CSC112 Spring 2011
Fundamentals of Computer Science
Lab 7 – Structures

Create a directory named Lab7. Keep all of your source and compiled programs in the directory Lab7.

Structures

Write a C++ program that does the following:

1. Your program reads a text file named "workers" and stores the information into a list (using an array) of employees. For this project, assume there can be at most 100 employees. This assumption allows us to simplify the project by using fixed-length arrays instead of dynamically allocated arrays. You can download a copy of this file from:

   `http://menehune.opt.wfu.edu/CSC112/Lab7/workers`

   - The text file consists of a sequence of employee records.
   - Each employee record consists of several comma-separated fields; there is one employee record per line.
     - Last name
     - First name
     - Company ID number. This is an integer.
     - Hourly pay rate. This is a floating point number.
     - Number of hours worked. This is a floating point number.

     When you read each name, you will need to read the name into a fixed-length array. You can assume no name is longer than 39 characters for this purpose. Remember one space in the array is needed for the null character to terminate the string. Use the library function `strdup()` to make a copy of this string. You can assign the pointer returned by `strdup()` to the lastname and firstname members of the employee structure.

2. Print a report listing each employee's name and their pay for this period. Pay is determined by the following rules:

   (a) Every hour up to and including 40 hours is compensated at the employee's pay rate.

   (b) Every hour above 40 hours is compensated at 1.5 times the employee's pay rate.

   To illustrate that your program is storing the names correctly (i.e., no leading or trailing spaces, no commas in names), print single quotes around each first and last name as part of your program output.

Structure Definitions

Use the following structure declarations to implement your program.

```
const int max_num_employees=100 ;

struct employee {
   char * lastname ;
   char * firstname ;
   int   id ;
   double pay_rate ;
   double hours    ;
} ;

struct emp_list {
   int n ;   // How many employees in the array.
   struct employee workers[max_num_employees] ;
} ;
```

In the main program, declare a variable of type **emp_list** as follows:

```
struct emp_list personnel ;
```

Notice that the `lastname` and `firstname` data members of the structure above are pointers. You will need to use the library function `strdup` to copy each string you read into a newly allocated block of memory. Relax. The function `strdup()` does the memory allocation and the copying for you.

Sample Worker File

```
 Smith , John ,       7,   15.65,  40
  Doe , Jane,         8,   15.65,  44.5
Jones , William ,  11,   20.90,  35
Bell, Susan  ,     12,   20.90,  37.5
Davis, Bob,        19,   11.50,  40
```

As you may notice, there are some extra spaces between the comma separated fields, both before and after some names. Your program MUST filter them out when you build the list of employees. I.e., the first worker's name needs to be stored as "Smith", not " Smith " or " Smith ,". To do this, write two functions:

```
void delete_leading_spaces( char name[] ) ;
```

and

```
void delete_trailing_spaces( char name[] ) ;
```

Pseudo-code for Reading the Worker File

Declare local arrays Lname and Fname to hold each last and first name as you read it.

```
loop = true ;  // Loop control variable.
while (loop) {
    Use getline with the delimeter ',' to read the last name (Lname).
    delete_leading_spaces( Lname ) ;
    delete_trailing_spaces( Lname ) ;
    loop = NOT end-of-file for the input stream.
    if (loop is still true) {
       Use getline with the delimeter ',' to read the first name (Fname).
       delete_leading_spaces( Fname ) ;
       delete_trailing_spaces( Fname ) ;
       Use the stream operator >> to read the id
       Skip to the next comma (including the comma itself)
       Use the stream operator >> to read the rate
       Skip to the next comma (including the comma itself)
       Use the stream operator >> to read the number of hours
       Skip to the end of line (including the newline character)

       Add the information you have just read into the list of employees.
    }
}
```

Turn in: Change to the directory containing the sub-directory "Lab7" Create a file named "lab7.tar" using the command:

```
tar cf lab7.tar Lab7
```

Upload the file "lab7.tar" to your account on telesto.