10
Missing Required Items	(-20 pts maximum)
Pseudocode (with code), Flowchart missing	-20
Screen shots cursory or incomplete (at least 2 different from examples)	-5
Screen shots completely missing	-10
An output file not included as a separate file	-5
Decimal output showing two decimal places	-5
Awkward Code Internal Structure	(-10 pts maximum)
Error messages are missing when input file is missing or no employees to process	-5 each
Incorrect parameters passed to functions or parameters missing	-5 each
Program title, "processing complete", or programmer name missing from console or file	-5 each
Poor structured programming: more than one “return” statement at end of each function ("break" and "continue" used)	-5 each
Code reads past -1 sentinel	-5
Other poor coding practices not mentioned	-5