Assignment 4 (100 points)

Description
The ABC Company manages a pool of part-time employees. Up until mid-2004, it had a homemade custom accounting system with very simple data files. That year they upgraded to an industry-standard package and archived old printouts of their payroll records. Then they deleted the old accounting system. Last week, one of their new employees was cleaning out a closet, and, you guessed it: threw out some of the old printouts: those from the first half of 2004. It is your job to reconstruct them. They don't need anything fancy: just a list of the employees and how much they were paid during the first six months (Jan to June) of 2004.

Data for the period is kept in two types of files: the *department files* which have records describing employees in the department and their pay rates, and the *payroll files*, which contain information about what employees worked in a particular week and how much time they worked. The department files are named for the department number. This consists of a **D** or **E** followed by two or three digits.
Each department file contains records of the form
Employee Name:Department:Employee ID:Pay Rate:Overtime Pay Rate
Here is a sample record from the department file **D433**
Bennett,Fina:D433:181:20.40:30.00

The payroll files are named for the date of the pay period. Each date is a six-digit date in the format **MMDDYY**. Your program should only assemble data from payroll files named for the first six months of 2004. The payroll files contain records of the form
Employee ID#Hours Worked#Overtime Hours Worked#Date

Here is a sample record from the payroll file **010304** (that happens to be for the above employee)
181#96##010204
(employee **181** worked **96** regular hours and no overtime hours in the period ending Jan 2, 2004.

Note that the date field in this record is for internal company use and should be ignored.)
Your program must take these files and produce a listing that contains

- the total amount paid to each employee by name and employee id.
- the total amount paid out. This should be broken down as totals paid in regular pay and overtime pay.
- the total number of employees paid.
- if there are any payroll records that do not have a corresponding department record, they should be output in error messages.
- if you find duplicate payroll records (duplicate copies of the same record), you must eliminate them from the calculations.

Besides programming constructs, you will need to do floating-point arithmetic as part of this process. As you know, shell arithmetic is limited to integers. You must use an external program such as **bc** to do floating point arithmetic. **bc** is a simple calculator. You invoke it, then interactively type expressions to it and it outputs the result. You can also use **bc** from the command line if you like. I suggest you experiment with **bc** on your own. Type **quit** to exit **bc**. Here are two ways to calculate the quantity **14.35*24.10** using **bc**:

```
bash$ bc
14.35*24.10
345.83
quit
bash$ # scale=2 for max of 2 digits to the right of the decimal point.
bash$ echo "scale=2; 14.35*24.10" | b
345.83
bash$
```

Begin by looking at the data files: both the department files and the payroll files. They are in the **asmt04/payroll** directory. The names follow the pattern described. Note that there are some other
files in the way.
You will probably want to make some assumptions about the names of files with payroll records to limit the amount of work you have to do. You should document these assumptions. payinfo is a command line program. It should never ask the user questions.

Procedure
Your program's name must by **payinfo** It takes two arguments: **-d path** where **path** is the path to the directory containing the data files. You should do appropriate checking on the arguments before you continue.
Below is a suggested algorithm for this program. This algorithm contains some restrictions on how you may proceed. Other than these restrictions, you are free to use the algorithm or write your own.

I highly suggest that you work out these steps at the command-line before you write your shell script. In this part I have outlined these steps. Once you know how to do them, you can begin writing the shell script.

Begin by creating two files: one containing the department records, and the other containing the payroll records. You must create each file by combining the contents of the appropriately-named files with the **cat** program. (Hint: use wildcards to specify the filenames.) You may not copy the individual files -leave them where they are. Your program must stay in the directory it was in when it was invoked (i.e., it may not use the cd command.)

Step 1. Generate a temporary file that contains all the legal payroll records. (Combine all the payroll records from the files with legal names). *Hint: if this file is empty, it means there are no payroll records in the directory you were given. You should abort with an error to this effect.*

Step 2. Generate a temporary file that contains all the department records for the legal departments. (Again, combine the contents of the files whose names are **D** or **E** followed by two or three digits.)

When you write your shell script, of course, if you could not create the temporary files or if there were nolegal payroll or department files you have a problem and should stop.
Make sure you examine the contents of the files you produced. If you weren't careful enough in selecting which files to include, you will get very strange looking messages in the file. You should refine the wildcard patterns you used to match the files and redo the step.

Using your file of payroll records and your file of department records, do the following:

Step 3. Generate a list of all the employee ids that had payroll records in the period. The list should be sorted numerically, should only have employee ids, and have exactly one copy of each employee id paid.

Now you will generate the payment information for each of the employees in the list. In your shell script, you will use variables and command substitution to pass information from one step to the next and a loop to process each employee id in your list. When you are doing this by hand, select one of the employee ids and examine the output of the commands you use as you solve each of the succeeding steps.

Step 4. Get the department record for that employee only. This includes the pay rates for the employee as well as their name. If there is no department record for that employee, you cannot produce the pay information for them. When you implement your shell script, this should be indicated by an error message and the employee should be skipped. (If you are doing it by hand to get started, pick another employee)

Step 5. Get the payroll record(s) for that employee only.

Step 6. Using the payroll records for that employee, add up the number of hours worked for regular pay and the number of hours worked for overtime pay.

Step 7. You will now calculate the amount the employee earned using the total of the numbers of hours and overtime hours worked and the employee's pay rate and overtime pay rate. You must use **bc** to do this and to produce a total.

Step 8. Using **echo**, output a message on the screen that contains:
• the employee's name
• the employee's employee id
• the total amount paid to the employee during the period

Once you can do this for one employee, you are ready to write your shell script. You must also implement the running totals to output the summary information at the end of the program.
Hint: this program lends itself very well to several *for loops*.
Note: you may find some duplicate payroll records in the data. Remove them.