### Functions and Call Routines

*regex-id = prxparse(perl-regex)*

Compile Perl regular expression *perl-regex* and return *regex-id* to be used by other PRX functions.

*pos = prxmatch(regex-id | perl-regex, source)*

Search in *source* and return position of match or zero if no match is found.

*new-string = prxchange(regex-id | perl-regex, times, old-string)*

Search and replace *times* number of times in *old-string* and return modified string in *new-string*.

*call prxchange(regex-id, times, old-string, new-string, res-length, trunc-value, num-of-changes)*

Same as prior example and place length of result in *res-length*, if result is too long to fit into *new-string*, *trunc-value* is set to 1, and the number of changes is placed in *num-of-changes*.

*text = prxposn(regex-id, n, source)*

After a call to *prxmatch* or *prxchange*, *prxposn* return the text of capture buffer *n*.

*call prxposn(regex-id, n, pos, len)*

After a call to *prxmatch* or *prxchange*, call *prxposn* sets *pos* and *len* to the position and length of capture buffer *n*.

*call prxnext(regex-id, start, stop, source, pos, len)*

Search in *source* between positions *start* and *stop*. Set *pos* and *len* to the position and length of the match. Also set *start* to *pos+len+1* so another search can easily begin where this one left off.

*call prxdebug(on-off)*

Pass 1 to enable debug output to the SAS Log. Pass 0 to disable debug output to the SAS Log.

*call prxfree(regex-id)*

Free memory for a *regex-id* returned by *prxparse*.

### Basic Syntax

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>/…/</td>
<td>Starting and ending regex delimiters</td>
</tr>
<tr>
<td>[ ]</td>
<td>Alternation</td>
</tr>
<tr>
<td>( )</td>
<td>Grouping</td>
</tr>
</tbody>
</table>

### Wildcards/Character Class Shorthands

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Match any one character</td>
</tr>
<tr>
<td>\w</td>
<td>Match a word character (alphanumeric plus &quot; &quot;)</td>
</tr>
<tr>
<td>\W</td>
<td>Match a non-word character</td>
</tr>
<tr>
<td>\s</td>
<td>Match a whitespace character</td>
</tr>
<tr>
<td>\S</td>
<td>Match a non-whitespace character</td>
</tr>
<tr>
<td>\d</td>
<td>Match a digit character</td>
</tr>
<tr>
<td>\D</td>
<td>Match a non-digit character</td>
</tr>
</tbody>
</table>

### Character Classes

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>[...]</td>
<td>Match a character in the brackets</td>
</tr>
<tr>
<td>^...</td>
<td>Match a character not in the brackets</td>
</tr>
<tr>
<td>[a-z]</td>
<td>Match a character in the range a to z</td>
</tr>
</tbody>
</table>

### Position Matching

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Match beginning of line</td>
</tr>
<tr>
<td>$</td>
<td>Match end of line</td>
</tr>
<tr>
<td>\b</td>
<td>Match word boundary</td>
</tr>
<tr>
<td>\B</td>
<td>Match non-word boundary</td>
</tr>
</tbody>
</table>

### Repetition Factors

(greedy, match as many times as possible)

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Match 0 or more times</td>
</tr>
<tr>
<td>+</td>
<td>Match 1 or more times</td>
</tr>
<tr>
<td>?</td>
<td>Match 1 or 0 times</td>
</tr>
<tr>
<td>(n)</td>
<td>Match exactly n times</td>
</tr>
<tr>
<td>(n,r)</td>
<td>Match at least n times</td>
</tr>
<tr>
<td>(n,r,m)</td>
<td>Match at least n but not more than m times</td>
</tr>
</tbody>
</table>

### Advanced Syntax

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-metacharacter</td>
<td>Match character</td>
</tr>
<tr>
<td>{ }</td>
<td>Metacharacters, to match these characters, override (escape) with \</td>
</tr>
<tr>
<td>\</td>
<td>Override (escape) next metacharacter</td>
</tr>
<tr>
<td>\n</td>
<td>Match capture buffer n</td>
</tr>
<tr>
<td>(? : ...)</td>
<td>Non-capturing group</td>
</tr>
</tbody>
</table>

### Lazy Repetition Factors

(match minimum number of times possible)

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Match 0 or more times</td>
</tr>
<tr>
<td>+</td>
<td>Match 1 or more times</td>
</tr>
<tr>
<td>?</td>
<td>Match 0 or 1 time</td>
</tr>
<tr>
<td>(n)</td>
<td>Match exactly n times</td>
</tr>
<tr>
<td>(n,)</td>
<td>Match at least n times</td>
</tr>
<tr>
<td>(n,m)</td>
<td>Match at least n but not more than m times</td>
</tr>
</tbody>
</table>

### Look-Ahead and Look-Behind

<table>
<thead>
<tr>
<th>Character</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?=...)</td>
<td>Zero-width positive look-ahead assertion. E.g. regex1 (?=regex2), a match is found if both regex1 and regex2 match. regex2 is not included in the final match.</td>
</tr>
<tr>
<td>(?!...)</td>
<td>Zero-width negative look-ahead assertion. E.g. regex1 (?!regex2), a match is found if regex1 matches and regex2 does not match. regex2 is not included in the final match.</td>
</tr>
<tr>
<td>(?&lt;=...)</td>
<td>Zero-width positive look-behind assertion. E.g. (?&lt;=regex1) regex2, a match is found if both regex1 and regex2 match. regex1 is not included in the final match.</td>
</tr>
<tr>
<td>(?&lt;!...)</td>
<td>Zero-width negative look-behind assertion.</td>
</tr>
</tbody>
</table>
Basic Example

```sas
data _null_;  pos=prxmatch('/world/', 'Hello world!');  put pos=;  txt=prxchange('s/world/planet/', -1, 'Hello world!');  put txt=; run;
```

Output:
```
pos=7
  txt=Hello planet!
```

Data Validation

```sas
data phone_numbers;
  length first last phone $ 16;
  input first last phone & $16.;
  datalines;
  Thomas Archer     (919)319-1677
  Lucy Barr         (800)899-2164
  Tom Joad          (508) 852-2146
  Laurie Gil        (252)152-7583
;
```

```sas
data invalid;
  set phone_numbers;
  where not prxmatch("\([2-9]\d\d\) \?" || "[2-9]\d\d-\d\d\d\d/", phone);
  run;
```

```sas
data _null_;  length first last phone $ 16;
  retain re;
  if _N_ = 1 then do;
    re = prxparse("\((\[2-9]\d\d\) \?" || "[2-9]\d\d-\d\d\d\d/"));
  end;
  input first last phone & $16.;  if prxmatch(re, phone) then do;
    area_code = prxposn(re, 1, phone);
    if area_code "in ("828" "336" "704" "910" "919" "252") then
      putlog "NOTE: Not in NC: ";
    end;
  run;
```

Output:
```
  NOTE: Not in NC, Lucy Barr (800)899-2164
  NOTE: Not in NC, Tom Joad (508) 852-2146
```

Search and Replace #1

```sas
data _null_;  input;
  _infile_ = prxchange('s/&lt;/</, -1, _infile_);
  put _infile_;  datalines;
  x + y &lt; 15
  x &lt; 10 &lt; y
  y &lt; 11;
run;
```

Output:
```
x + y < 15
x < 10 < y
y < 11
```

Search and Replace #2

```sas
data _null_;  input;
  _infile_ = prxchange('s/\w+$/\w+$/1/,
  \w+$/$2 $1/',' -1, name);
  put _infile_;  datalines;
  Jones, Fred
  Kavich, Kate
  Turley, Ron
  Dulix, Yolanda
;
```

```sas
data reversed_names;
  input name & $32.;
  datalines;
  Jones, Fred
  Kavich, Kate
  Turley, Ron
  Dulix, Yolanda
;
```

```sas
data names;
  set reversed_names;
  name = prxchange('s/\w+$/\w+$/1/,
  \w+$/$2 $1/',' -1, name);
  run;
```

```sas
data _null_;  length first last phone $ 16;
  retain re;
  if _N_ = 1 then do;
    re = prxparse("\((\[2-9]\d\d\) \?" || "[2-9]\d\d-\d\d\d\d/"));
  end;
  input first last phone & $16.;  if prxmatch(re, phone) then do;
    area_code = prxposn(re, 1, phone);
    if area_code "in ("828" "336" "704" "910" "919" "252") then
      putlog "NOTE: Not in NC: ";
    end;
    put log "NOTE: Not in NC, Lucy Barr (800)899-2164"
    "NOTE: Not in NC, Tom Joad (508) 852-2146"
  end;
```

Output:
```
  NOTE: Not in NC, Lucy Barr (800)899-2164
  NOTE: Not in NC, Tom Joad (508) 852-2146
```

Search and Extract

```sas
data _null_;  length first last phone $ 16;
  retain re;
  if _N_ = 1 then do;
    re = prxparse("\((\[2-9]\d\d\) \?" || "[2-9]\d\d-\d\d\d\d/"));
  end;
  input first last phone & $16.;  if prxmatch(re, phone) then do;
    area_code = prxposn(re, 1, phone);
    if area_code "in ("828" "336" "704" "910" "919" "252") then
      putlog "NOTE: Not in NC: ";
    end;
    put log "NOTE: Not in NC, Lucy Barr (800)899-2164"
    "NOTE: Not in NC, Tom Joad (508) 852-2146"
  end;
```