Ex. No. 10a

DATE:

TEXT PROCESSING USING PERL

AIM:

To implement text processing applications using Perl.

MATCHING A PATTERN

– m/ or just / is used to match a pattern in a string.
match a pattern syntax: /pattern/ or m/pattern/
following program uses default variable $_

```
$_="hello how are you";
if (/hello/){
print "default variable =$_\n";
print "found hello\n";
}
# i modifier ignores case difference
if (m/HELLO/i){
print "found HELLo\n";
}
#match with negation
if (! /hallo/){
print "not found hallo\n";
}
print "\n\n";
```

OUTPUT :

```
hello how are you
found hello
found HELLo
not found hallo
```

BIND OPERATOR:

In the above program we used default variable $_.
You can use any variable for any RE using =~ which is called as bind operator.

```
#bind operator
#above example with a variable
$line="hello how are you";
print "line=$line\n";
if ($line =~ /hello/){
```

```
print “found hello\n”;
}
print “\n\n”;
```

OUTPUT:
hello how are you
found hello

MATCHING SINGLE CHARACTER: . (Dot)

dot matches any single character except new line.
at least one character should match

```
$line=”help how are you”;
print “line=$line\n”;
if ($line=~ /hel.p/){
print “hel.p was found\n”;
}
else {
print “hel.p was not found\n”;
}
if ($line=~ /he.p/){
print “he.p was found\n”;
}
else {
print “he.p was not found\n”;
}
print “\n\n”;
```

OUTPUT:
help how are you
hel.p was not found
he.p was found

In the above example you will notice that first if condition fails. This is because there is no character after hel followed by p. The second if condition succeeds because in ‘help’ he is followed by a character and ‘p’.

MATCHING ANY SINGLE CHARACTER ZERO OR ONE TIME – ?

Question mark matches any character zero or one time in a string.

```
$line=”help how are you”;
print “line=$line\n”;
if ($line=~ /hel?p/){
print “hel? was found\n”;
}
```

```
else {
print "hel? was not found\n";
}
```

OUTPUT:

help how are you
hel? was found

```
#another example
$line="help how are yu";
print "line=$line\n";
if ($line=~ /y?u?/){
print "you? was found\n";
}
else {
print "you? was not found\n";
}
print "\n\n";
```

OUTPUT:

help how are yu
you? was found

MATCHING A SINGLE CHARACTER INCLUDING NEWLINE – \S

\s is same as . but including \n

```
print  "\s is same as . but includes \n character".".\n";
$line= "hel\np how are you";
print "line=$line\n";
if ($line=~ /hel\sp/){
print "hel\s was found".".\n";
}
else {
print "hel\s was not found".".\n";
}
print "\n\n";
```

OUTPUT:

\s is same as . but includes \n character
hel
p how are you
hel\s was found

QUANTIFIER: ZERO OR MORE TIMES –

Quantifiers are symbols that will tell perl to match n number of times a particular pattern.

* star is a quantifier to match zero or more times

```
$line = “hello how are you”;
print “line=$line\n”;
if ($line=~ /hel*o/){
print “hel*o found“.”\n” ;
}
else {
print “hel*o not found“.”\n”;
}
$line = “helo how are you”;
print “line=$line\n”;
if ($line=~ /hel*o/){
print “helo found“.”\n”;
}
else {
print “hel*o not found“.”\n”;
}
$line = “heo how are you”;
print “line=$line\n”;
if ($line=~ /hel*o/){
print “hel*o found „.”\n”;
}
else {
print “hel*o not found“.”\n”;
}
print “\n\n”;
```

OUTPUT:

hello how are you
hel*o found.
helo how are you
helo found.
heo how are you
hel*o not found.

MATCH ONE OR MORE TIME – + (PLUS)

plus quantifies one or more time;

```
$line = “hello how are you”;
print “line=$line\n”;
if ($line=~ /hel+o/){
print “hel+o found „.”\n”;
}
else {
print „hel+o not found „.”\n”;
}
$line = “heo how are you”;
print “line=$line\n”;
if ($line=~ /hel+o/){
print “ hel+o found „.”\n”;
}
else {
print “hel+o not found „.”\n”;
}
print “\n\n”;
```

OUTPUT:
hello how are you
hel+o found.
heo how are you
hel+o not found

GROUPING

You can group with ()

```
$line =”hhellohhelloh how are you”;
print “line=$line\n”;
if ($line=~ /h(hello)+h/){
print “(hello)+ found“.”\n”;
}
else {
print “(hello}+ not found „.”\n”;
}
$line =”hellohello how are you”;
print “line=$line\n”;
if ($line=~ /h(hello)*h/){
print „(hello)* found“.”\n”;
}
else {
```

```
print „(hello}* not found „.”\n”;
}
print “\n\n”;
```

OUTPUT:

hhellohhelloh how are you
(hello)+ found.
hellohello how are you
(hello)* found.

OR operator

| pipe is or operator used to connect to regex

```
$line =”vanakkam how are you”;
print “line=$line\n”;
if ($line=~ /(hello)|(vanakkam)/){
print “(hello)|(vanakkam) found „.”\n”;
}
else
{
print “(hello)|(vanakkam) not found „.”\n”;
}
print “\n\n”;
```

OUTPUT:

vanakkam how are you
(hello)|(vanakkam) found.

CHARACTER CLASS [0-9] [A-Z] [A-Z]

[ ] : denotes character class e.g [A-Z] means A to Z

```
$line=”pc180a07″;
print “line=$line\n”;
if ($line=~ /pc[0-9]+[a-z]07/){
print “pc[0-9]+[a-z] occurred“.”\n”;
}
else {
print “pc[0-9]+[a-z] not occurred“.”\n”;
}
$line=”pc180b07″;
print “line=$line\n”;
if ($line=~ /pc[0-9]+[a,b]07/){
print “pc[0-9]+[a,b] occurred“.”\n”;
}
else {
```

```
print “pc[0-9]+[a,b] not occurred“.”\n”;
}
```

OUTPUT:

pc180a07
pc[0-9]+[a-z] occurred.
pc180b07
pc[0-9]+[a,b] occurred.

Inside a character class ^ negates

```
$line=”pcaa0b07″;
print “line=$line\n”;
if ($line=~ /pc[^0-9]+a/){
print “pc[^0-9]+ found“.”\n”;
}
else {
print “pc[^0-9]+ not found „.”\n”;
}
print “\n\n”;
```

OUTPUT:

pcaa0b07
pc[^0-9]+ not found.

SHORTCUT FOR [0-9] \d

\d : \d is a short cut for [0-9]

```
$line=”pc180a07″;
print “line=$line\n”;
if ($line=~ /pc\d+[a-z]07/){
print “pc\d+[a-z]07 occurred“.”\n”;
}
else {
print “pc\d+[a-z]07 not occurred“.”\n”;
}
```

OUTPUT:

pc180a07
pc\d+[a-z]07 occurred.

TR OPERATOR TRANSLATES CHARACTERS IN A LIST

```
$line="original string";
print "line=$line\n";
$line=~ tr/r/xx/;
print „after applying tr/r/xx the result is „."\n"."$line\n";
print "\n";
$line="original string";
print "line=$line\n";
$line=~ tr/rin/RIN/;
print „after applying tr/rin/RIN/ the result is „."\n"."$line\n";
print "\n\n";
```

OUTPUT:

original string
after applying tr/r/xx the result is
oxiginal stxing
original string
after applying tr/rin/RIN/ the result is
original stRINg

SUBSTITUTION

s/// substitutes a substring with given string

```
$line="original string";
print "line=$line\n";
$line=~ s/r/xx/g;
print „after applying s/r/xx the result is „."\n"."$line\n";
print "\n";
```

OUTPUT:

original string
after applying s/r/xx the result is

oxxiginal stxxing

Result:

Thus the text processing using perl is successfully executed.