CIS-387: Digital Forensics (4 credits)

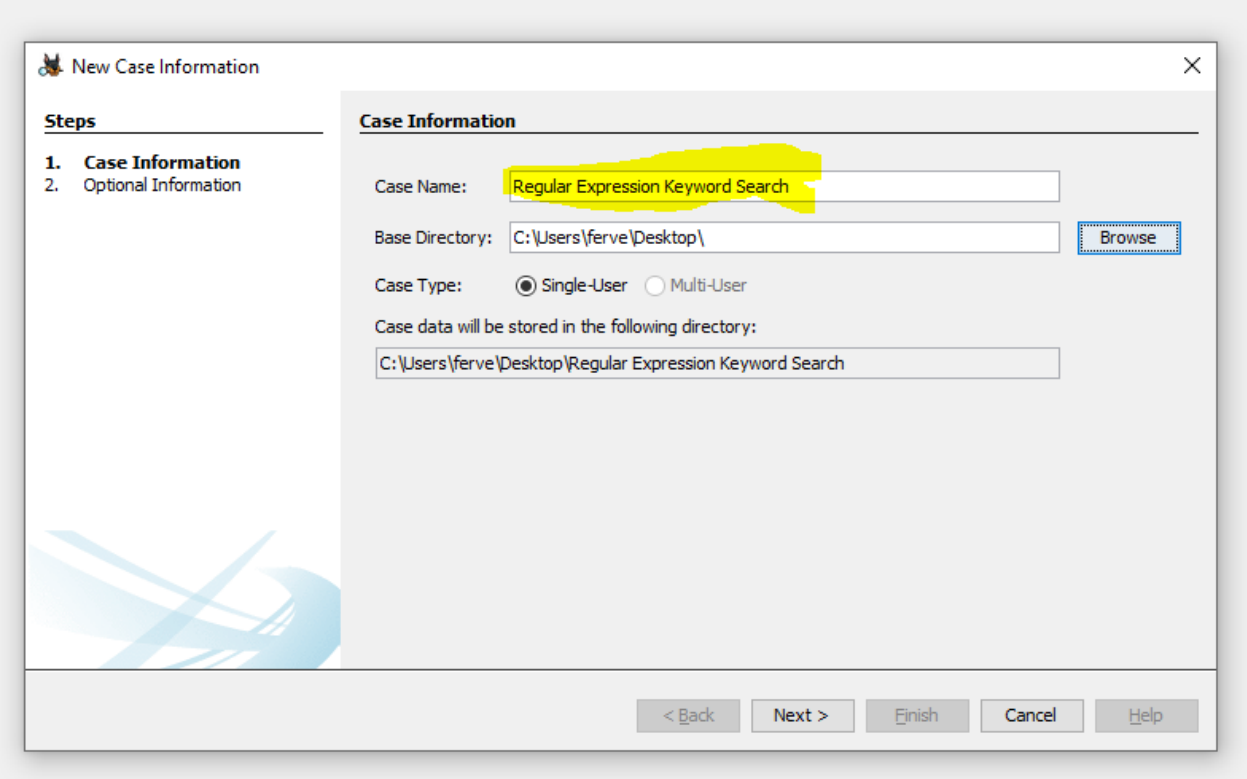
With Dr. Jinhua Guo

Lab 6

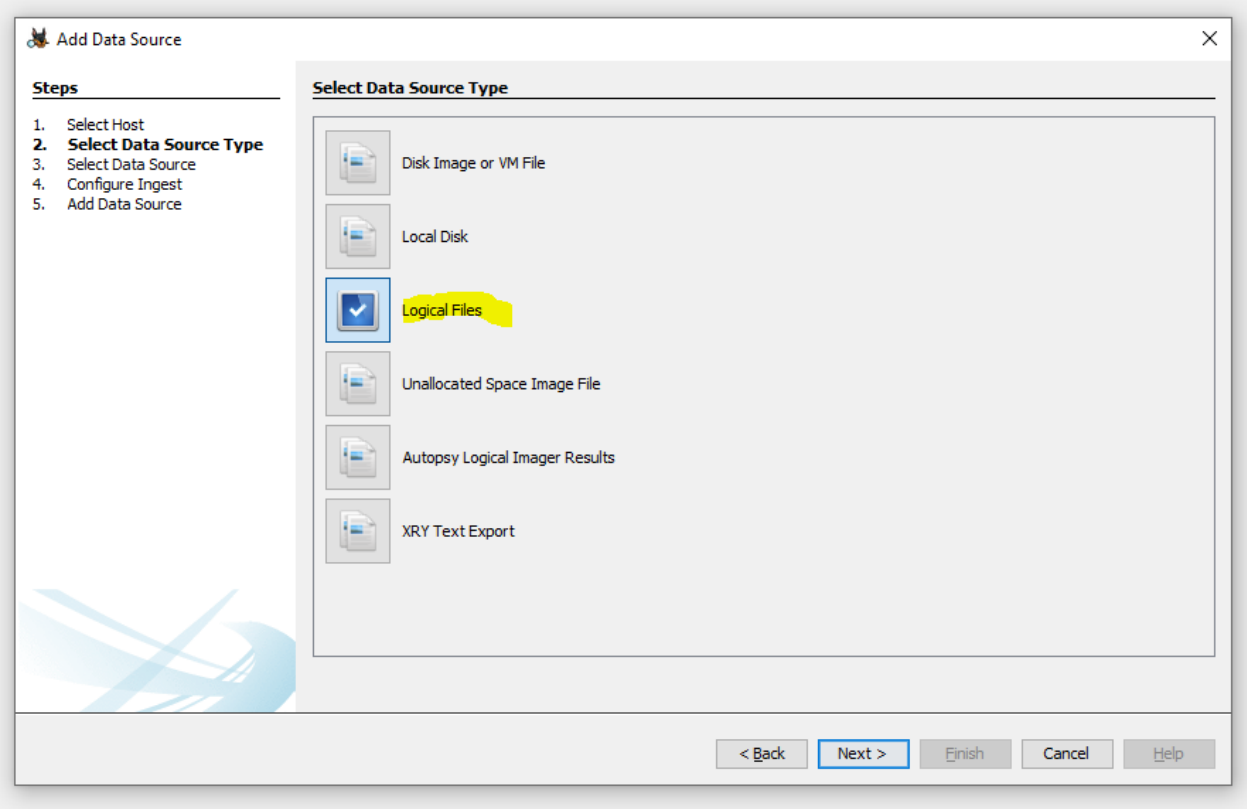
Demetrius Johnson

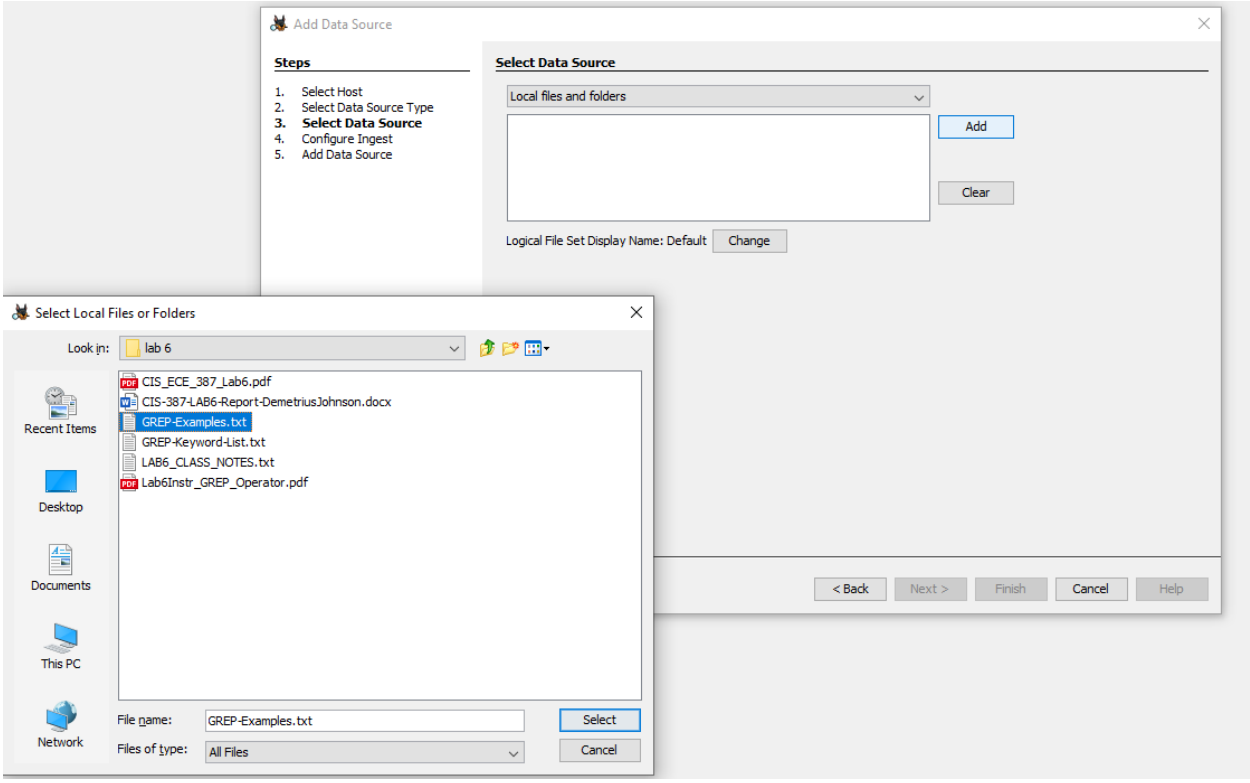
November 07, 2022

1. Launch Autopsy and create a case, Create New Case and name it as “Regular Expression Keyword Search”.

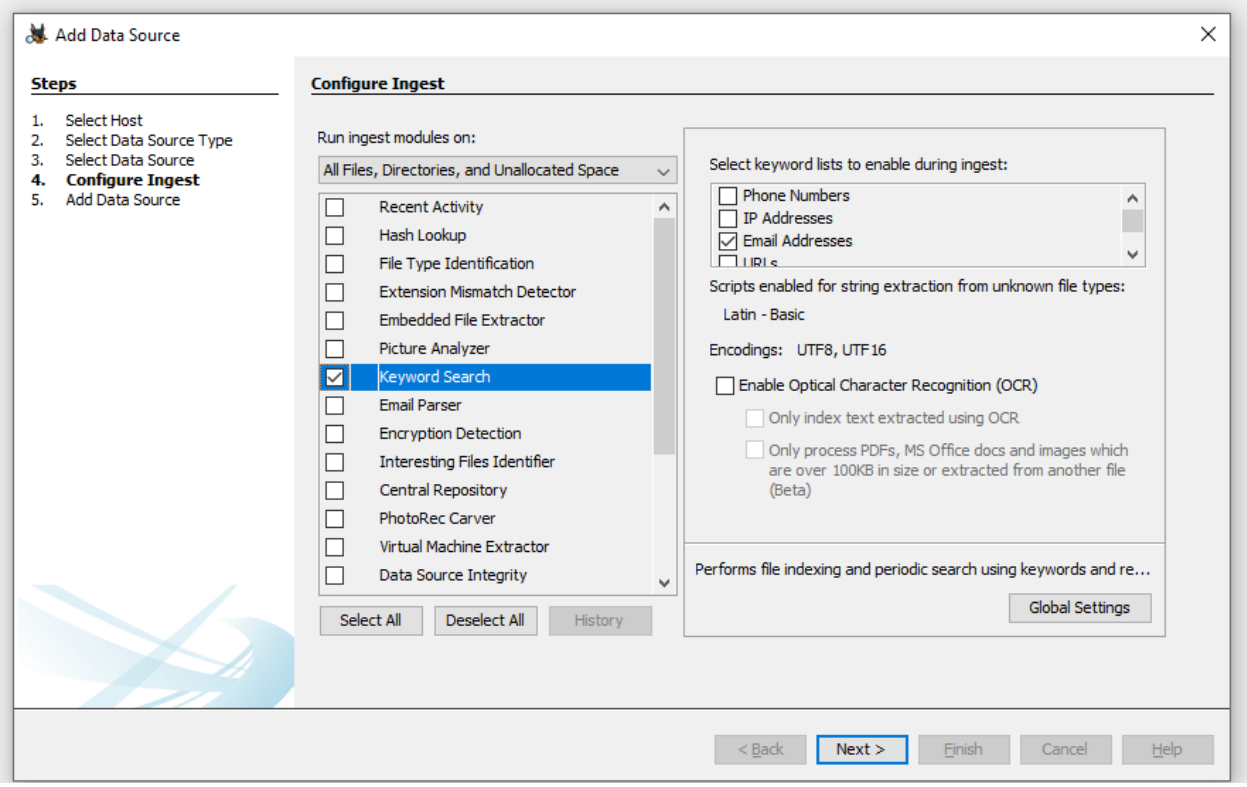


2. Add data source type: choose Logical Files; browse and select the path to "GREP-Examples.txt".

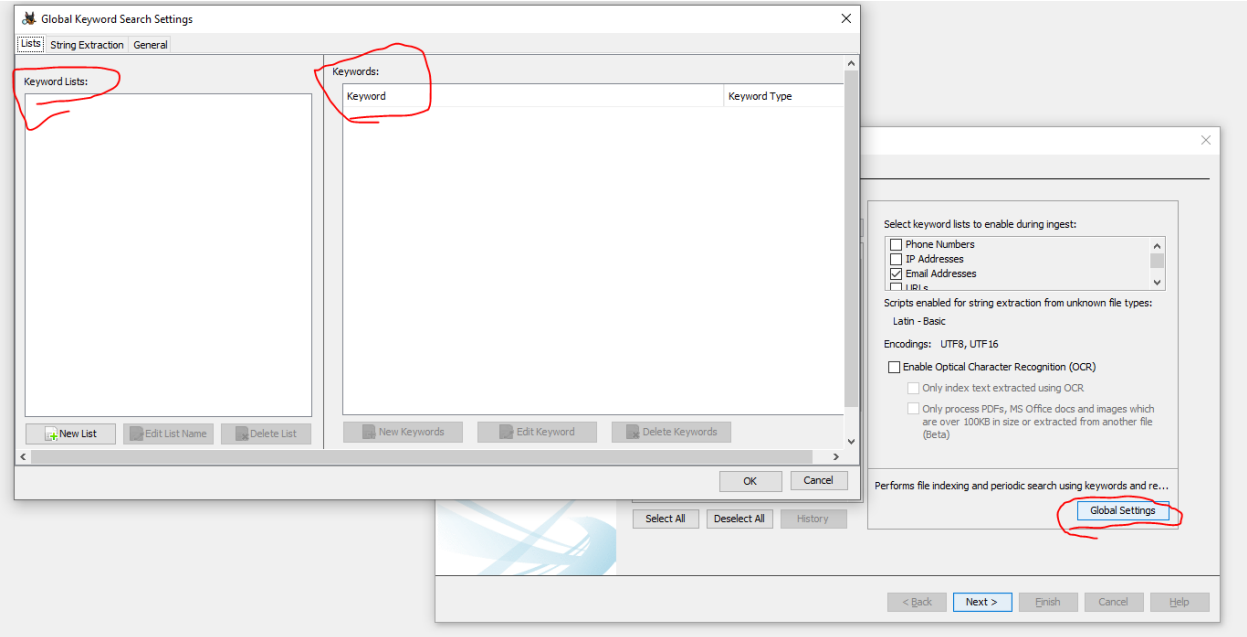




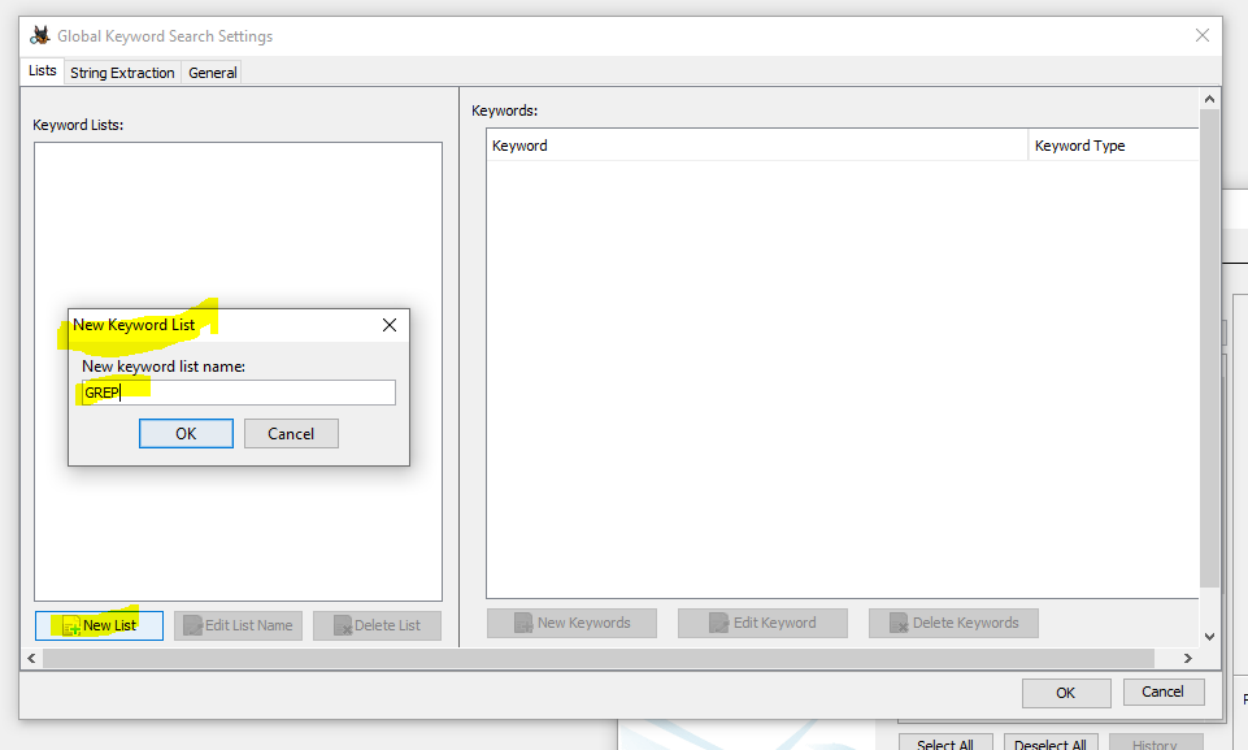
3. In the Ingest (processing) modules window, uncheck all modules except the “Keyword Search”;



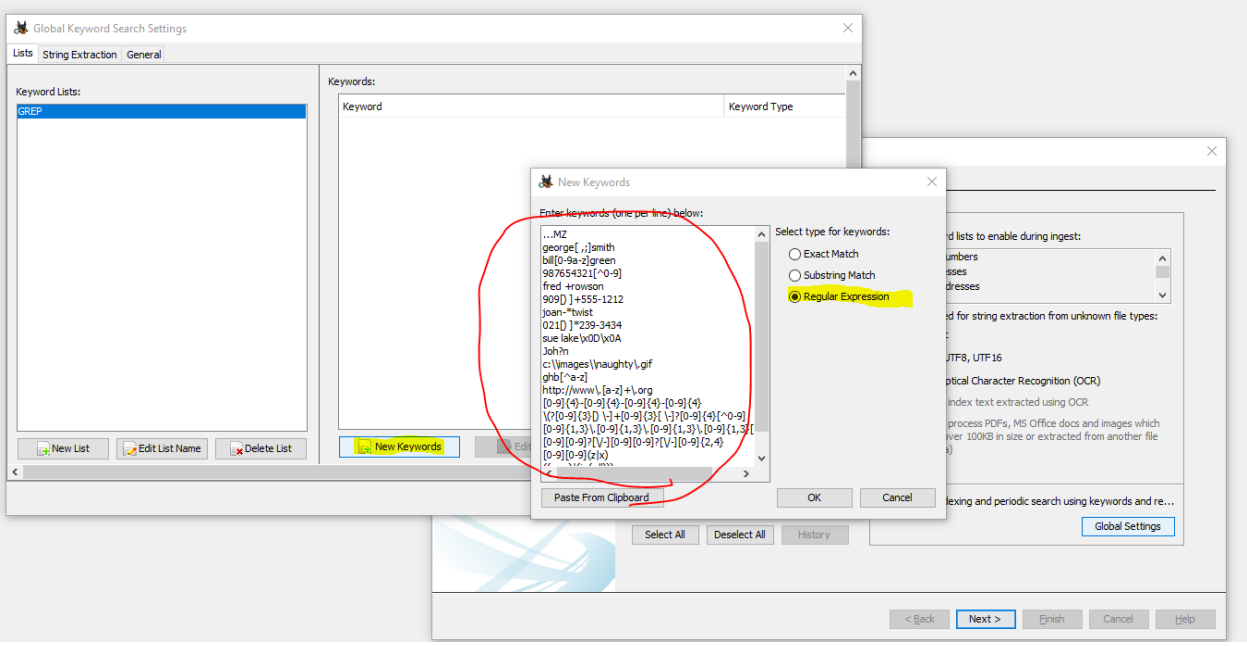
4. Click “Keyword Search” and then click Global Setting.



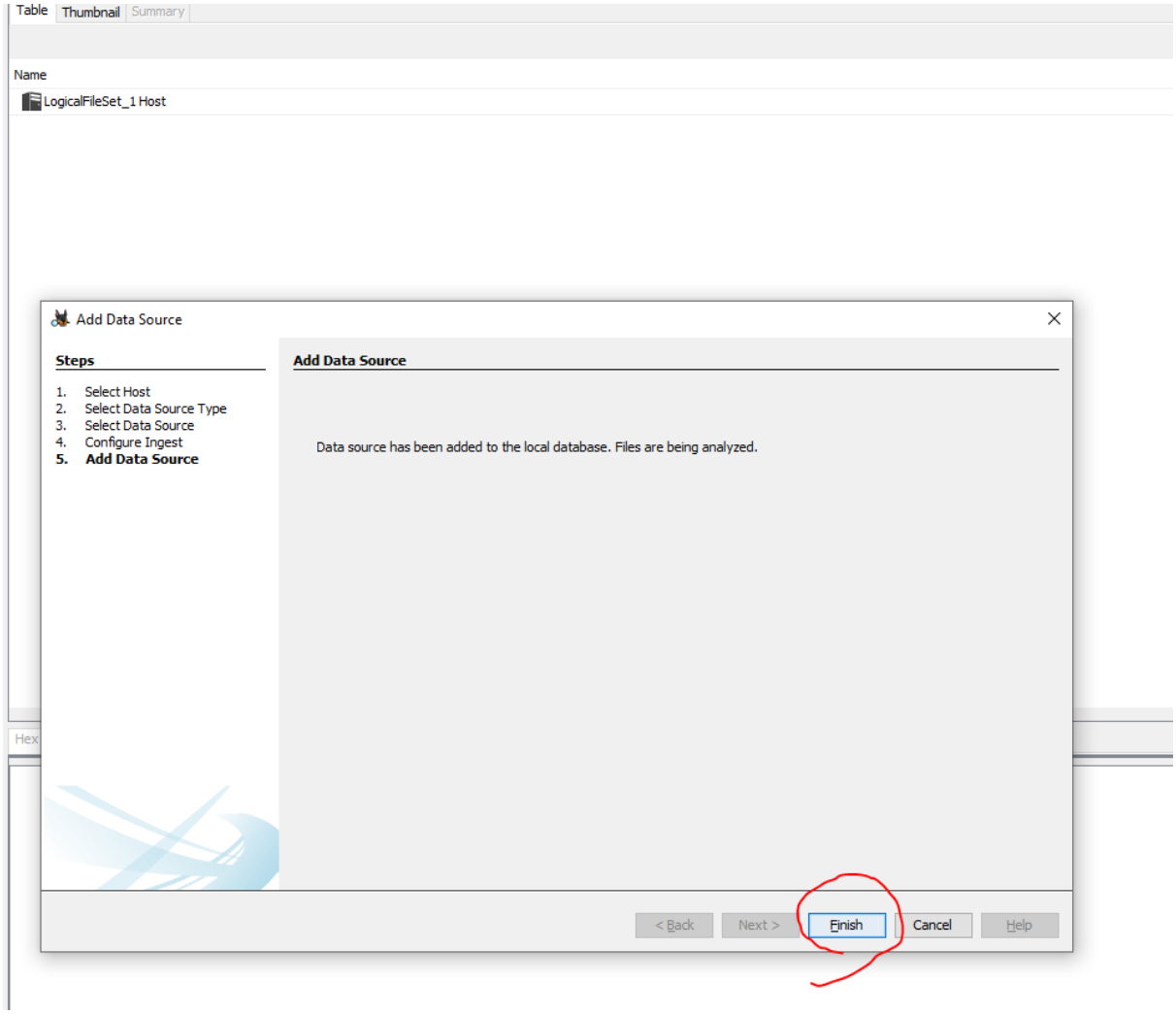
5. At the Global Keyword Search Setting window, click “new list” and input the “GREP” as the list name.



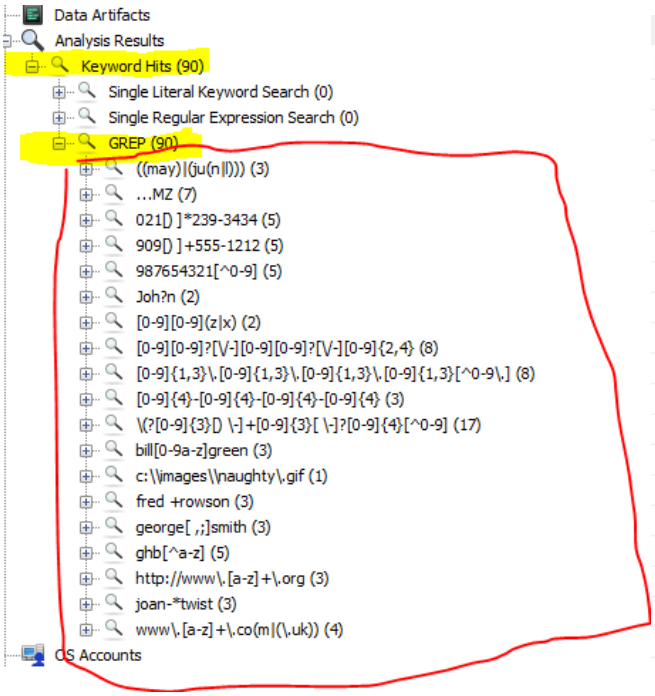
6. Click “New Keywords” and then copy and paste the keyword list from the “GREP-keyword-list.txt” file, and check “regular expression” button.



7. Click “OK” twice; click Next and then click Finish.



8. Review the search results under Results > Keyword Hits > GREP, and find all the matches.



List Name	Files with Hits
((may)|(ju(n|l))) (3)	3
...MZ (7)	7
021[]]*239-3434 (5)	5
909[]]+555-1212 (5)	5
987654321[^0-9] (5)	5
Joh?n (2)	2
[0-9][0-9](z|x) (2)	2
[0-9][0-9]?[\/-][0-9][0-9]?[\/-][0-9]{2,4} (8)	8
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}[^0-9\.] (	8
[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4} (3)	3
\(?[0-9]{3}[] \-]+[0-9]{3}[\-]?[0-9]{4}[^0-9] (17)	17
bill[0-9a-z]green (3)	3
c:\\images\\naughty\.gif (1)	1
fred +rowson (3)	3
george[,;]smith (3)	3
ghb[^a-z] (5)	5
http://www\.[a-z]+\.org (3)	3
joan-*twist (3)	3
www\.[a-z]+\.co(m|(\.uk)) (4)	4

9. Highlight all the matches in the logical evidence file (GREP-examples.txt) and embedded it in the report

GREP Examples . txt file:

The following examples show some of the power that GREP expressions give you when looking for text. The first line of the example, followed by an explanation of the tokens used, followed by examples of text found using the expression.

---

1 ...MZ

The '.' period matches any character, 00h - FFh. This expression finds_"MZ" proceeded by any three characters. This is an example of finding a file header.

ADMZ

ZMZ

ASDDEFJHMZKJYTRDF

02,MZ

,,-MZ

ABNMZLK

.!=MZ

MZ

---

2 george[,;]smith

The square brackets form a set. The set, absent any other GREP operator, controls a single character location, and defines within the square brackets what can exist in that single locaction. This expression finds "george" followed by a space OR a comma OR a semi-colon followed by "smith".

george ,smith

george smith

georgesmith

george,smith

george;smith

george ,;smith

george smith

---

3 bill[0-9a-z]green

The dash indicates a range of characters when inside a set. This expression finds "bill" followed by any character between ('0' and '9' or 'a' and 'z') followed by "green".

bill green

bill0green

bill20green

bill1green

billzgreen

billa-zgreen

bill,green

---

4 987654321[^#]

The '^' at the start of a set indicates any character other than those in the set.

This expression finds "987654321" followed by any character other than '0'-'9'.

Note: the expression [^#] is identical to [^0-9].

YM987654321AS

1298765432145

987654321

1111111987654321000000

1111111987654321A00000

....987654321.

---

5 fred +rowson

The '+' plus sign says repeat the preceding character (or set) any number

(maximum of 255 occurances) of times, but at least once.

This expression finds "fred" followed by any number spaces followed by "rowson".

fredrowson

fred rowson

fred rowson

fred +rowson

Fred Rowson

FRED ROWSON

---

6 909[)]+555-1212

The '+' is most commonly used after a set in more complex statements.

This expression finds 909 followed by a space or a parenthesis, or a space

and a parenthesis.

909 555-1212

909 555-2121

(909)555-1212

(909) 555-1212

(909)) 555-1212

909-555-1212

(909))) 555-1212

909555-1212

---

7 joan-*twist

The '*' star says repeat the preceding character (or set) any number of times

(maximum of 255) including zero. This expression finds "joan" followed by

any number dashes followed by "twist". This operator is commonly used after a set.

Joan Twist

Joan-TWIST

joantwist

joan-twist

joan---twist

---

8 021[)]*239-3434

The '*' is most commonly used after a set in more complex statements.

This expression finds 021 followed by 239, OR by a parenthesis or a space,

or both. This is the same as the '+' except it allows for zero characters.

021 239-3434

021)239-3434

021) 239-3434

021))))239-3434

021-239-3434

021239-3434

0212393434

---

9 sue lake\x0D\x0A

The '\' slash followed by an 'x' indicates a two digit hexidecimal

representation for a character. This expression finds "Sue Lake" followed

by a carriage-return linefeed sequence.

Sue LAKE

sue lake

sue lake.

sue,lake

sue lake

Sue Lake was a driving...

---

10 Joh?n

The '?' question mark says repeat the preceding character (or set) one

or zero times. This expression finds "Jon" or "John", with or without the 'h'.

John

JON

Jack

join

---

11 c:\\images\\naughty\.gif

The '\' slash preceding any character (including '\') indicates that this is a literal character and not a GREP symbol. Be careful when expressing file names and paths in GREP. Slashes and dots should be preceeded by a '\'.

C:\images\noughty.gif

c:\images\naughty.gif

c:\\images\\naughty.gif

c:\\images\\naughty\.gif

---

12 ghb[^a-z]

This expression matches "ghb" followed by any non-alphabetic character. This ensures that short names and words are not found inside other words.

neighborhood

thighbone

ghb

highboard

Get some GHB!!!

5.GHB,

---

13 http://www\.[a-z]+\.org

This expression matches "http://www." followed by any alphabetic characters

followed by ".org". This is a good way to look for web site references.

http://www.bozo.net

http://www.theonewiththeverylongsitename.org/index.htm

http://www.to-wong-foo.co.fr

http://microsoft.msdn.au.com

http://www.bozo.org

http://www.the_one_with_the_very_long_site_name.org

---

14 ####-####-####-####

The '#' character matches any number. This expression matches a credit

card number with the numbers separated by dashes.

6666-6666-6666-6666-6666-6666

1234-3623-3410-2232

4534-2123-9866-651

4534212398666512

1233456780007654

456

---

15 [456]###-?####-?####-?####[^#]

This expression matches a credit card number with the dashes between the numbers

being optional and the first number being constrained to either 4, 5 or 6.

6234-3623-3410-2232

4534212398666512

6666666666666666

4444444444444444

1233456780007654

323345680007654

---

16 \(?###[) \-]+###[\-]?####[^#]

This expression matches a US phone number in one of several formats.

The (* expression means that the open paren '(' character can be present or not.

The [) \-] expression means that either a space or a close paren or a dash can be present or not.

(909) 875-4125

(123)-255 1700

204-725-2436

103 875 4344

345)))--4562134aDE/}

9098721344

---

17 ##?#?\.##?#?\.##?#?\.##?#?[^#\.]

This expression matches an IP number in regular form with 4 (up to 3 digit)

numbers separated by periods.

123.235.23.1

255.255.255.255

0.0.0.0

512.79.789.367

234.1234.123.123

0.0.0.0.

12..12.34.2

---

18 ##?[/\-]##?[/\-]###?#?

This expression matches a date in regular form with a 4 digit year and either one or two digit months and days separated by either forward slashes or dashes.

13/3/1999

13/03/1999

14-2-1998

14-02-2000

1-2-02

01/02/02

34/19/5079

73-49-3030

01-Feb-2002

3-Dec-2003

---

19 ##z|x

The vertical bar character '|' is used as a logical OR. This search expression will find any two digits (0 through 9) followed by the character 'z' or the character 'x'

abc

cba

123zbc

321cba

432xbk

234gbn

---

20 [\-/](may)|(jun|l)[\-/]

This expression looks for a literal space or hyphen or slash followed by

either 'May' or, 'Ju' followed by either an 'n' or an 'l', followed by another space.

3rd Dec 2002

20th May 2003

1-Jun-03

14 Jul 2003

11-Apr-2004

10 Dec 01

9/jun/01

---

21 From ?:.{20,200}To ?:

This expression will search for email headers by finding the word 'From'

followed by zero or one space character and then a colon, followed by

between 20 and 200 of any character, followed by the word 'To' followed by

zero or one space character and then a colon.

From : sharren_redmond@yahoo.co.uk

To : larissa_moore@hotmail.com

Subject :Did I bother you?

Date :Wed, 26 Mar 2003 10:25:14 +0000

Received: from bellsouth.net ([194.232.78.135]) by mc6-f18.law1.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600); Tue, 22 Apr 2003 02:33:13 -0700

Received: from [214.112.45.243] by bellsouth.net with esmtp; Mon, 28 May 2001 03:09:53 -0300

X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP

X-Mailer: Internet Mail Service (5.5.2650.21)

Return-Path: podqsupyqrdM@pacbell.net

CIS-387 – LAB 6 – MEECH

Message-ID: <MC6-F18JMXehYSoOZzH0013a4a7@mc6-f18.law1.hotmail.com>

X-OriginalArrivalTime: 22 Apr 2003 09:33:13.0655 (UTC) FILETIME=[323CC470:01C308B2]

From :bob.weitershausen@encase.com

To : jamey.tubbs@encase.com

Subject :Latest training materials

Date :Thu, 27 Mar 2003 18:37:11 +0000

MIME-Version: 1.0

Received: from 3com.com ([211.162.89.61]) by mc7-f37.law1.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600); Mon, 21 Apr 2003 19:50:19 -0700

X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP

Message-ID: <HMKPOGPOBBJEOKEPBELFEILLMIAB.wpeterson@att.com>

In-Reply-To: <25ce01c2ee96$210f2c95$39f4ccfd@zcc2iia>

X-MIMEOLE: Produced By Microsoft MimeOLE V6.00.2800.1106

X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)

Return-Path: wpeterson@att.com

X-OriginalArrivalTime: 22 Apr 2003 02:50:19.0429 (UTC) FILETIME=[E9469D50:01C30879]

From: bill.sutter@encase.com

To: bill.thompson@encase.com

Subject: Simon_Key Unemployed? Need a Job

Date: Mon, May 28 2001 00:54:37 +1100

MIME-Version: 1.0

Received: from aexeij ([65.69.110.33]) by mc9-f5.bay6.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600); Mon, 21 Apr 2003 09:54:44 -0700

X-Message-Info: JGTYoYF78jEHjJx36Oi8+Q1OJDRSDidP

Message-Id: <dfunysdebwviq@dmelkcs.bigfoot.com>

Return-Path: dtyiht@dmelkcs.bigfoot.com

X-OriginalArrivalTime: 21 Apr 2003 16:54:49.0038 (UTC) FILETIME=[B84D9EE0:01C30826]

---

22 www\.[a-z]+\.com|(\.uk)

This search expression searches for URLs starting 'www.', followed by

at least one other character or a hyphen, followed by '.co' and then

followed by an 'm' or '.uk'

http://www.encase.com/

http://www.torent.co.uk/cgi-bin/honat.cgi?si=tr&st=1&ty=3+Bedroom&to=LIVERPOOL

http://www.belvoirgroup.com/sthelens.htm

http://www.haltabuse.org/help/headers/

http://www.planetdv.net/Frameset.asp?show=content&rID=818&RC=10&RV=85

http://www.aol.co.uk/try/aod/aim

Summary/Reflection

Your report should include the activity log (the steps you take or the commands you run) with some screenshots and/or outputs, highlight all the matches in the logical evidence file (GREP-examples.txt) and embedded it in the report, and a brief reflection on what you learned (one or two paragraphs).

I noticed that for the 9th search expression (sue lake\x0D\x0A) there was no hit. This is likely because Linux does not use carriage return line feed sequence characters (which is a Windows standard). So I am guessing the GREP EXAMPLES file was written and saved on a Linux machine. Also, for **regular expression 15** *[456]###-?####-?####-?####[^#]* the description is incorrect → it is actually *any 16 digit number that starts with 4, 5, or 6 and with optional dashes every 4 digits* ***AND that ends in any character that is not a number.*** The bolded part is what was missing. Special note: the regular expression was not in the list given in canvas and so I manually input it but the Autopsy program crashes when I did the search, so I just had to manually (by looking at it myself) say which strings were hits under that regular expression. The Autopsy program also crashes from you search for **regular expression 21** *From ?:.{20,200}To ?:* which is why I assume that is why this expression was also not in the keyword list provided in Canvas. So, I had to also highlight the regular expressions found manually by just looking at the text myself. Other than that, I had no issues using the ACME Autopsy tool to search for regular expressions and help with my understanding of the syntax for searching text using regular expressions logic.