**COMP 2766: Introduction to Linux**

**Assignment #5: Pipelines and Regular Expressions**

Download the employee.tbl, payroll.dat, and student.lst files from Learning Hub module 7 and use grep -E to answer these questions.

14, 16 et 17

**IMPORTANT: Your answers must be generalized and work for any records that might be added to employee.tbl, payroll.dat, and student.lst, not just the current data set.** **Assume that the data for any newly added employees/students will appear in the same format as the existing data in those three files. For testing, you can and should add additional records to the files.**

**UNLESS SPECIFIED OTHERWISE, you may only use commands and techniques that have been explained in this course, either in the lecture, lecture slides, or the NDG Linux Essentials Course. Failure to follow this results in a mark of 0.**

***IMPORTANT: For each step, insert a screenshot showing the command line that you used and the output (or, at least, the first 10 lines of output); pipe to the head command to obtain the first 10 lines. NO SCREENSHOTS, NO MARKS.***

***When doing this assignment, login as the user based on your first name and last initial (ex: justint). Failure to do this will result in a mark of 0.***

You might need to use a grep switch to do some of these questions. Here are some common switches:

-c : output the quantity (a count) of the lines that match the regex

-i : do a case insensitive search; that is, uppercase, lowercase or mixed case

-l : instead of outputting the matching lines, output the names of the files containing the matching lines

-r : recursive search; by default, (e)grep only searches the present working directory

-v : invert the search; that is, output only the lines that do **NOT** match the regex

The student.lst file contains one row per student and five fields that contain, in order, from left to right: (1) student ID, (2) full name, (3) program of study, (4) age, (5) GPA.

Give a (that is, one) command line that outputs only:

1. (**1 mark**) the entire line for Linda Hui from student.lst

grep -E "Linda Hui" student.lst

1. (**1 mark**) the entire line of only students in student.lst who are ***NOT*** in the ELEC program

grep -E ",ELEC," student.lst

The employee.tbl file (***NOT emp.lst***) contains one row per employee and six fields that contain, in order, from left to right: (1) employee ID, (2) full name, (3) position, (4) department, (5) birthdate in MM/DD/YY format, (6) salary.

Give a (that is, one) command line that outputs only:

1. (**1 mark**) only the quantity of employees in employee.tbl who work in sales (***NOTE***: your output ***MUST*** be only a single number; also, sales can be an employee’s surname, as in the late American comedian soupy sales, and sales can appear in the position field in the title salesman)

cut -d\| -f4 employee.tbl | grep -E "sales" | wc -l

1. (**1 mark**) the entire line of only employees in employee.tbl whose ID does ***not*** start with the last ***non-zero*** digit in ***YOUR*** BCIT student #. You ***MUST*** use a relevant grep switch to solve this.

grep -E "^[0-9]{3}1" employee.tbl

1. (**1 mark**) the entire line of only employees in employee.tbl whose ID does ***not*** start with an odd digit character. You must ***NOT*** use an grep switch to solve this.

grep -E "^[^13579]" employee.tbl

1. (**1 mark**) the entire line of only employees in employee.tbl born in the 1960’s

grep -E "/6[0-9]\|" employee.tbl

1. (**1 mark**) the entire line of only employees in employee.tbl whose employee ID does ***not*** end with the last digit in your BCIT student number

grep -E "^[0-9]{3}[^1]" employee.tbl

1. (**1 mark**) the entire line of only employees in employee.tbl who have the position g.m. (**NOT** d.g.m.); your solution must also ***NOT*** match a position called gamer

grep -E "\|g.m.[ ]{5}\|" employee.tbl

To solve problems in Linux, we often use a pipeline that consists of one or more pipes and multiple commands. ***When using a pipeline, give the name of the relevant file as an argument only to the first command in the pipeline***. Do ***NOT*** do something like this:

command1 /etc/passwd | command2 /etc/passwd

1. (**4 marks**) Using a pipeline, give a single command line that outputs only the name and salary of the two lowest earning sales staff in employee.tbl in reverse alphabetical order by name.

sort -t\| -n -k6 employee.tbl | head -2 | cut -d\| -f2,6 | sort -r

The payroll.dat file contains weekly payroll data that is colon delimited. In order, from left to right, the fields contain (1) employee ID, (2) first name, (3) hours worked and (4) hourly wage:

5203:Pam:50:9.50

1425:Juan:18:14.25

4321:Ben:22:21.11

6781:Anna:44:16.77

1451:George:36:21.77

2277:Tuan:16:18.77

***Using gedit, vi, or nano, add a new line to the end of payroll.dat that contains the last 4 digits of your student ID, your first name, 50 as the number of hours, and 25.00 as the hourly wage. Failure to do this will result in a mark of zero.***

1. (**2 marks**) Using a pipeline, give a single command line that outputs only your hourly wage. Note that employees who share your first name could be hired in the future.

grep -E "^0991" payroll.dat | cut -d: -f4

1. (**3 marks**) Using a pipeline, give a single command line that outputs only the name and hourly wage, separated by a comma, of employees who worked 50 or more hours, sorted in reverse alphabetical order, by name. Assume that employees work less than 100 hours per week. Your answer must be generalized so that it works for ***any*** new data that might appear in the payroll file. (**HINT**: The grep command has no concept of what a field is and, thus, no field switch.)

grep -E ':[5-9][0-9]:' payroll.dat | cut -f2,4 -d: | sort -rk1 -t:

1. (**4 marks**) Give a single command line that outputs only the lines in /usr/share/dict/words that begin with a vowel (lowercase or uppercase) and are followed only by consonants (lowercase or uppercase). Vowels are the letters a, e, i, o, u, A, E, I, O, and U. Consonants are letters that are not vowels. Examples of such words include: abyss, Egypt, inn, onyx, UBC, etc. but ***NOT*** act**o**r, ey**e**, **F**lynn, US**A**. **Remember, words in this and other dictionaries contain characters that are not letters.**

grep -Ei "^[aeiou][^aeiou]\*$" /usr/share/dict/words

1. (**4 marks**) Your organization has developed a new product code whose format appears as follows: *MMM*-*NNN*-*ZZZZ*

MMM consists of 3 digit characters representing the numeric range 524-973, inclusive.

NNN consists of 1 to 3 digit characters representing the numeric range 5-482, inclusive.

ZZZZ consists of 4 digit characters representing the numeric range from 1984-7215, inclusive.

Using (e)grep, give a single command line that outputs only the names of the visible files anywhere under the / directory that contain a product code of the form described above. For example, 575-9-4415, 632-15-5718, and 791-124-6017 are all valid product codes. Examples of invalid codes include

You will need to create your own sample data for testing purposes. NOTE: The regular expressions is long.

^((52[4-9])|(5[3-8][0-9])|(59[0-9])|([6-8][0-9]{2})|(9[0-6][0-9])|(97[0-3]))-(([5-9])|([1-9][0-9])|([1-3][0-9]{2})|(4[0-7][0-9])|(48[0-2]))-((198[4-9])|(199[0-9])|([2-6][0-9]{3})|(7[0-1][0-9]{2})|(720[0-9])|(721[0-5]))$

1. (**2 marks**) Output in reverse alphabetical order only the names of the three eldest g.m.’s in employee.tbl. To figure out how to sort a date field, consult your best friend Google.

grep -E "\|g.m.[ ]{5}\|" employee.tbl | sort -k5.7 -k5.8 -k5.1 -k5.2 -k5.4 -k5.5 -t\| | head -n3 | cut -f2 -d\|

1. (**1 mark**) Write a regex that matches lines containing only a BCIT student number. A BCIT student number consists of the capital letter A followed immediately by exactly eight digits. Create your own data to test your regex.

grep –E “^A[0-9]{8}$”

1. (**2 marks**) There are many words whose British variant ends with tre and whose American variants ends with ter. For example, the Brits write centre while the Americans write center. Canadians usually adopt the British variant, though sometimes we follow the Americans. Write a regex that outputs all the lines in /usr/share/dict/words that consist of at most 10 characterrs and end in either tre or ter, lowercase.

grep -Ei "^[a-z]{1,7}(tre|ter)$" /usr/share/dict/words

1. (**2 marks**) Every printable and non-printable character is encoded using an ASCII (American Standard for Information Interchange) value which is converted into binary for storage on a computer. For example, uppercase A has an ASCII value of 65 which is stored on a computer as binary 1000001. ASCII values for all characters can be found in an ASCII table such as this:

<https://www.asciitable.com/>

Using the tr command, encrypt a given word (ex: helpful) by replacing each character in that word with the corresponding character whose ASCII value is ten less than that of the character being encrypted.

For example, the letter h (ASCII 104) would be replaced with the ^ symbol (ASCII 94), the letter e (ASCII 101) with [ (a left square bracket, ASCII 91), the letter l (ASCII 108) with b, etc.

Test your tr command line by using the echo command with the word that you want to encrypt as an argument and redirecting its stdout to a pipe as shown in Module 10, Lab 10 (NOT Chapter 10), sections 10.2.8., 10.2.9, and 10.2.10. Your screenshot must show at least three executions of your solution command line encrypting a different word each time.