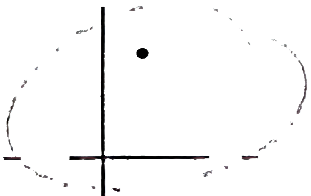
**Utility Regular Expressions**

|  |  |
| --- | --- |
| Special Characters | Function |
| [123] | Matches any one character 1,or 2,or 3 |
| [^123] | Matches any character except 1,2,and 3 |
| [1-3] | Matches any one character 1 through 3 inclusive |
| \* | Matches zero or more occurrences of the preceding character |
| ^Tom | Anchors the pattern to the beginning of the line |
| $ | Anchors the pattern to the end of the line |
| \ | Used to quote special characters such as \*, ^,$,etc. |
| t.e | Matches any single character,toe,the |

**Notes**

Regular expressions are special characters used by certain UNIX commands such as grep,sed,awk, vi,etc.



**Differences between the Shell regular**

**expressions and regular expressions used**

**within utilities**

|  |  |  |
| --- | --- | --- |
|  | Shell Special  Characters | Regular Expression Characters |
| Negation | ! |  |
| Matches single character |  |  |
| \* | Matches any string | Matches zero or occur-  rences of preceding |
| [ ] | Matches any single character in set | Matches any single charac-ter in set |
| Anchors begin-ning of line | N/A |  |
| Anchors end of line | N/A | $ |

**Notes**

Looks at the beginning of each line for all user names that begin with j.

$ grep "^j" /etc/passwd

· Finds all lines in the password file that end with/sh.

$ grep "/sh$" /etc/passwd

Changes each number in the password file to x.

$ sed "s/[0-9]/x/g" /etc/passwd

Finds all currently logged-in users whose names do not begin with the letters a through g.$ who | grep "^[^abcdefg]"

Deletes all characters following the first space in each line of the output of the who command.$ who | sed "s/ .\*//"

· Finds all words in words beginning with zeb.

$ grep "^zeb\*" words

Finds all words in words ending with tty.

$ grep "tty$" words

**Evaluate an Expression**

**Format:** expr expression



**Comparison Operator**

**compares a string to a regular expression**

**Arithmetic Operators**

**multiplicative** op \*

**additive operators** +-

**Relational Operators**

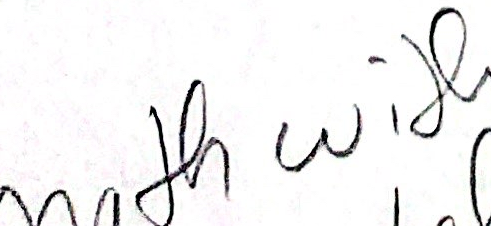
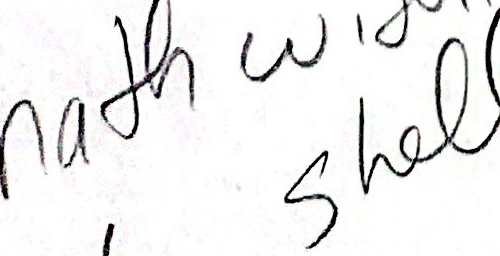
**relational** operators <><=>=

equality operators =!=

And, Or Operators &

**Notes**

expr evaluates an arithmetic operation and displays the result. Performing arithmetic operations on variables is only possible by using expr. It evaluates character strings that represent numeric or non-numeric values. Operators separate the strings.

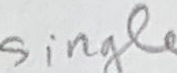
 



**Notes on expr**

· All terms of the expresion must be separated by blanks.

· The special characters:

& <> \* () 

must be preceded with a backslash or quoted.

·The arithmetic operators operate on strings that contain integers 0 through 9.

The relational operators work on both numeric and non-numeric arguments.

The & and | operators and parentheses must be enclosed in single quotes or preceded with a backslash.

&(and) displays 0 if one or both of the arguments are zero or the null string.

· |(or) displays the first operand if it is not zero or the null string, otherwise it displays the second operator.

All arguments to expr are character strings which expr converts to a numeric value when applicable.

Notes

**expr Examples**



**1.**

8

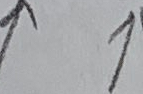


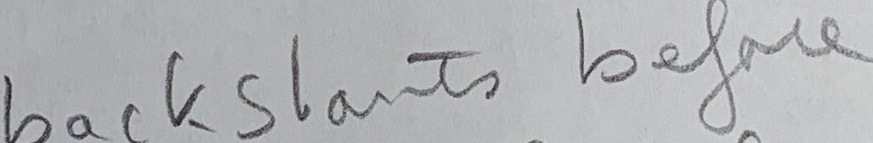
**2. Multiplies i by two:**

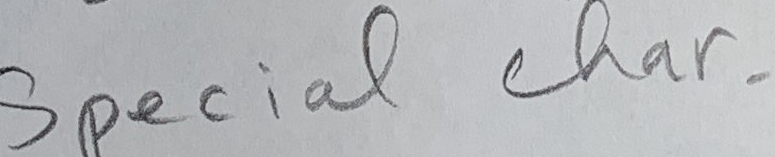
$ 

**3.**  

/2

2





**Notes**

1. expr can produce negative numbers by using the unary operator "-". Some systems do not support the unary operator.

2. Because \*has a special meaning to the Shell, multiplication must be quoted (or precede it with a backslash)

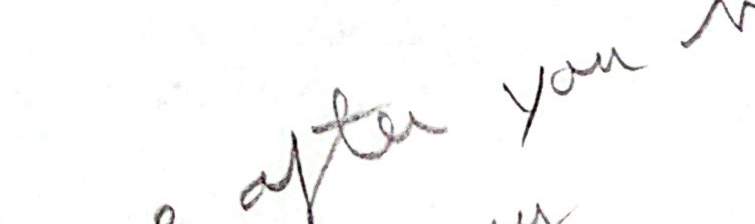
3. Parentheses can be used to change the order of operation. Each parenthesis must be quoted and surrounded with blanks.

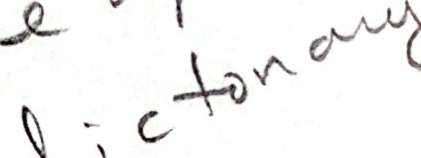
UNIX Shel1 Programming

**expr Examples (Cont'd)**

**4.** 5 

0



**5.** $ 

1

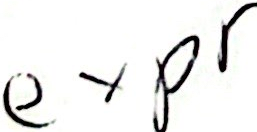
0

**6. Use & (and) and | (or) to compare numeric or non-numeric arguments.**

**Notes**

4. expr returns a 0 if two arguments are unequal and a 1 if they are equal.

5. Relational operators must be quoted. O implies false and 1 implies true.



**Summary**

**·Pattern matching characters provide a notation for describing** text when **searching in files and for filenames.**

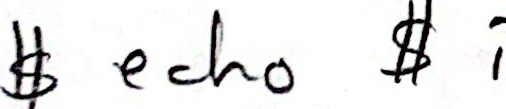
**Shell patterns are a limited form of regular expression. Many** UNIXutilities **use more capable regular expressions.**

**Regular expression arguments should be quoted so that the shell does not** **treat them as a file pattern.**

**The** expr **command can be used to match text against a regular** expression.**It is also used for doing arithmetic in the shell.**

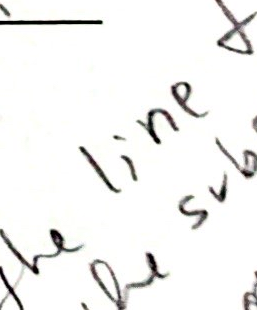
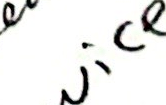
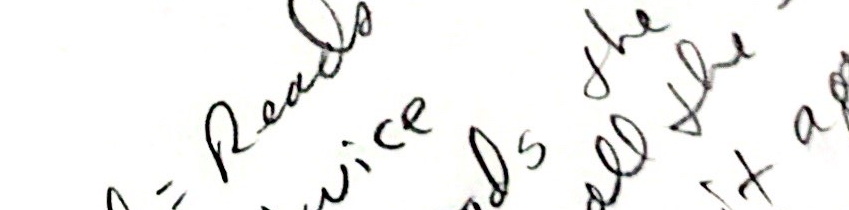
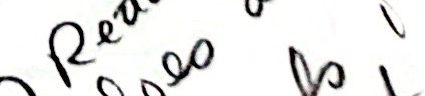
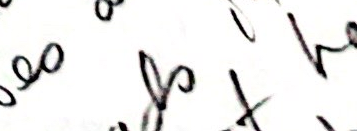
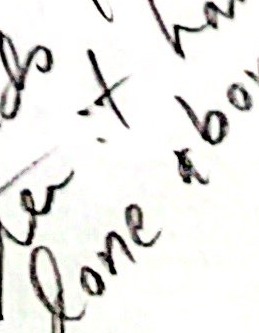






Notes

   8oeo ell fhe s  ha    



**Lab 7**

**Objective:**

Wild cards are used for Shell file name pattern matching as well as for pattern matching with certain applications such as vi, grep, sed, awk, and others. However, the same wildcards are not used in both instances. You will investigate both circumstances. Additionally you will see how Shell metacharacters affect the wide cards. Finally you will experiment with uses of expr.

**Exercises:**

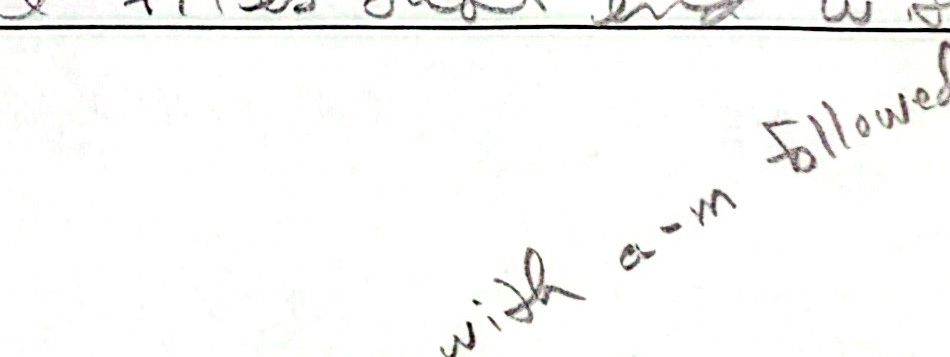
1. What files are matched with the following command?



2. Make the following assignments:

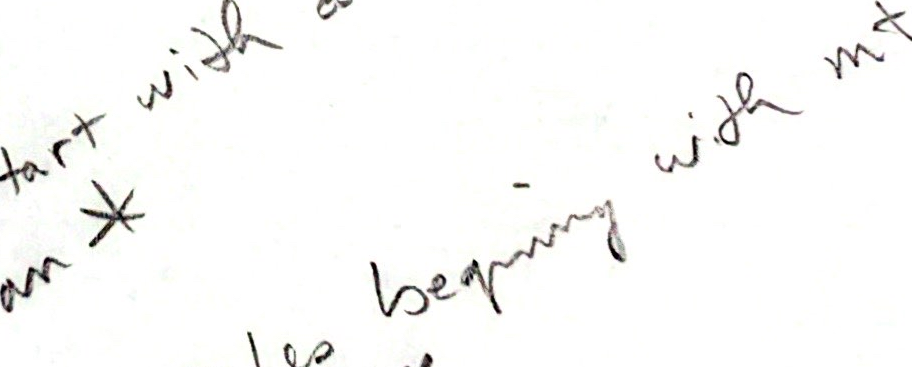
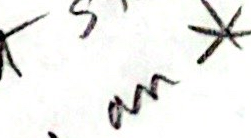


m Jones

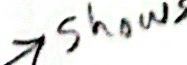


|  |  |
| --- | --- |
| Command | Matches |
| 1s -d.\* |  |
| 1s???\* |  |
| 1s [!a-m]\* |  |
| 1s \*[0-9] |  |

Atlanta,Ga 30342



Investigate the output from the following commands



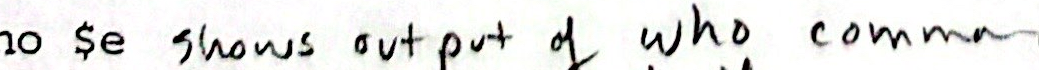
echo [a-m]$a echo [a-m]$b

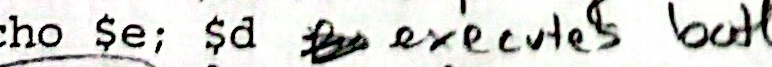
echo [a-m]\* echo [a-m]?

eval echo $d  

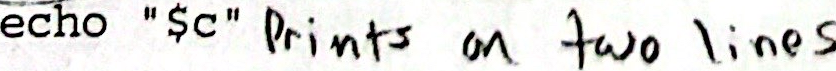
eval

$d

echo

echo

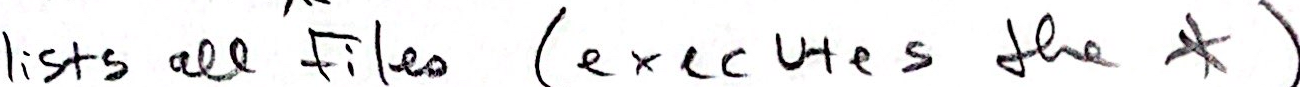
echo



echo 

echo 

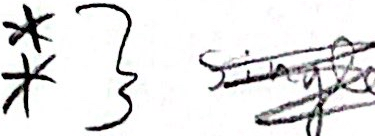
echo

echo $a  \*)

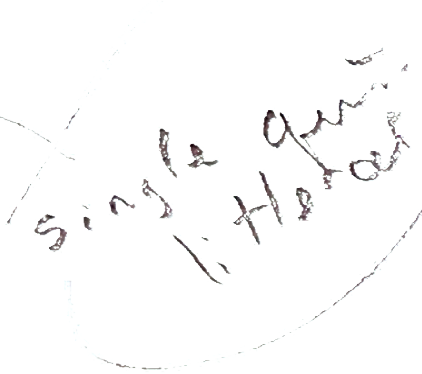
3. Discover the difference when using metacharacters.



echo \\*echo \*

echo "\*"echo '\* 

4. Run the following program and explain the differences in the output.

Shell Script:

echo no quotes: term is $TERM, files are \*

echo single quotes ' term is $TERM,files are

echo double quotes " term is $TERM, files are \*"

Explanation (Yes, No) Expanded Expanded

no quotes: wildcards  variables: 

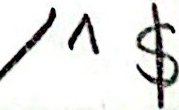
single quotes: wildcards variables:

double quotes: wildcards  variables: 

5. Use the utilities grep and vi to experiment using regular expressions: Copy the file why (location on the board) to your HOME directory.

Use vi

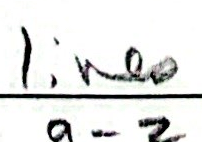
·Find all occurrences of t or T at the beginning of a line.

·Find all words meeting the criteria t.e.  

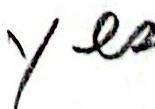
·Use a regular expression to find blank lines. How many are there?

3

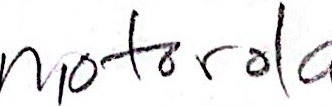
·What does the following match?

Use grep

·Is there any thing about dos in the why file? 

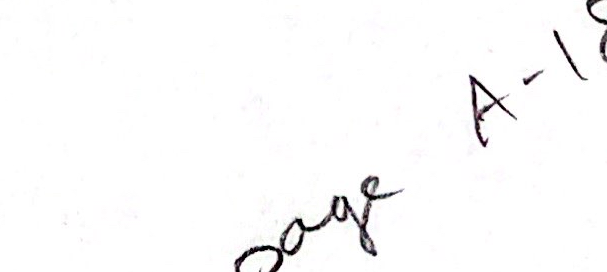
·Find all lines in the file wth references to integers. 

Find lines that end in rola. 

·Find all word contain at least 2 capital letters.

6. Evaluate the following using expr.





5(2+3)/10

tom>mary

tom=mary

27%2

6-3\*2-4

2<1 or 6>3

1<2 and 6>3

These are expressions in algebraic notation. You will have to use the correct operators for expr.

