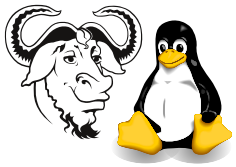

Introduction to Regular Expressions in Perl — Solutions

For each exercise where you write a program, keep the original program from each exercise and modify a copy for the next exercise. You will need to submit all your programs by email to me at `nicku@vtc.edu.hk`. See the subject web site for the format of the subject line of your assignments.

Background: We have seen how character classes can match a set of characters. For example, the character class `/[0-9]/` matches any one digit, and `/[0-9][0-9]/` matches any two digits, one after the other.

This next topic is like gold mining: extracting useful information from among other less useful material.

You can use parentheses in a regular expression, and if there is a match, then the variable `$1` is set to the contents of the first set of parentheses in the regular expression. For example, this code:

```
my $line = "STUDENT REGISTER           2001/02 2nd Term  MODE : PTE";
if ( $line =~ /MODE : ([Pp][Tt][Ee])/ )
{
    print "The mode of study is $1\n";
}
```

prints:

```
The mode of study is PTE
```

1. Download the artificial student data from `http://nicku.org/snm/lab/regular-expressions/artificial-student-data.txt`. This file is in the old format of the student registration system, but contains no real data about any student. We will work toward generating system accounts from this file over the next few classes.

2. Write a Perl program that can read all the lines of this file when it is given as a command line parameter, and display it on standard output. For example, if your program is called `printit`, then the following command will display the content of the big file of student data:

   ```
   $ printit artificial-student-data.txt
   ```

   ```
   #! /usr/bin/perl -w

   while ( <> )
   {
       print;
   }
   ```

or more simply,

```
#! /usr/bin/perl -w
print <>;
```

3. Make a copy of your program, and modify it so that it prints only the lines that contain a number with eight or more digits.

```
#! /usr/bin/perl -w

while ( <> )
{
    print if /[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]/;
}
```

The "backwards" `if` statement is convenient here.

4. Modify the last program so that it prints all lines that contain a Hong Kong ID.

```
#! /usr/bin/perl -w

while ( <> )
{
    print if /[a-zA-Z][0-9][0-9][0-9][0-9][0-9][0-9]\([0-9A-Z]\)/;
}
```

Note that here we need to put a backslash in front of each of the two literal parentheses around the last letter or digit of the Hong Kong ID, otherwise they will have the meaning of selecting that last letter or digit for `$1` (and the pattern will not match a Hong Kong ID).

5. Modify this last program further so that it prints only the Hong Kong ID, and nothing else for each line. Each Hong Kong ID should be printed one to each line. There should be no other output from your program.

```
#! /usr/bin/perl -w

while ( <> )
{
    print "$1\n" if /([a-zA-Z][0-9][0-9][0-9][0-9][0-9][0-9]\([0-9A-Z]\))/;
}
```

Oh, they were all too easy, weren't they? Here I have used just the methods shown so far in lectures, but you could shorten the patterns using `\d` to represent one digit, and using the construct `{6}` to mean a repetition of six times, so the patterns could become:

```
/\d{8}/
/[a-zA-Z]\d{6}\([\dA-Z]\)/
```