ECE 331

Homework 4

See course web site for due date

Place your typed homework answers in `vim`. Print single sided with your name using a **mono space font**. No need to restate questions. Fully investigating questions is required for a higher grade. Please use the kernel coding style for all code. Please use your RPi for developing answers. Although code should be written and run on a RPi, it should run on ANY POSIX compliant OS. As always, all code shall be comment, conform to the Linux Kernel Coding Style, and error conditions shall be checked and appropriately handled.

1.
```
#!/usr/bin/perl
# REs for each sub part
$date=qr/(\d{4}-\d{2}-\d{2})/;
$time=qr/\d{2}:\d{2}:\d{2} [A-Z]{3}/;
$path=qr/[^:]+/;
$time2=qr/(\d{6})h/;
$lat=qr/(\d{2})(\d{2}\.\d{2})/;
$ns=qr/([NS])/;
$lon=qr/(\d{3})(\d{2}\.\d{2})/;
$ew=qr/([EW])/;
$o=qr/O/;
$course=qr/\d{3}/;
$speed=qr/\d{3}/;
$alt=qr/A=(\d{6})/;
# After the alt field, the data is highly variable.  Just pick up any characters.
# Iterate over input
while (<STDIN>) {
        # Skip non-matches
        next unless (m-^$date $time:$path:/$time2$lat$ns/$lon$ew$o$course/$speed/$alt.*$-);
        print;
}
```

2.
```
#!/usr/bin/perl
# REs for each sub part
$date=qr/(\d{4}-\d{2}-\d{2})/;
$time=qr/\d{2}:\d{2}:\d{2} [A-Z]{3}/;
$path=qr/[^:]+/;
$time2=qr/(\d{6})h/;
$lat=qr/(\d{2})(\d{2}\.\d{2})/;
$ns=qr/([NS])/;
$lon=qr/(\d{3})(\d{2}\.\d{2})/;
$ew=qr/([EW])/;
$o=qr/O/;
$course=qr/\d{3}/;
$speed=qr/\d{3}/;
$alt=qr/A=(\d{6})/;
# After the alt field, the data is highly variable.  Just pick up any characters.
# Iterate over input
while (<STDIN>) {
        # Skip non-matches
        next unless (m-^$date $time:$path:/$time2$lat$ns/$lon$ew$o$course/$speed/$alt.*$-);
        # Extract parts
        $d=$1;
        $t=$2;
        $y=$3+$4/60.;
        $yy=$5;
        $x=$6+$7/60.;
        $xx=$8;
        $a=$9;

        # Get date/time
        ($year,$month,$day)=split(/-/,$d);
        $hour=int($t/10000);
        $min=int(($t-$hour*10000)/100);
        $sec=int(($t-$hour*10000-$min*100));

        # Fix lat/lon sign
        $y=-$y if ($yy eq "S");
        $x=-$x if ($xx eq "W");
        # Altitude in meters
        $a=$a*12*25.4/1000;

        # Save
        push @data,"$year $month $day $hour $min $sec $y $x $a";
}

# Generate the gpx file header
print "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
print "<gpx version=\"1.0\" creator=\"GPSBabel\" xmlns=\"http://www.topografix.com/GPX/1/0\">\n";
($year,$month,$day,$hour,$min,$sec,$lat,$lon,$alt)=split(',',$data[0]);
print "  <time>$year-$month-$day";
print "T$hour:$min:$sec.000Z</time>\n";
print "  <trk>\n";
print "    <trkseg>\n";
# For each point
foreach (@data) {
        ($year,$month,$day,$hour,$min,$sec,$lat,$lon,$alt)=split();
        print "      <trkpt lat=\"$lat\" lon=\"$lon\">\n";
        print "        <ele>$alt</ele>\n";
        print "        <time>$year-$month-$day";
        print "T$hour:$min:$sec.000Z</time>\n";
        print "      </trkpt>\n";
}
# Cleanup
print "    </trkseg>\n  </trk>\n</gpx>\n";
```

3.
```
// A. Sheaff 2/16/17
// A program to mimic strstr() without using any libc
// string functions
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdint.h>

// Search for a substring within a string
char *x_strstr(const char *haystack, const char *needle);
// String length
size_t x_strlen(const char *s);

// Entry
int main(int argc, char *argv[])
{
        char buf[4096];

        // Read haystack on stdin
        // Pass needle on command line

        if (argc!=2) {
                printf("Usage: %s needle\n",argv[0]);
                return 1;
        }

        printf("%lu\n",x_strlen(argv[1]));
        while (fgets(buf,4096,stdin)) {
                if (x_strstr(buf,argv[1])) {
                        printf("%sMatch\n",buf);
                }
        }

        return 0;
}

// Mimic strstr()
// Search for the substring needle in the string haystack
char *x_strstr(const char *haystack, const char *needle)
{
        size_t i, j;
        int found=1;

        // Sanity
        if (haystack==NULL) return NULL;
        if (needle==NULL) return NULL;

        if (x_strlen(haystack)<x_strlen(needle)) return NULL;

        // Search for the substring
        for (i=0;i<x_strlen(haystack)-x_strlen(needle)+1;i++) {
                found=1;
```

```
		for (j=0;j<x_strlen(needle);j++) {
			// Short circuit if any character is a mismatch
			if (needle[j] != haystack[i+j]) {
				found=0;
				break;
			}
		}
		if (found) return (char *)&haystack[i];
	}
	return NULL;
}


// String length
size_t x_strlen(const char *s)
{
	size_t i=0; // Length count

	// Sanity
	if (s==NULL) return 0;

	// Count to null character
	while (s[i++]);

	// Return count
	return i-1;
}
```