

Foundations in Perl Programming – for experienced programmers

Week Three

This weeks topics

- **Review assignments, walk-through your code**
- **Review of Regular Expressions**
- **More on Regular Expressions**
- **Present to class what your project is going to be**
- **Assignments**

Review Assignments / Code Walk-throughs

Approx
50 mins

Review of Regular Expressions

Approx
60-90 mins

Present to Class Your Project Idea

Approx
45 mins

More on Regular Expressions

Greed

Fundamental Truth:

A quantifier (*, +, ?) settles for the minimum number of matches allowed, but will continue matching as many times as it can.

Manual: p65-75

Approx
2 hours

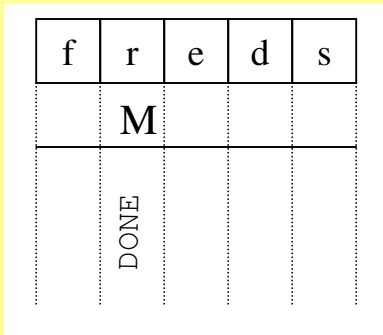
More on Regular Expressions

Greedy

```
$str = fred
```

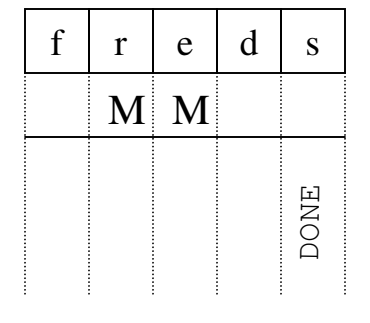
Not greedy

```
$str =~ /\w[re]/
```



match

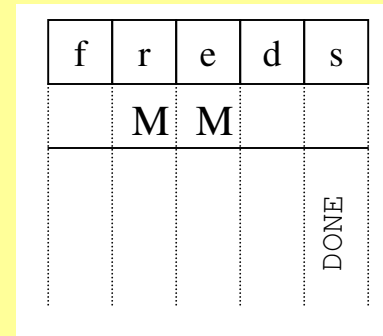
```
$str =~ /\w[re]*/
```



match

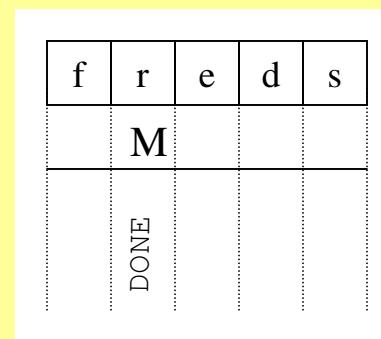
...and Truth

```
$str =~ /\w[re]+/
```



match

```
$str =~ /\w[re]?/
```



match

If they all match
So what?

More on Regular Expressions

Greed

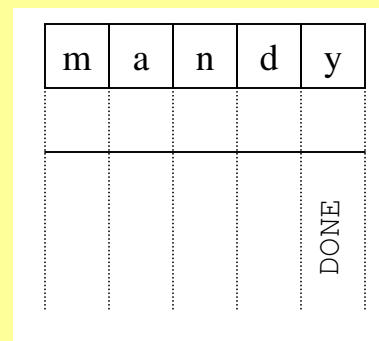
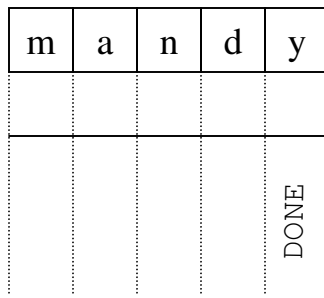
```
$str = mandy
```

...and Truth

Not greedy

```
$str =~ /\w[re]/
```

```
$str =~ /\w[re]+/
```



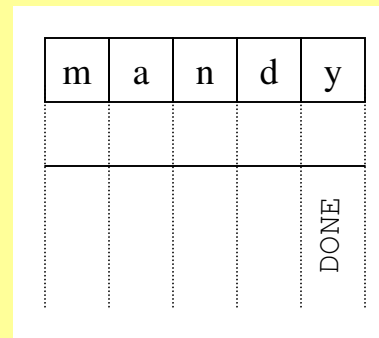
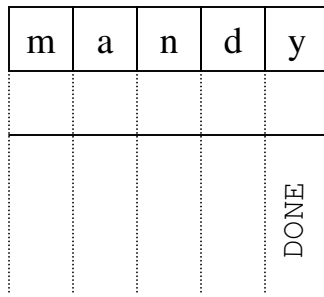
no match

no match

```
$str =~ /\w[re]*/
```

```
$str =~ /\w[re]?/
```

So what again?



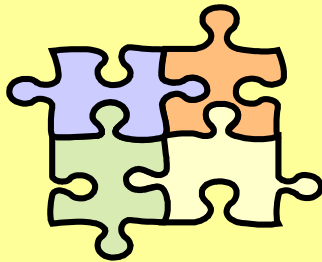
match

match

More on Regular Expressions

Greed

...and Truth



This is a revelation moment. If you understand all this you are well and good. If you do not, this is the point where you should be having an “aha”. If you have not had the “aha” at least remember that it is very important to have the “aha” at some point in time as this is mission critical to understanding and working with regular expressions.



More on Regular Expressions

Greed

Fundamental Truth:

A greedy quantifier will ‘gobble up’ (gobble is a technical term) anything that it can!

We can use the gobbler to our advantage...

More on Regular Expressions

Greed

Using gobbling to our advantage...

A practical example (I think)

I have

Course: Foundations in Perl Programming - for experienced programmers

Course: Foundations in OOP Programming using Java - for experienced programmers

I want

Foundations in Perl Programming - for experienced programmers

Foundations in OOP Programming using Java - for experienced programmers

Using greed to our advantage, here's how

```
my $str="Course: Foundations of Perl
Programming - for experienced
programmers";

print ("$str\n");

if ($str =~ /^Course: (.*)/)
{
    print $1;
}
```

re1.pl

Greed

Parenthesis memory (is beyond frozen)

```
my $str="Course: Foundations of Perl  
Programming - for experienced  
programmers";  
  
print ("$str\n");  
  
if ($str =~ /^Course: (.*)/) {  
    print $1;  
}
```



More on Regular Expressions

Greed

Parenthesis memory

```
my $str="Course: Foundations of Perl  
Programming - for experienced  
programmers";  
  
if ($str =~ /(^Course:) (.*)/) {  
    print $1;  
    print "\n";  
    print $2;  
}
```

re2.pl

Greed

Fundamental Truth:

Greed is stingy. It will only share (give something back) if it absolutely has to. In other words...

Greed only yield to Greed the bare minimum, while it must cough up everything that is requested by a non-greedy request.

pmem.pl

pmem2.pl

As this is a more compelling topic let's look at this works over the next few slides.

More on Regular Expressions

Greed

pmem.pl

\$1

\$2

\$3

\$4

How is this processed?

```
if ($str =~ /(.*)(.*)((.*)(.*)/))
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

This is a rope

\$2

Based on our regex \$1 gobbles the entire string

\$3

\$4

```
if ($str =~ /(.*)(.*)((.*)(.*/))
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

This is a rope

\$2

\$3

\$4

Our second `(.*)` is asking for everything as well. But since the minimum required by `*` is zero, the first `(.*)` is not required to cough anything up.

```
if ($str =~ /(.*)(.*)((.*) (.*)/))
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

This is a rope

\$2

\$3

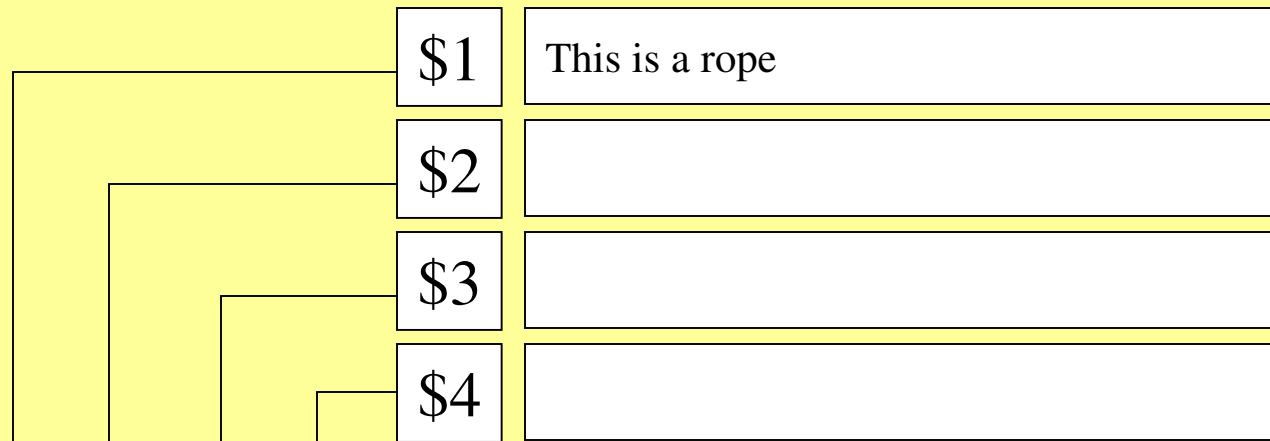
\$4

Our third (.*) is asking for everything as well. But since the minimum required by * is zero, the first (.*) is not required to cough anything up.

```
if ($str =~ /(.*)(.*)((.*)(.*))/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed

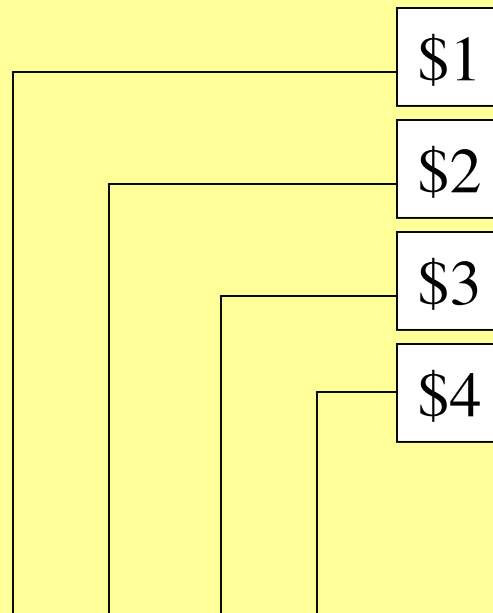


```
if ($str =~ /(.*)(.*)(.*)(.*)/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

Our fourth (.*?) is asking for everything as well. But since the minimum required by * is zero, the third (.*?) is not required to cough anything up.

More on Regular Expressions

Greed



How is this one processed?

```
if ($str =~ /(.) (.+) (.+) (.+) /)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

This is a rope

\$2

Based on our regex \$1 gobbles the entire string

\$3

\$4

```
if ($str =~ /(.)+(.)+(.)+(.)+/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

This is a rop

\$2

e

\$3

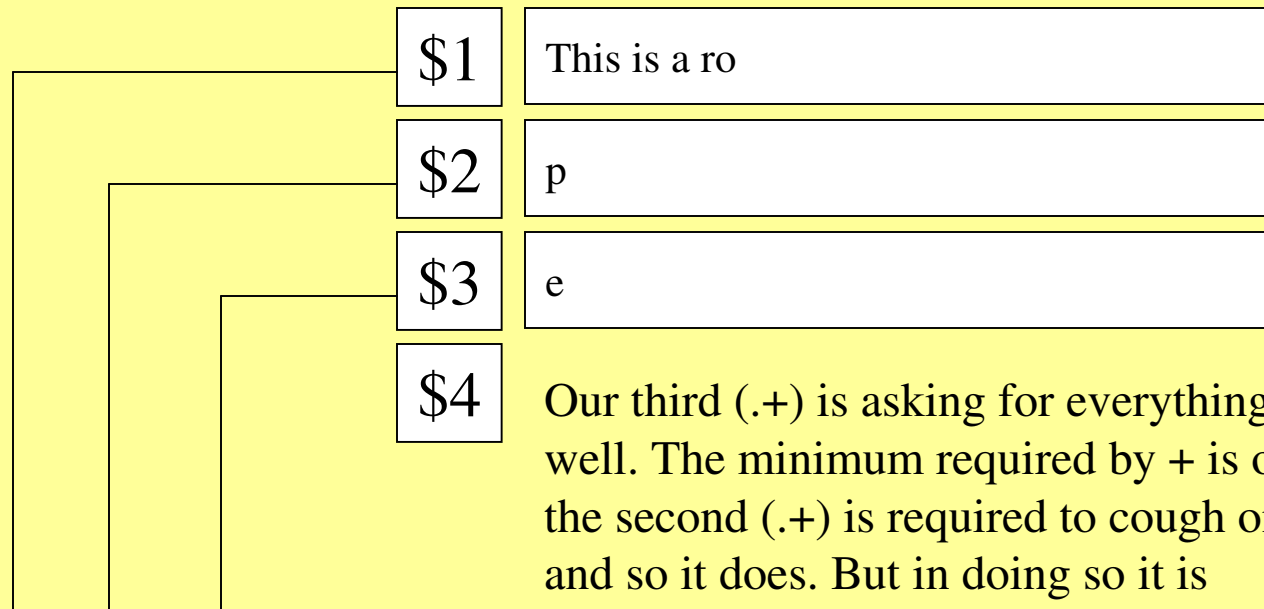
Our second (.+) is asking for everything as well. The minimum required by + is one, the first (.+) is required to cough one and so it does.

\$4

```
if ($str =~ /(.)+(.)+(.)+(.)/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed

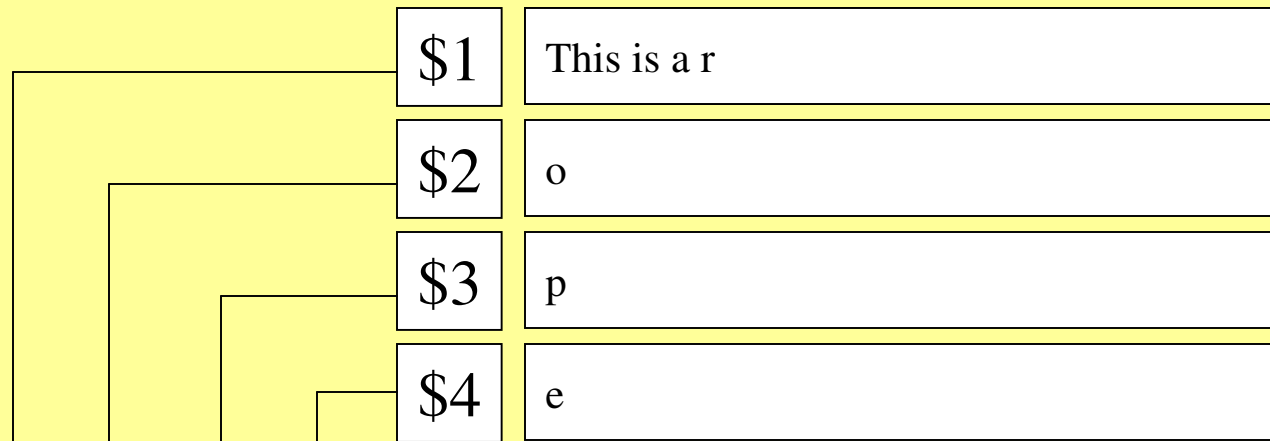


```
if ($str =~ /(.)+(.)+(.)+(.)/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

Our third (.)+ is asking for everything as well. The minimum required by + is one, the second (.)+ is required to cough one and so it does. But in doing so it is without one and one is required, so \$1 must give one more up.

More on Regular Expressions

Greed



```
if ($str =~ /(.) (.+) (.+) (.+)/)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

See the previous slide, because the chain reaction works the same.

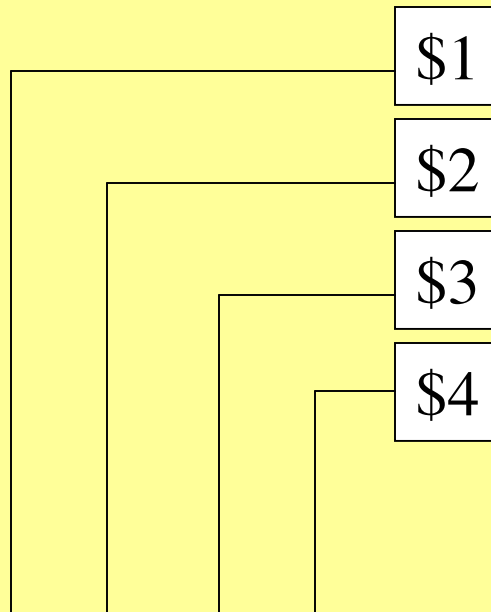
More on Regular Expressions

Greed

```
if ($str =~ /(.?)(.?.)(.?.)(.?.)/)
{
print "$1\n";
print "$2\n";
print "$3\n";
print "$4\n";
}
```

More on Regular Expressions

Greed



How is this processed?

```
if ($str =~ /(.*?) (.*?) (.*?) (.*?) /)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

T

\$2

Based on our regex \$1 takes only the first character because one is allowed.

\$3

\$4

```
if ($str =~ /(.?)(.?) (.?) (.?) /)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

Greed

\$1

T

\$2

h

\$3

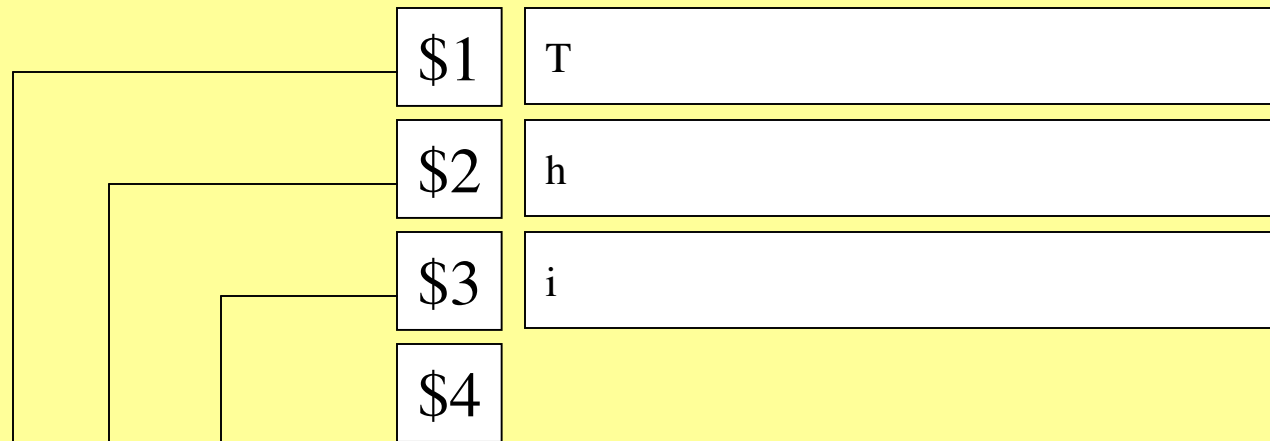
\$4

\$2 needs a character. While one is not required by the greedy operator ?, one is allowed, and you know how greed works. Given this \$1 has nothing to give up so \$2 gets it's character from what is left in \$str.

```
if ($str =~ /(.*?) (.*?) (.*?) (?.?)/)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

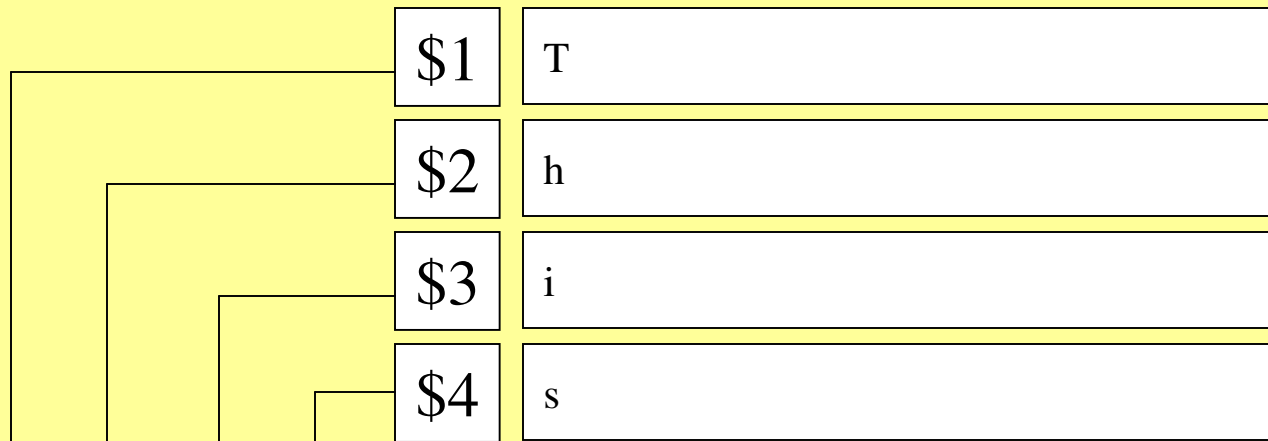
Greed



```
if ($str =~ /(.?)(.?) (.?) (.?) /)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

More on Regular Expressions

Greed



```
if ($str =~ /(.) (.?) (.?) (.?) /)
{
  print "$1\n";
  print "$2\n";
  print "$3\n";
  print "$4\n";
}
```

In class exercise

```
use strict;

my $str = "This is a rope";

if ($str =~ /(.*)(a)(.*)/)
{
print "$1\n";
print "$2\n";
print "$3\n";
}
```

pmem2.pl

Do not run this Perl script! I want you to analyze it and tell me the contents of \$1,2,3.

\$1

\$2

\$3

More on Regular Expressions

Less greed

Greed is not always desired. Greedy processing against the quasi/combo NFA/DFA engine can be labor intensive. Certainly more labor intensive than if you could turn the greedy part off but still use the operators. Below is a table on how to do just that.

{n}	Matches n times
{n,}	Matches at least n times
{n,m}	Matches at least n times but not more than m times

More on Regular Expressions

Compiling Regex

When using regular expressions in a loop, for each iteration of the loop you regex needs to be compiled.

```
$str =~ /\w[re]/o
```

Substitution

In class exercise

Pick one of the next three and code

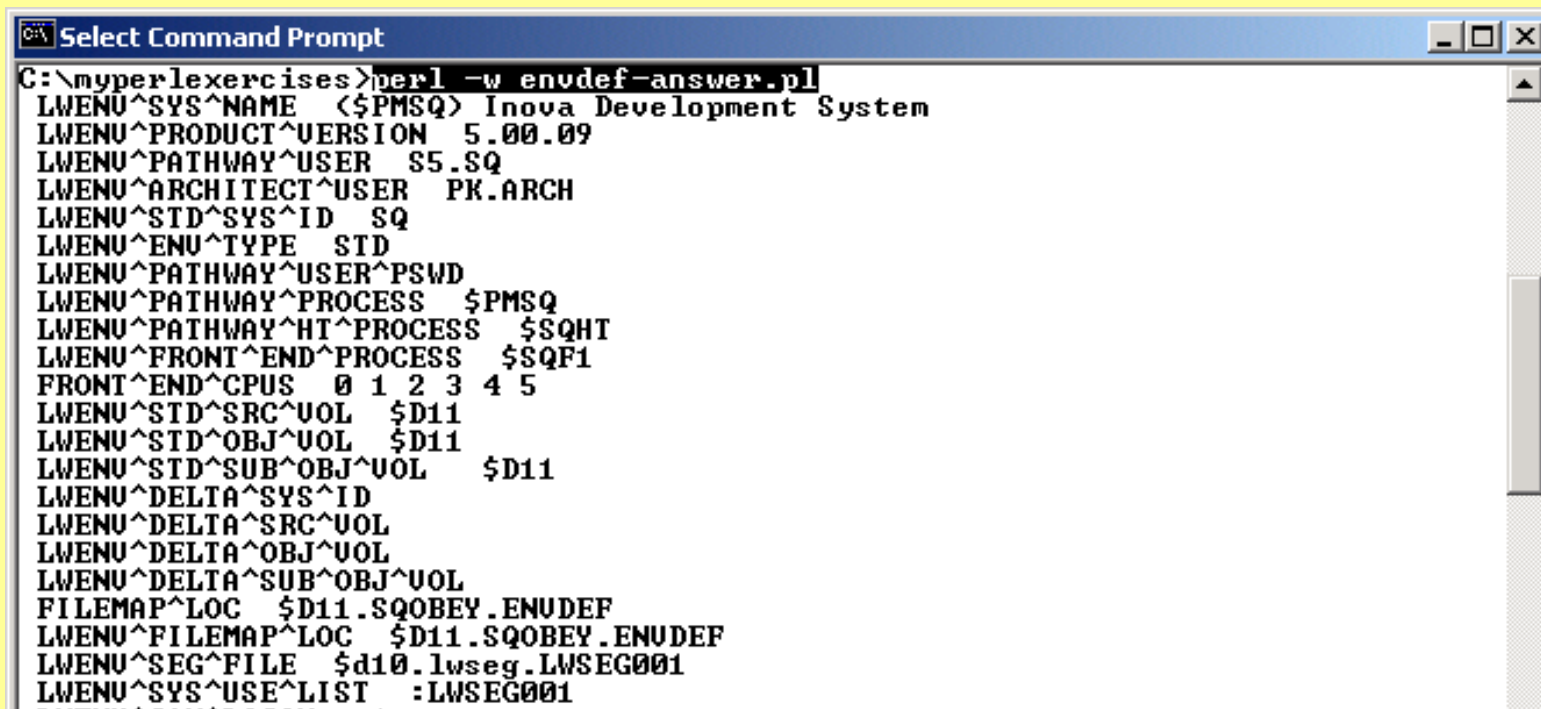
If you do not finish during class time you can add this to your homework.

Approx
2 hours

In class exercise

Return environment variables and their values

- Return all environment variables, and their values (1st value only is required)
- Test program against file “envdef.txt”
- Added challenge: some values extend multiple lines, return these values as well



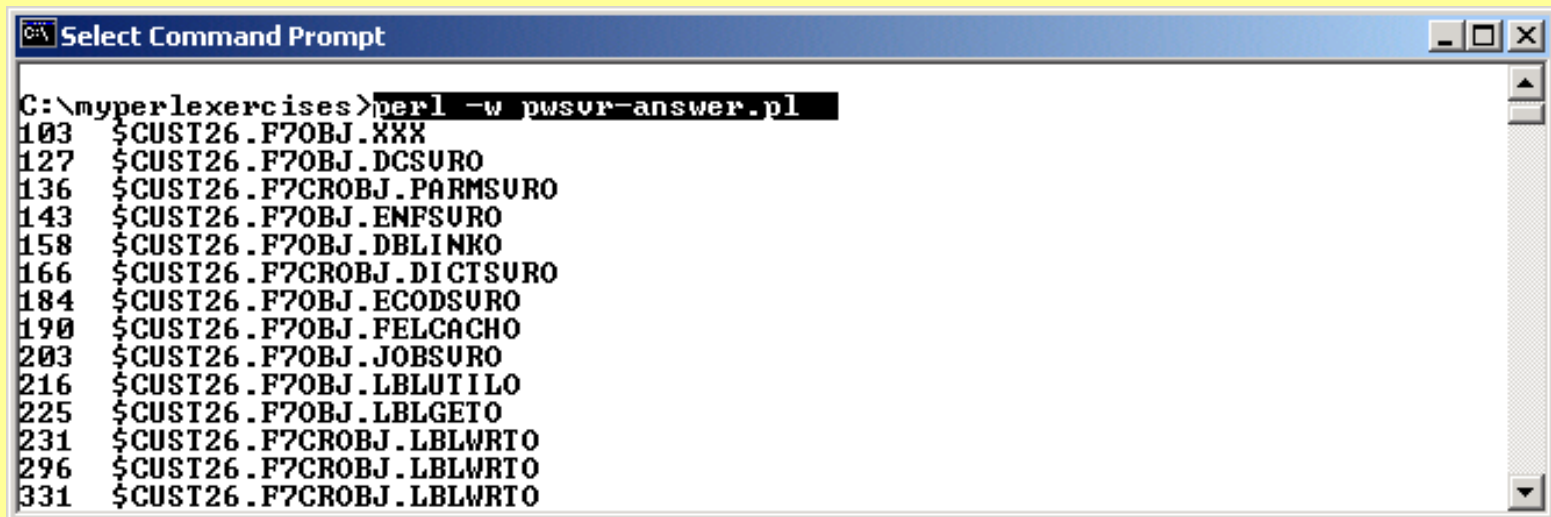
```
Select Command Prompt
C:\myperlexercises>perl -w envdef-answer.pl
LWENU^SYS^NAME <${PMSQ} Inova Development System
LWENU^PRODUCT^VERSION 5.00.09
LWENU^PATHWAY^USER S5.SQ
LWENU^ARCHITECT^USER PK.ARCH
LWENU^STD^SYS^ID SQ
LWENU^ENU^TYPE STD
LWENU^PATHWAY^USER^PSWD
LWENU^PATHWAY^PROCESS ${PMSQ}
LWENU^PATHWAY^HT^PROCESS $$SQHT
LWENU^FRONT^END^PROCESS $$SQF1
FRONT^END^CPUS 0 1 2 3 4 5
LWENU^STD^SRC^UOL $D11
LWENU^STD^OBJ^UOL $D11
LWENU^STD^SUB^OBJ^UOL $D11
LWENU^DELTA^SYS^ID
LWENU^DELTA^SRC^UOL
LWENU^DELTA^OBJ^UOL
LWENU^DELTA^SUB^OBJ^UOL
FILEMAP^LOC $D11.SQOBEY.ENUDEF
LWENU^FILEMAP^LOC $D11.SQOBEY.ENUDEF
LWENU^SEG^FILE $d10.lwseg.LWSEG001
LWENU^SYS^USE^LIST :LWSEG001
```

envdef-ANSWER.pl

In class exercise

Return list of server programs (with line #)

- Return a list of all server programs in pcold
- Include line# from file
- Test program against file “pcold.txt”



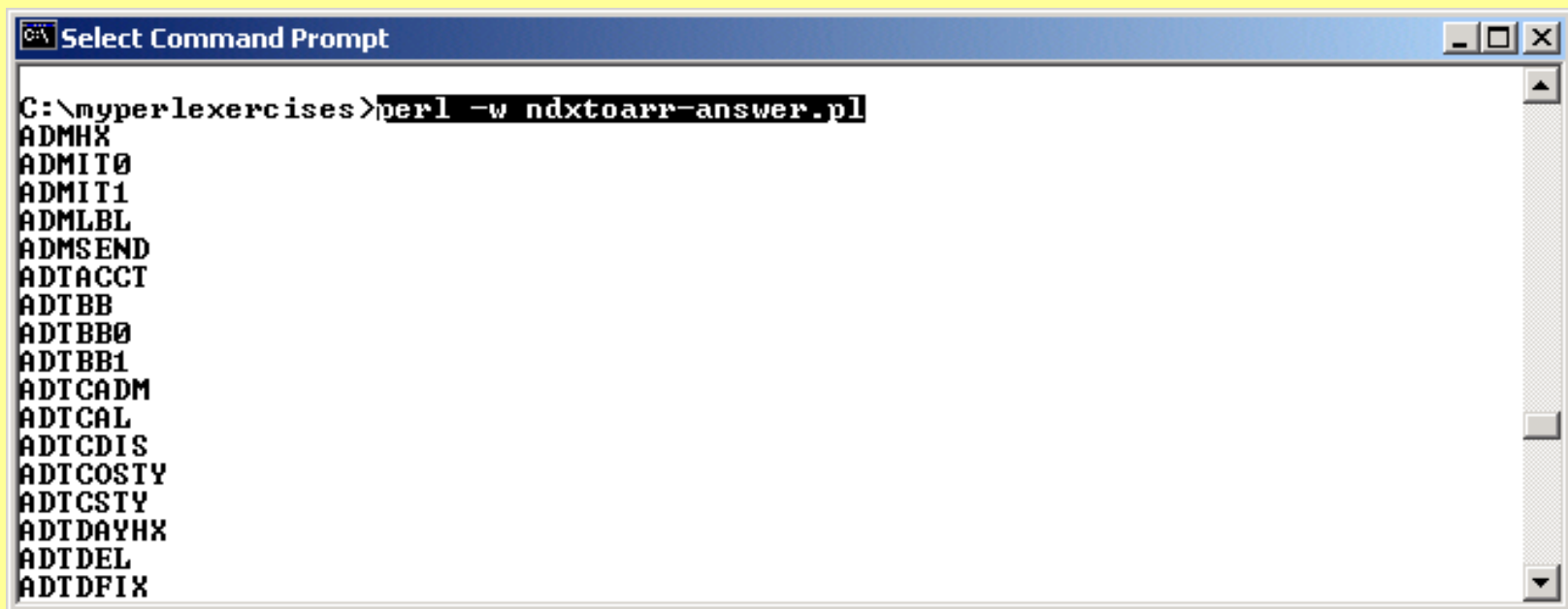
```
Select Command Prompt
C:\myperlexercises>perl -w pwsvr-answer.pl
103 $CUST26.F70BJ.XXX
127 $CUST26.F70BJ.DCSURO
136 $CUST26.F7CROBJ.PARMSURO
143 $CUST26.F70BJ.ENFSURO
158 $CUST26.F70BJ.DBLINKO
166 $CUST26.F7CROBJ.DICTSURO
184 $CUST26.F70BJ.ECODSURO
190 $CUST26.F70BJ.FELCACHO
203 $CUST26.F70BJ.JOBSURO
216 $CUST26.F70BJ.LBLUTILO
225 $CUST26.F70BJ.LBLGETO
231 $CUST26.F7CROBJ.LBLWRTO
296 $CUST26.F7CROBJ.LBLWRTO
331 $CUST26.F7CROBJ.LBLWRTO
```

pwsvr-ANSWER.pl

In class exercise

Return list of filenames of programs from the index file

- Return a list of only filenames for all Cobol, Scobol, Tal and Entrude programs
- Test program against file “index.txt”



```

C:\myperl\exercises>perl -w ndxtoarr-answer.pl
ADMHX
ADMIT0
ADMIT1
ADMLBL
ADMSEND
ADTACCT
ADTBB
ADTBB0
ADTBB1
ADTCADM
ADTCAL
ADTCDIS
ADTCOSTY
ADTCSTY
ADTDAYHX
ADTDEL
ADTDFIX

```

ndxTOarr-ANSWER.pl

Homework Assignment

Pick one of the following and code

Homework Assignment

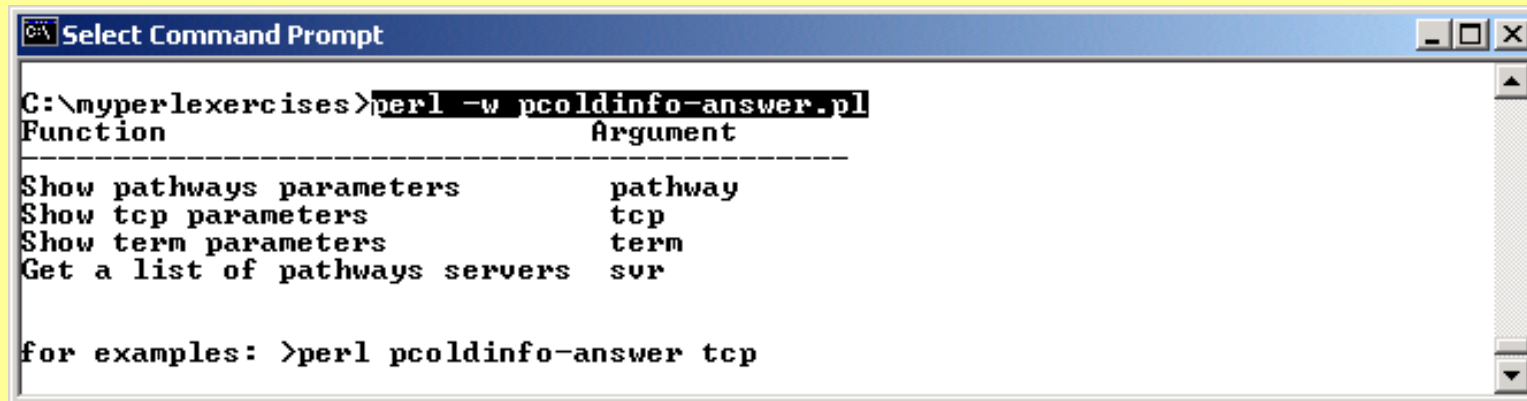
Assignment Choice 1

Homework Assignment

Write a small program that provides the following interface and function points

NOTE: If your occupational title is programmer you cannot Choose this one.

- Display pathway parameters
- Display TCP parameters
- Display term parameters
- Display list of pathway servers
- For file pcold.txt



```
C:\myperlexercises>perl -w pcoldinfo-answer.pl
Function                Argument
-----
Show pathways parameters    pathway
Show tcp parameters        tcp
Show term parameters       term
Get a list of pathways servers  svr

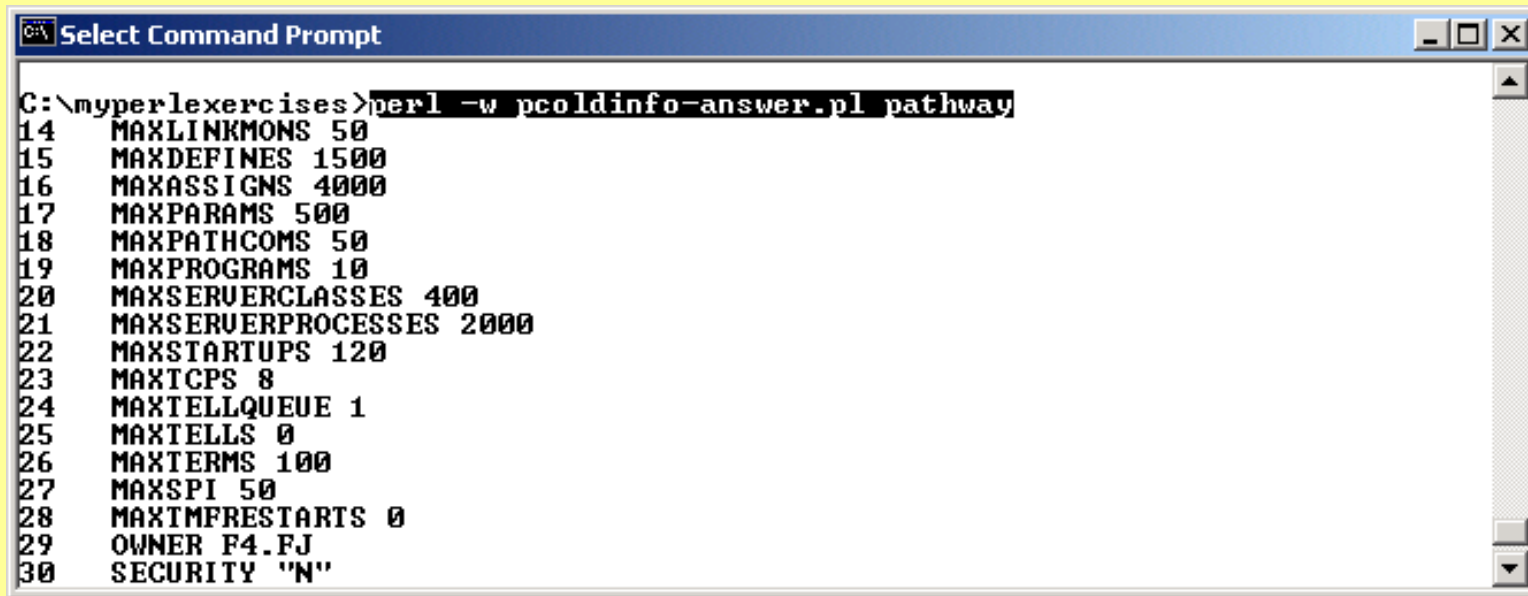
for examples: >perl pcoldinfo-answer tcp
```

pcoldinfo-answer.pl

Homework Assignment

(continuation of previous slide)

- Display pathway parameters

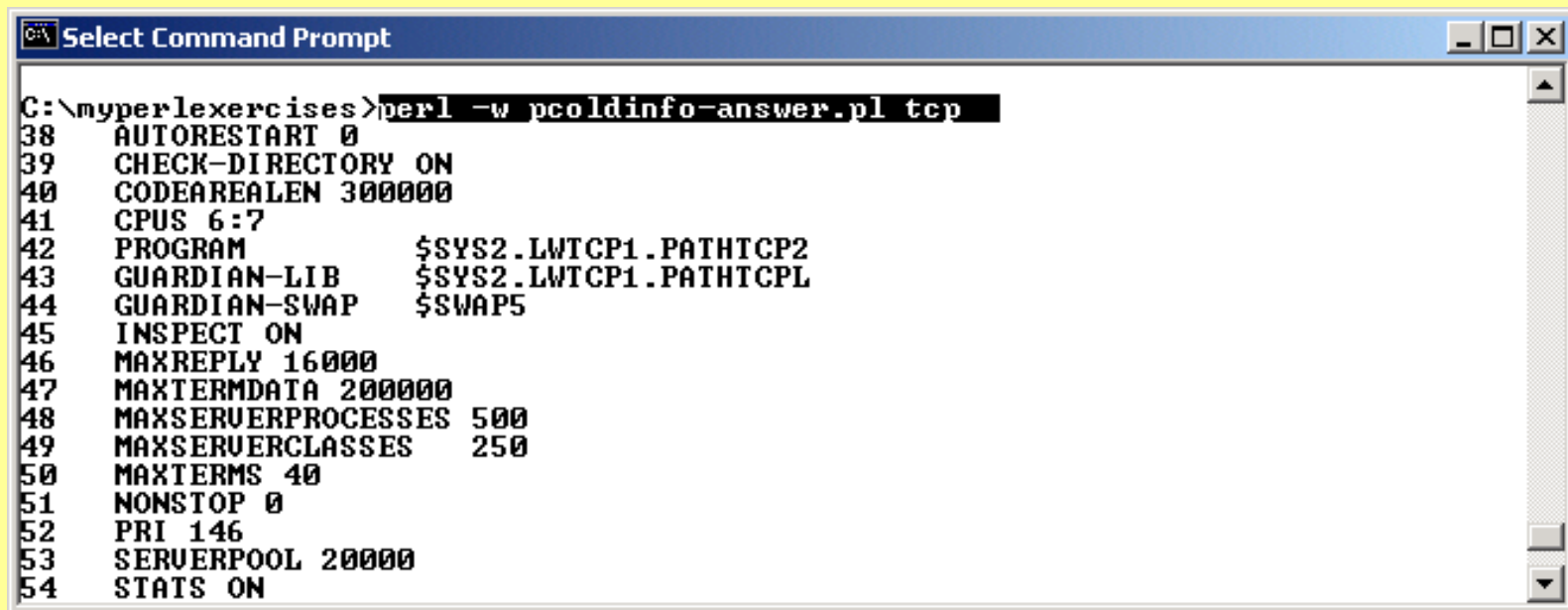


```
Select Command Prompt
C:\myperl\exercises>perl -w pcoldinfo-answer.pl pathway
14  MAXLINKMONS 50
15  MAXDEFINES 1500
16  MAXASSIGNS 4000
17  MAXPARAMS 500
18  MAXPATHCOMS 50
19  MAXPROGRAMS 10
20  MAXSERVERCLASSES 400
21  MAXSERVERPROCESSES 2000
22  MAXSTARTUPS 120
23  MAXTCPS 8
24  MAXTELLQUEUE 1
25  MAXTELLS 0
26  MAXTERMS 100
27  MAXSPI 50
28  MAXIMFRESTARTS 0
29  OWNER F4.FJ
30  SECURITY "N"
```

Homework Assignment

(continuation of previous slide)

- Display TCP parameters

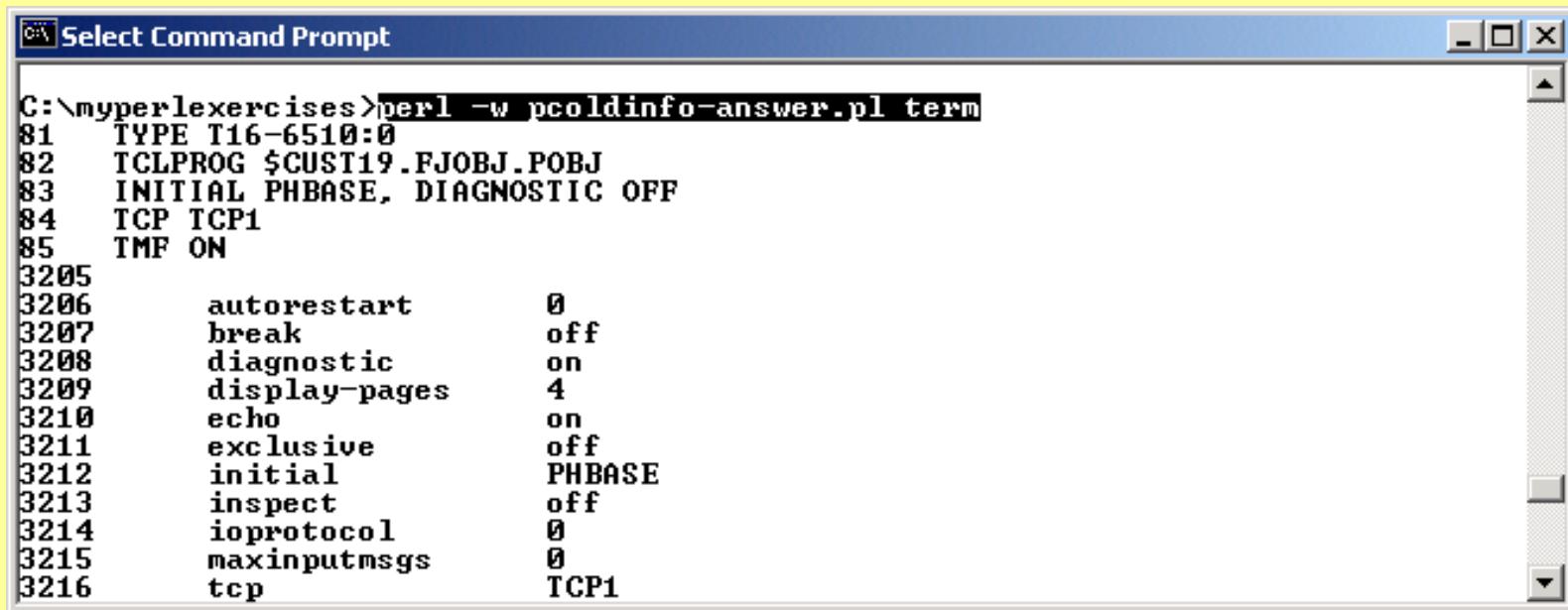


```
C:\myperl\exercises>perl -w pcoldinfo-answer.pl tcp
38  AUTORESTART 0
39  CHECK-DIRECTORY ON
40  CODEAREALEN 300000
41  CPUS 6:7
42  PROGRAM $SYS2.LWTCPL.PATHTCP2
43  GUARDIAN-LIB $SYS2.LWTCPL.PATHTCPL
44  GUARDIAN-SWAP $SWAP5
45  INSPECT ON
46  MAXREPLY 16000
47  MAXTERMDATA 200000
48  MAXSERVERPROCESSES 500
49  MAXSERVERCLASSES 250
50  MAXTERMS 40
51  NONSTOP 0
52  PRI 146
53  SERVERPOOL 20000
54  STATS ON
```

Homework Assignment

(continuation of previous slide)

- Display term parameters

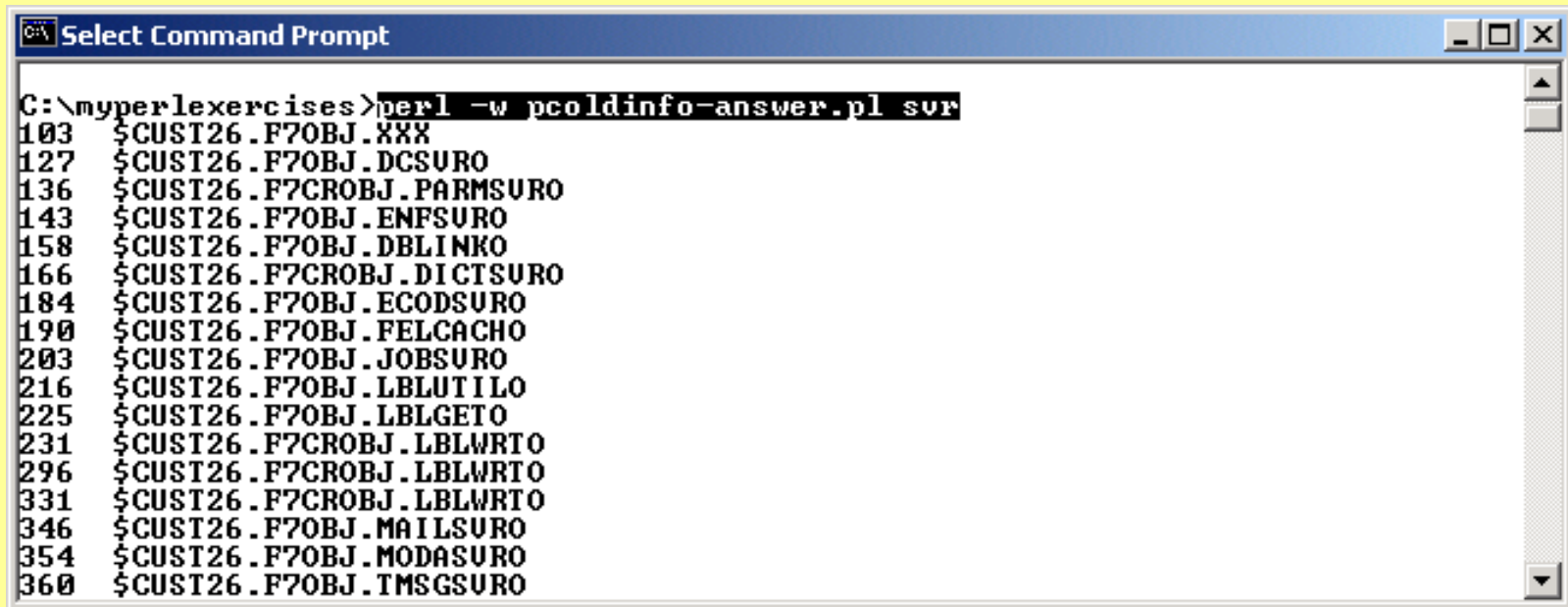


```
C:\myperl\exercises>perl -w pcoldinfo-answer.pl term
81  TYPE T16-6510:0
82  TCLPROG $CUST19.FJOB.POBJ
83  INITIAL PHBASE, DIAGNOSTIC OFF
84  TCP TCP1
85  TMF ON
3205
3206      autorestart      0
3207      break             off
3208      diagnostic        on
3209      display-pages     4
3210      echo              on
3211      exclusive         off
3212      initial           PHBASE
3213      inspect           off
3214      ioprotocol        0
3215      maxinputmsgs      0
3216      tcp               TCP1
```

Homework Assignment

(continuation of previous slide)

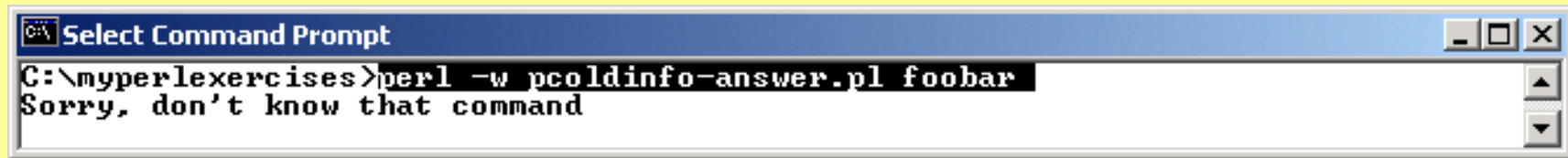
- Display list of pathway servers



```
C:\myperlexercises>perl -w pcoldinfo-answer.pl sur
103 $CUST26.F7OBJ.XXX
127 $CUST26.F7OBJ.DCSURO
136 $CUST26.F7CROBJ.PARMSURO
143 $CUST26.F7OBJ.ENFSURO
158 $CUST26.F7OBJ.DBLINKO
166 $CUST26.F7CROBJ.DICTSURO
184 $CUST26.F7OBJ.ECODSURO
190 $CUST26.F7OBJ.FELCACHO
203 $CUST26.F7OBJ.JOBSURO
216 $CUST26.F7OBJ.LBLUTILO
225 $CUST26.F7OBJ.LBLGETO
231 $CUST26.F7CROBJ.LBLWRTO
296 $CUST26.F7CROBJ.LBLWRTO
331 $CUST26.F7CROBJ.LBLWRTO
346 $CUST26.F7OBJ.MAILSURO
354 $CUST26.F7OBJ.MODASURO
360 $CUST26.F7OBJ.TMSGSURO
```

Homework Assignment

(continuation of previous slide)



```

C:\myperlexercises>perl -w pcoldinfo-answer.pl foobar
Sorry, don't know that command

```

The image shows a screenshot of a Windows Command Prompt window. The title bar reads "Select Command Prompt". The command prompt shows the current directory as "C:\myperlexercises" and the command "perl -w pcoldinfo-answer.pl foobar" has been entered. The output of the command is "Sorry, don't know that command".

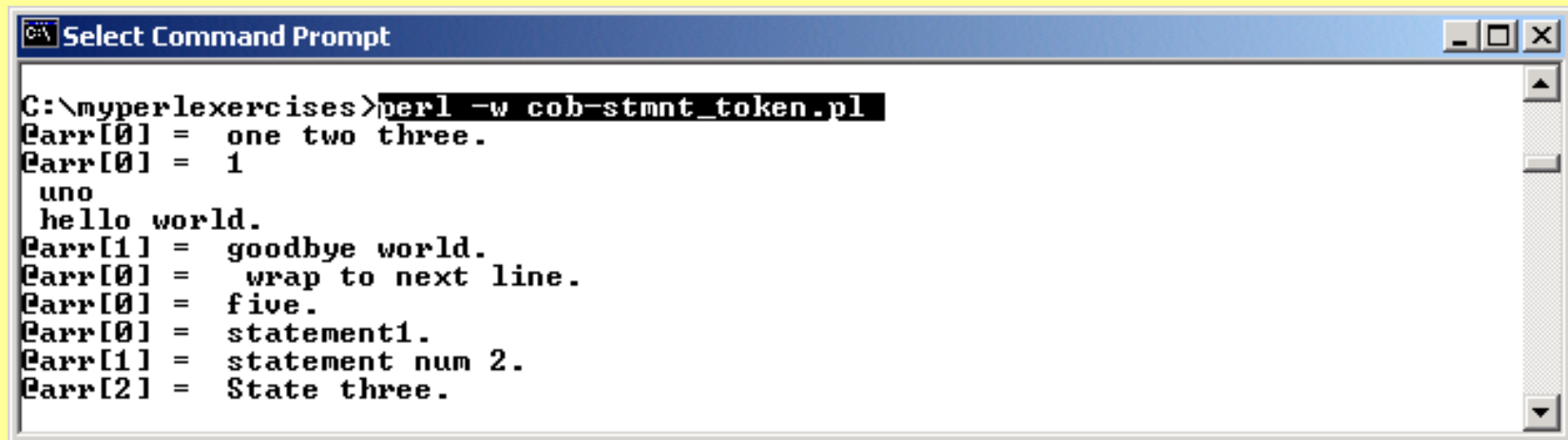
Homework Assignment

Assignment Choice 2a

Homework Assignment

Write a small program that creates a array element for each Cobol statement in a Cobol program

- You must account for the following conditions...
 - Statement spans multiple lines
 - Multiple statements per line
- Test program against file “cob1s.txt”



```
C:\myperlexercises>perl -w cob-stmnt_token.pl
@arr[0] = one two three.
@arr[0] = 1
  uno
  hello world.
@arr[1] = goodbye world.
@arr[0] = wrap to next line.
@arr[0] = five.
@arr[0] = statement1.
@arr[1] = statement num 2.
@arr[2] = State three.
```

cob-stmnt_token.pl

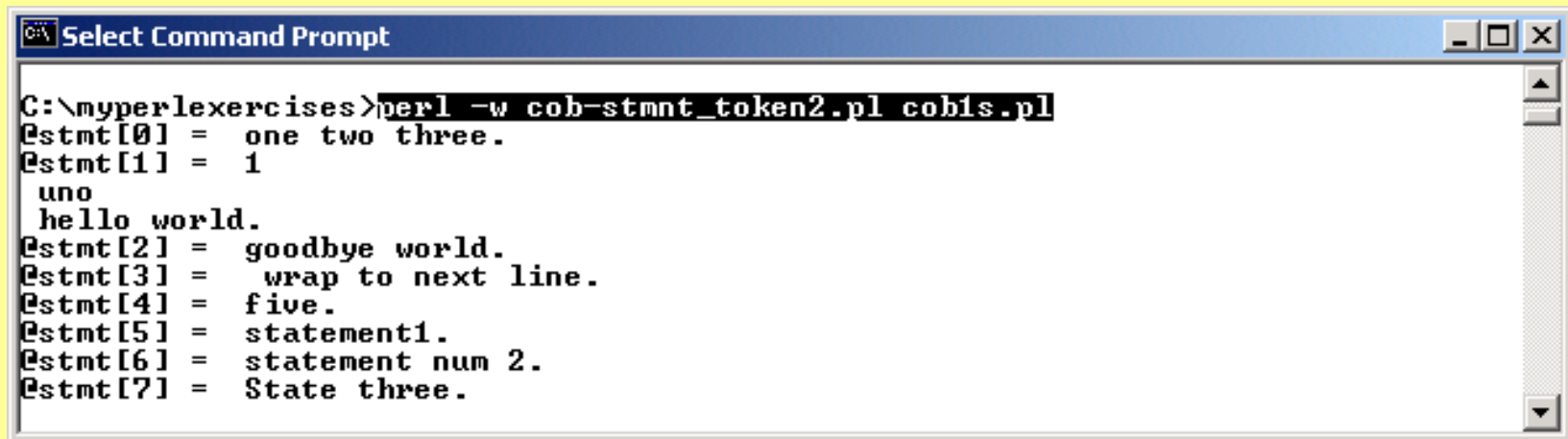
Homework Assignment

Assignment Choice 2b

Homework Assignment

Write a small program that creates a array element for each Cobol statement in a Cobol program (variation)

- You must account for the following conditions...
 - Statement spans multiple lines
 - Multiple statements per line
- Code each in the program as an element is a single array
- Allow file argument at runtime
- Test program against file “cob1s.txt”



```
Select Command Prompt
C:\myperlexercises>perl -w cob-stmnt_token2.pl cob1s.pl
@stmt[0] = one two three.
@stmt[1] = 1
      uno
      hello world.
@stmt[2] = goodbye world.
@stmt[3] = wrap to next line.
@stmt[4] = five.
@stmt[5] = statement1.
@stmt[6] = statement num 2.
@stmt[7] = State three.
```

cob-stmnt_token2.pl

cob-stmnt_token3.pl

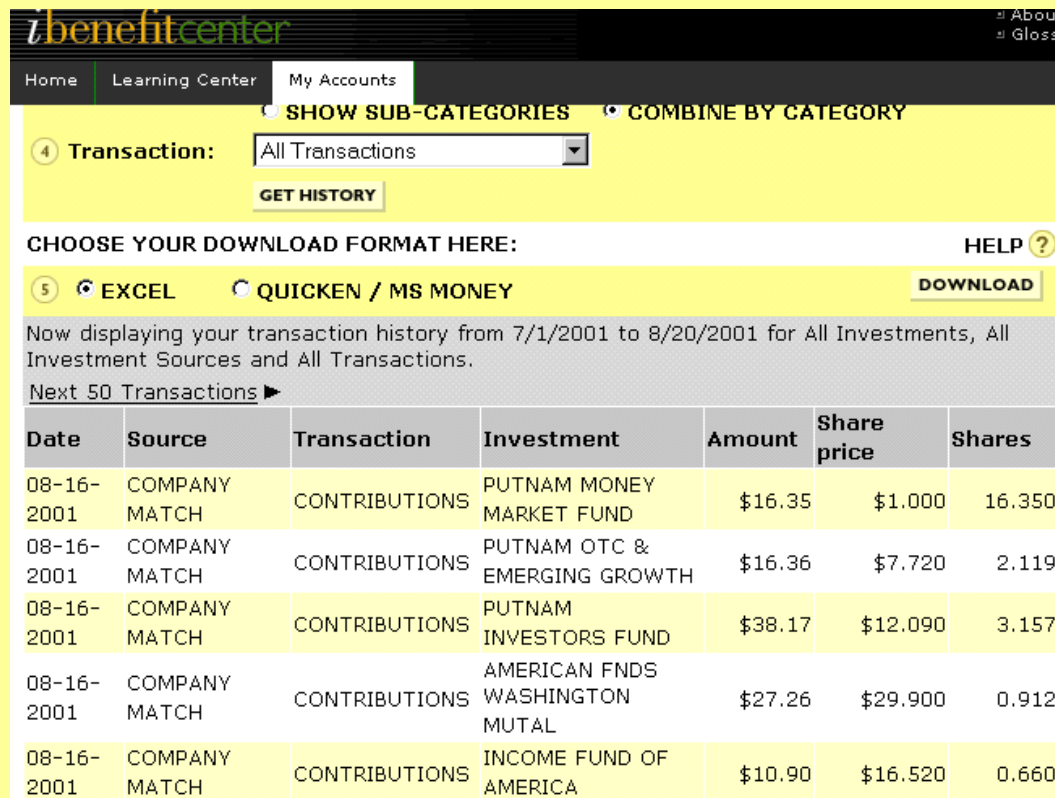
Homework Assignment

Assignment Choice 3a

Homework Assignment

Write a program that extracts the transaction table and converts the data to comma delimited format.

- File to parse is named PersonHeaderLeft[1].html
- Sample results on next page.



Date	Source	Transaction	Investment	Amount	Share price	Shares
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM MONEY MARKET FUND	\$16.35	\$1.000	16.350
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM OTC & EMERGING GROWTH	\$16.36	\$7.720	2.119
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM INVESTORS FUND	\$38.17	\$12.090	3.157
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	AMERICAN FNDS WASHINGTON MUTAL	\$27.26	\$29.900	0.912
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	INCOME FUND OF AMERICA	\$10.90	\$16.520	0.660

commadelim.pl

Homework Assignment

Results

```
Command Prompt
D:\myperl>perl -w commadelim.pl
08-16-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM MONEY MARKET FUND,$16.35,$1.000,16.350,
08-16-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM OTC & EMERGING GROWTH,$16.36,$7.720,2.119,
08-16-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM INVESTORS FUND,$38.17,$12.090,3.157,
08-16-2001,COMPANY MATCH,CONTRIBUTIONS,AMERICAN FNDS WASHINGTON MUTAL,$27.26,$29.900,0.912,
08-16-2001,COMPANY MATCH,CONTRIBUTIONS,INCOME FUND OF AMERICA,$10.90,$16.520,0.660,
08-16-2001,EMPLOYEE 401(K),CONTRIBUTIONS,PUTNAM MONEY MARKET FUND,$43.61,$1.000,43.610,
08-16-2001,EMPLOYEE 401(K),CONTRIBUTIONS,PUTNAM OTC & EMERGING GROWTH,$43.62,$7.720,5.650,
08-16-2001,EMPLOYEE 401(K),CONTRIBUTIONS,PUTNAM INVESTORS FUND,$101.77,$12.090,8.418,
08-16-2001,EMPLOYEE 401(K),CONTRIBUTIONS,AMERICAN FNDS WASHINGTON MUTAL,$72.69,$29.900,2.431,
08-16-2001,EMPLOYEE 401(K),CONTRIBUTIONS,INCOME FUND OF AMERICA,$29.08,$16.520,1.760,
08-08-2001,EMPLOYER PROFIT SHARING,FEE,PUTNAM MONEY MARKET FUND,$-.01,$1.000,-.010,
08-08-2001,COMPANY MATCH,FEE,PUTNAM MONEY MARKET FUND,$-.02,$1.000,-.020,
08-08-2001,EMPLOYEE 401(K),FEE,PUTNAM MONEY MARKET FUND,$-.07,$1.000,-.070,
08-08-2001,EMPLOYER PROFIT SHARING,REALIZED GAIN/LOSS,PUTNAM OTC & EMERGING GROWTH,$0.01,$0.000,0.000,
08-08-2001,EMPLOYER PROFIT SHARING,FEE,PUTNAM OTC & EMERGING GROWTH,$-.01,$8.070,-.001,
08-08-2001,COMPANY MATCH,FEE,PUTNAM OTC & EMERGING GROWTH,$-.01,$8.070,-.001,
08-08-2001,EMPLOYEE 401(K),REALIZED GAIN/LOSS,PUTNAM OTC & EMERGING GROWTH,$-.05,$0.000,0.000,
08-08-2001,EMPLOYEE 401(K),FEE,PUTNAM OTC & EMERGING GROWTH,$-.04,$8.070,-.005,
08-08-2001,EMPLOYER PROFIT SHARING,FEE,PUTNAM INVESTORS FUND,$-.02,$12.260,-.002,
08-08-2001,COMPANY MATCH,FEE,PUTNAM INVESTORS FUND,$-.03,$12.260,-.002,
08-08-2001,EMPLOYER MATCH,FEE,PUTNAM INVESTORS FUND,$-.01,$12.260,-.001,
08-08-2001,EMPLOYEE 401(K),REALIZED GAIN/LOSS,PUTNAM INVESTORS FUND,$-.03,$0.000,0.000,
08-08-2001,EMPLOYEE 401(K),FEE,PUTNAM INVESTORS FUND,$-.12,$12.260,-.010,
08-08-2001,EMPLOYER PROFIT SHARING,REALIZED GAIN/LOSS,AMERICAN FNDS WASHINGTON MUTAL,$0.01,$0.000,0.000,
08-08-2001,EMPLOYER PROFIT SHARING,FEE,AMERICAN FNDS WASHINGTON MUTAL,$-.01,$29.550,0.000,
08-08-2001,COMPANY MATCH,FEE,AMERICAN FNDS WASHINGTON MUTAL,$-.03,$29.550,-.001,
08-08-2001,EMPLOYER MATCH,REALIZED GAIN/LOSS,AMERICAN FNDS WASHINGTON MUTAL,$0.01,$0.000,0.000,
08-08-2001,EMPLOYER MATCH,FEE,AMERICAN FNDS WASHINGTON MUTAL,$-.01,$29.550,0.000,
08-08-2001,EMPLOYEE 401(K),FEE,AMERICAN FNDS WASHINGTON MUTAL,$-.12,$29.550,-.004,
08-08-2001,COMPANY MATCH,FEE,INCOME FUND OF AMERICA,$-.01,$16.380,-.001,
08-08-2001,EMPLOYEE 401(K),REALIZED GAIN/LOSS,INCOME FUND OF AMERICA,$0.01,$0.000,0.000,
08-08-2001,EMPLOYEE 401(K),FEE,INCOME FUND OF AMERICA,$-.04,$16.380,-.002,
07-31-2001,EMPLOYER PROFIT SHARING,DIIVIDEND,PUTNAM MONEY MARKET FUND,$1.27,$1.000,1.270,
07-31-2001,COMPANY MATCH,DIIVIDEND,PUTNAM MONEY MARKET FUND,$3.21,$1.000,3.210,
07-31-2001,EMPLOYER MATCH,DIIVIDEND,PUTNAM MONEY MARKET FUND,$0.43,$1.000,0.430,
07-31-2001,EMPLOYEE 401(K),DIIVIDEND,PUTNAM MONEY MARKET FUND,$11.26,$1.000,11.260,
07-30-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM MONEY MARKET FUND,$20.13,$1.000,20.130,
07-30-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM OTC & EMERGING GROWTH,$20.13,$8.540,2.357,
07-30-2001,COMPANY MATCH,CONTRIBUTIONS,PUTNAM INVESTORS FUND,$46.97,$12.490,3.761,
07-30-2001,COMPANY MATCH,CONTRIBUTIONS,AMERICAN FNDS WASHINGTON MUTAL,$33.55,$29.800,1.126,
07-30-2001,COMPANY MATCH,CONTRIBUTIONS,INCOME FUND OF AMERICA,$13.42,$16.350,0.821,
```

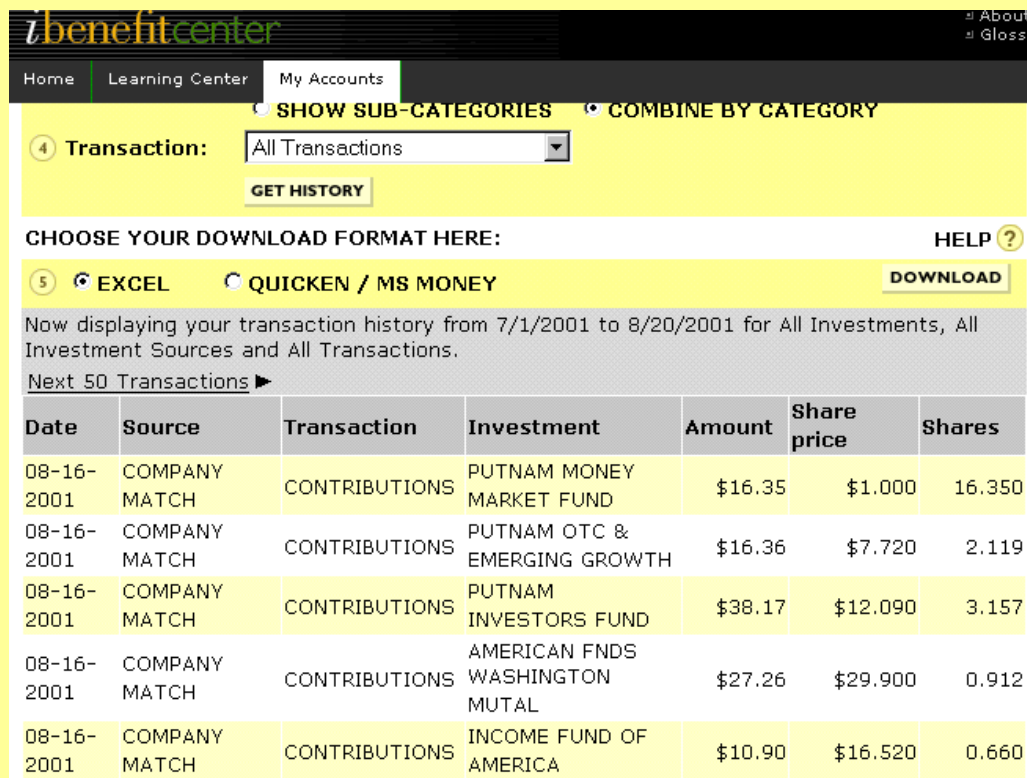
Homework Assignment

Assignment Choice 3b

Homework Assignment

Write a program that extracts the transaction table and converts the data to xml format. (variation)

- File to parse is named PersonHeaderLeft[1].html
- HTML code has “&” as “&”. You need to account for this as restore text back to “&”.
- Sample results on next page.



Date	Source	Transaction	Investment	Amount	Share price	Shares
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM MONEY MARKET FUND	\$16.35	\$1.000	16.350
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM OTC & EMERGING GROWTH	\$16.36	\$7.720	2.119
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	PUTNAM INVESTORS FUND	\$38.17	\$12.090	3.157
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	AMERICAN FNDS WASHINGTON MUTAL	\$27.26	\$29.900	0.912
08-16-2001	COMPANY MATCH	CONTRIBUTIONS	INCOME FUND OF AMERICA	\$10.90	\$16.520	0.660

htmltoxml.pl

Homework Assignment

Results

```
<TRANSACTION>
  <DATE>08-16-2001</DATE>
  <SOURCE>COMPANY MATCH</SOURCE>
  <TRANSACTION_TYPE>CONTRIBUTIONS</TRANSACTION_TYPE>
  <AMOUNT>PUTNAM MONEY MARKET FUND</AMOUNT>
  <SHAREPRICE>$16.35</SHAREPRICE>
  <SHARES>$1.000</SHARES>

  <DATE>08-16-2001</DATE>
  <SOURCE>COMPANY MATCH</SOURCE>
  <TRANSACTION_TYPE>CONTRIBUTIONS</TRANSACTION_TYPE>
  <AMOUNT>PUTNAM OTC & EMERGING GROWTH</AMOUNT>
  <SHAREPRICE>$16.36</SHAREPRICE>
  <SHARES>$7.720</SHARES>

  ...

  <DATE>07-16-2001</DATE>
  <SOURCE>COMPANY MATCH</SOURCE>
  <TRANSACTION_TYPE>CONTRIBUTIONS</TRANSACTION_TYPE>
  <AMOUNT>AMERICAN FNDS WASHINGTON MUTAL</AMOUNT>
  <SHAREPRICE>$27.26</SHAREPRICE>
  <SHARES>$29.600</SHARES>

</TRANSACTION>
```

Homework Assignment

Assignment Choice 3c

Homework Assignment

Combine the XML and Comma Delimited scripts into a single script

- Accept which output as an argument at program runtime.

Homework Assignment

Assignment Choice 4

Homework Assignment

Extract a named section from a file and dump into another file

- Write a program that extracts a ?section from a file and dumps it into another file.
- Test the program against the file ddl.txt
- Make sure to test that program works the same when extracting the last ?section in a file as it does for extracting all other ?section's.

Additional challenge:

- For performance reasons write program so that it exits once it finds the desired ?section.
- Abstract your code so that it could be used for a more general purpose than just extraction of ?section

sect.pl; sect1a.pl; sect1b.pl