Unit 1

Q1) what is perl write advantages and disadvantages

-> Perl is a programming language developed by Larry Wall, especially designed for text processing. It stands for Practical Extraction and Report Language. It runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

ADVANTAGES:

1. Perl contains the features of different languages like C, sed, awk, and sh etc. which makes the Perl more useful and productive.

System Administration: Due to having the different scripting languages capabilities Perl make the task of system administration very easy.

1. Perl Provides supports for cross platform and it is compatible with mark-up languages like HTML, XML etc.

• It is very efficient in text-manipulation i.e. Regular Expression. It also provides the socket capability.

• It is free and a Open Source software which is licensed under Artistic and GNU General Public License (GPL).

• It is an embeddable language that’s why it can embed in web servers and database servers.

• It supports more than 25, 000 open source modules on CPAN(Comprehensive Perl Archive Network) which provide many powerful extensions to the standard library. For example, XML processing, GUI(Graphical User Interface) and DI(Database Integration) etc.

Disadvantages of Perl:

• Perl doesn’t supports portability due to CPAN modules.

• Programs runs slowly and program needs to be interpreted each time when any changes are made.

• In Perl, the same result can be achieved in several different ways which make the code untidy as well as unreadable.

• Usability factor is lower when compared to other languages.

Perl advantages:

Its syntax is simple, which makes it easy to understand.

It supports OOP concepts.

Perl programs run easily on the system.

It is more flexible to use.

It supports all platforms and is much more portable.

It has a rich set of Perl modules and free software.

It is much more efficient to work on text and string manipulation as it is a scripting language.

Perl combines features of many other languages which make it easy to understand.

Perl disadvantages

A Perl program containing CPAN modules will not run on another system which doesn't have CPAN modules installed.

It is an interpretative language, so it is slower in comparison with other languages.

It has untidy and unreadable codes.

It starts creating problems when the code is larger than 200 lines.

It is not portable.

Why we learn perl

• Perl is a stable, cross platform programming language.

• Though Perl is not officially an acronym but few people used it as Practical Extraction and Report Language.

• It is used for mission critical projects in the public and private sectors.

• Perl is an Open Source software, licensed under its Artistic License, or the GNU General Public License (GPL).

• Perl was created by Larry Wall.

Features of perl

• Perls database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.

• Perl works with HTML, XML, and other mark-up languages.

• Perl supports Unicode.

• Perl is Y2K compliant.

• Perl supports both procedural and object-oriented programming.

• Perl interfaces with external C/C++ libraries through XS or SWIG.

• Perl is extensible. There are over 20,000 third party modules available from the Comprehensive Perl Archive Network (CPAN).

• The Perl interpreter can be embedded into other systems.

Example:

# This will print "Hello, World"

print "Hello, world\n";

Applications of Perl

1. Perl Is Best Suited For Text Manipulation

2. Bioinformatics With BioPerl

3. Use It For Scripting System Administration Tasks

4. Using Perl For Text-To-Speech Translation

5. Speech recognition is a complex, but increasingly important feature for a variety of applications from virtual assistants to chatbots. Here, too, Perl can help.

6. Perl makes it easy to process and manipulate long sequences such as DNA and proteins. Perl makes it convenient to write a program that controls one or more other programs. As a final example, Perl is used to put biology research labs, and their results, on their own dynamic web sites. Perl does all this and more.

Q2) Explain perl and it’s uses in bioinformatics

-> Perl is a programming language developed by Larry Wall, especially designed for text processing. It stands for Practical Extraction and Report Language. It runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Perl uses in bioinformatics:

In bioinformatics, a DNA sequence might be represented as an object inheriting from a more general implementation that covers the properties of all biological sequences. OOP would code this object by describing its properties such as length, checksum, and certainly the string of letters that comprises the sequence itself. Then one would implement accessor methods to retrieve or set these properties, and also more complex functions such as transcribe() that would take as an argument an organism-specific codon matrix and transform the DNA object to an RNA object.

Q3) Explain Variable Types in detail

A variable is a place to store values. They can be manipulated throughout the program. When variables are created they reserve some memory space.

Perl Variable Declaration

The equal sign (=) is used to assign values to variables. At the left of (=) is the variable name and on the right it is the value of the variable.

Example:

1. $name = "Anastasia";

2. $rank = "9th";

3. $marks = 756.5;

There are three types of variables:

• Scalar defined by $

• Arrays defined by @

• Hashes defined by %

The same variable can be used for all these three types of variables in a program. For example, $name, @name and %name, all three variables will be considered different in a program.

1. Scalar Variable :

A scalar is a single unit of data. That data might be an integer number, floating point, a character, a string, a paragraph, or an entire web page. Simply saying it could be anything, but only a single thing.

Example:

#!/usr/bin/perl

$age = 25; # An integer assignment

$name = "John Paul"; # A string

$salary = 1445.50; # A floating point

print "Age = $age\n";

print "Name = $name\n";

print "Salary = $salary\n";

Output:

Age = 25

Name = John Paul

Salary = 1445.5

2. Array Variable:

An array is a variable that stores an ordered list of scalar values. Array variables are preceded by an "at" (@) sign. To refer to a single element of an array, you will use the dollar sign ($) with the variable name followed by the index of the element in square brackets.

Example:

#!/usr/bin/perl

@ages = (25, 30, 40);

@names = ("John Paul", "Lisa", "Kumar");

print "\$ages[0] = $ages[0]\n";

print "\$ages[1] = $ages[1]\n";

print "\$ages[2] = $ages[2]\n";

print "\$names[0] = $names[0]\n";

print "\$names[1] = $names[1]\n";

print "\$names[2] = $names[2]\n";

Here we used escape sign (\) before the $ sign just to print it. Other Perl will understand it as a variable and will print its value. When executed, this will produce the following result −

Output:

$ages[0] = 25

$ages[1] = 30

$ages[2] = 40

$names[0] = John Paul

$names[1] = Lisa

$names[2] = Kumar

3. Hash Variable:

A hash is a set of key/value pairs. Hash variables are preceded by a percent (%) sign. To refer to a single element of a hash, you will use the hash variable name followed by the "key" associated with the value in curly brackets.

Example:

#!/usr/bin/perl

%data = ('John Paul', 45, 'Lisa', 30, 'Kumar', 40);

print "\$data{'John Paul'} = $data{'John Paul'}\n";

print "\$data{'Lisa'} = $data{'Lisa'}\n";

print "\$data{'Kumar'} = $data{'Kumar'}\n";

Output:

$data{'John Paul'} = 45

$data{'Lisa'} = 30

$data{'Kumar'} = 40

Q4) Difference between single and doubles quotes in perl

Single Quote Strings

In Perl, when a string has single quote, it means everything is treated as a literal mere text. Even if it is the name of a variable.

Example:

# Single Quote Strings

$x = 5;

$my\_name = 'Include help $x';

print $my\_name;

Output:

Include help $x

As you can see from the example above, the variable name $x has been displayed just by its name $x.

Double Quote Strings

Run the above example with double quotes and see the output.

Example:

# Double Quote Strings

$x = 5;

$my\_name = "Include help $x";

print $my\_name;

Output:

Include help 5

The difference between single and double-quotes is that double quotes insert variables and special characters such as newlines \n, whereas a single quote does not insert any variable or special character.

Q5) what is Scalar Operator in perl elaborate on its type.

There are following miscellaneous operators supported by Perl language. Assume variable a holds 10 and variable b holds 20 then −

Sr.No. Operator & Description

1. .

Binary operator dot (.) concatenates two strings.

Example − If $a = "abc", $b = "def" then $a.$b will give "abcdef"

1. x

The repetition operator x returns a string consisting of the left operand repeated the number of times specified by the right operand.

Example − ('-' x 3) will give ---.

1. ..

The range operator .. returns a list of values counting (up by ones) from the left value to the right value

Example − (2..5) will give (2, 3, 4, 5)

1. ++

Auto Increment operator increases integer value by one

Example − $a++ will give 11

1. --

Auto Decrement operator decreases integer value by one

Example − $a-- will give 9

1. ->

The arrow operator is mostly used in dereferencing a method or variable from an object or a class name

Example − $obj->$a is an example to access variable $a from object $obj

Example

Try the following example to understand all the miscellaneous operators available in Perl. Copy and paste the following Perl program in test.pl file and execute this program.

#!/usr/local/bin/perl

$a = "abc";

$b = "def";

print "Value of \$a = $a and value of \$b = $b\n";

$c = $a . $b;

print "Value of \$a . \$b = $c\n";

$c = "-" x 3;

print "Value of \"-\" x 3 = $c\n";

@c = (2..5);

print "Value of (2..5) = @c\n";

$a = 10;

$b = 15;

print "Value of \$a = $a and value of \$b = $b\n";

$a++;

$c = $a ;

print "Value of \$a after \$a++ = $c\n";

$b--;

$c = $b ;

print "Value of \$b after \$b-- = $c\n";

When the above code is executed, it produces the following result −

Output:

Value of $a = abc and value of $b = def

Value of $a . $b = abcdef

Value of "-" x 3 = ---

Value of (2..5) = 2 3 4 5

Value of $a = 10 and value of $b = 15

Value of $a after $a++ = 11

Value of $b after $b-- = 14

Q6) elaborate on conditional statement in Perl

Perl conditional statements helps in the decision making, which require that the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general from of a typical decision making structure found in most of the programming languages −

￼

The number 0, the strings '0' and "" , the empty list () , and undef are all false in a boolean context and all other values are true. Negation of a true value by ! or not returns a special false value.

Perl programming language provides the following types of conditional statements.

Sr.No. Statement & Description

1. if statement

An if statement consists of a boolean expression followed by one or more statements. Syntax

The syntax of an if statement in Perl programming language is −

if(boolean\_expression) {

# statement(s) will execute if the given condition is true

}. If the boolean expression evaluates to true then the block of code inside the if statement will be executed. If boolean expression evaluates to false then the first set of code after the end of the if statement (after the closing curly brace) will be executed. Example: $a = 10;

# check the boolean condition using if statement

if( $a < 20 ) {

# if condition is true then print the following

printf "a is less than 20\n";

}

print "value of a is : $a\n";

1. if...else statement

An if statement can be followed by an optional else statement.

1. if...elsif...else statement

An if statement can be followed by an optional elsif statement and then by an optional else statement.

1. unless statement

An unless statement consists of a boolean expression followed by one or more statements.

1. unless...else statement

An unless statement can be followed by an optional else statement.

1. unless...elsif..else statement

An unless statement can be followed by an optional elsif statement and then by an optional else statement.

1. switch statement

With the latest versions of Perl, you can make use of the switch statement. which allows a simple way of comparing a variable value against various conditions

Q7) What is an array List down operators used in array.

A Perl array variable stores an ordered list of scalar values.

To refer a single element of Perl array, variable name will be preceded with dollar ($) sign followed by index of element in the square bracket.

Perl Simple Array Example

This is a simple example to use Perl array.

@num = (2015, 2016, 2017);

@string = ("One", "Two", "Three");

print "$num[0]\n";

print "$num[1]\n";

print "$num[2]\n";

print "$string[0]\n";

print "$string[1]\n";

print "$string[2]\n";

Output:

2015

2016

2017

One

Two

Three

Array Operators:

1. Push on Array

The push array function appends a new element at the end of the array.

@array = ("pink", "red", "blue");

push @array, "orange";

print "@array\n";

Output:

pink red blue orange

In the above program, "orange" element is added at the end of the array.

1. Pop on Array

The pop array function removes the last element from the array.

@array = ("pink", "red", "blue");

pop @array;

print "@array\n";

Output:

pink red

In the above program, "blue" element is removed from the end of the array.

1. Shift on Array

The shift array function removes the left most element of array and thus shorten the array by 1.

@array = ("pink", "red", "blue");

shift @array;

print "@array\n";

Output:

red blue

In the above program, "pink" is removed from the array.

1. Unshift on Array

The unshift array function adds a new element at the start of the array.

@array = ("pink", "red", "blue");

unshift @array, "orange";

print "@array\n";

Output:

orange pink red blue

In the above program, "orange" is added at the start of the array.

https://www.javatpoint.com/perl-array

Q8) what is an array and explain its advantages and disadvantages and its uses.

An array is a variable that stores an ordered list of scalar values. Array variables are preceded by an "at" (@) sign. To refer to a single element of an array, you will use the dollar sign ($) with the variable name followed by the index of the element in square brackets.

Here is a simple example of using the array variables

Example:

@ages = (25, 30, 40);

@names = ("John Paul", "Lisa", "Kumar");

print "\$ages[0] = $ages[0]\n";

print "\$ages[1] = $ages[1]\n";

print "\$ages[2] = $ages[2]\n";

print "\$names[0] = $names[0]\n";

print "\$names[1] = $names[1]\n";

print "\$names[2] = $names[2]\n";

Here we have used the escape sign (\) before the $ sign just to print it. Other Perl will understand it as a variable and will print its value. When executed, this will produce the following result −

Output :

$ages[0] = 25

$ages[1] = 30

$ages[2] = 40

$names[0] = John Paul

$names[1] = Lisa

$names[2] = Kumar

Advantages of using Array in Perl

Below are the advantages are as follows.

• Multiple data items are accessed by using an array.

• Using array we can save the memory.

• Using an array in Perl debugging of code is easy.

• It is efficient to declare array while defining a single element.

• The array is used to help increase the reusability of code.

Disadvantages of array:

• The size of the array should be known in advance.

• The array is a static data structure with a fixed size so, the size of the array cannot be modified further and hence no modification can be done during runtime.

• Insertion and deletion operations are costly in arrays as elements are stored in contiguous memory.

• If the size of the declared array is more than the required size then, it can lead to memory wastage.

Uses of array

In Perl, array is a special type of variable. The array is used to store the list of values and each object of the list is termed as an element. Elements can either be a number, string, or any type of scalar data including another variable.

Q9) What is an hash explain its literal representation of a hash

The hashes is the most essential and influential part of the perl language. A hash is a group of key-value pairs. The keys are unique strings and values are scalar values.

Hashes are declared using my keyword. The variable name starts with a (%) sign.

Hashes are like arrays but there are two differences between them. First arrays are ordered but hashes are unordered. Second, hash elements are accessed using its value while array elements are accessed using its index value.

No repeating keys are allowed in hashes which makes the key values unique inside a hash. Every key has its single value.

A hash is a set of key/value pairs. Hash variables are preceded by a percent (%) sign. To refer to a single element of a hash, you will use the hash variable name preceded by a "$" sign and followed by the "key" associated with the value in curly brackets..

Here is a simple example of using the hash variables −

Example:

%data = ('John Paul' => 45, 'Lisa' => 30, 'Kumar' => 40);

print "\$data{'John Paul'} = $data{'John Paul'}\n";

print "\$data{'Lisa'} = $data{'Lisa'}\n";

print "\$data{'Kumar'} = $data{'Kumar'}\n";

This will produce the following result −

Output:

$data{'John Paul'} = 45

$data{'Lisa'} = 30

$data{'Kumar'} = 40

Q10) elaborate on sub routine with the help of with and without para metres with suitable example

Perl Functions and Subroutines

Perl functions and subroutines are used to reuse a code in a program. You can use a function at several places in your application with different parameters.

There is only one difference in function and subroutine, subroutine is created with sub keyword and it returns a value. You can divide your code into separate subroutines. Logically each function in each division should perform a specific task.

Syntax of subroutine:

sub subName{

body

}

In the following example, we are defining a subroutine function 'myOffice' and call it

#defining function

sub myOffice{

   print "javaTpoint!\n";

}

# calling Function

myOffice();

Output:

javaTpoint!

Perl subroutine Function with Arguments

You can pass any number of arguments inside a subroutine. Parameters are passed as a list in the special @\_ list array variables. Hence, the first argument to the function will be $\_[0], second will be $\_[1] and so on

You can pass various arguments to a subroutine like you do in any other programming language and they can be acessed inside the function using the special array @\_. Thus the first argument to the function is in $\_[0], the second is in $\_[1], and so on.

Example:

# Function definition

sub Average {

# get total number of arguments passed.

$n = scalar(@\_);

$sum = 0;

foreach $item (@\_) {

$sum += $item;

}

$average = $sum / $n;

print "Average for the given numbers : $average\n";

}

# Function call

Average(10, 20, 30);

Output:

Average for the given numbers : 20

Perl subroutine Function with without Arguments

&marine; # style 2

sub marine {

state $n = 0; # private, persistent variable $n

$n += 1;

print "Hello, sailor number $n!\n";

}

Q11) What do you mean by reference and DeReference in perl explain its features along with suitable example ?

References

In Perl, we use variables to access data stored in a memory location(all data and functions are stored in memory). Variables are assigned with data values which are used in various operations. Perl Reference is a way to access the same data but with a different variable. A reference in Perl is a scalar data type which holds the location of another variable. Another variable can be scalar, hashes, arrays, function name etc. Nested data structure can be created easily as a user can create a list which contains the references to another list that can further contain the references to arrays, scalar or hashes etc.

Example:

# Scalar Value Reference

# defining scalar

$scalar\_val = 1234;

# making reference of scalar variable

$reference\_scalar = \$scalar\_val;

Features

• A reference to an anonymous hash can be created using the curly brackets {} around the key and value pairs.

Example:

# creating reference to anonymous hash

$ref\_to\_anonymous\_hash = {'GFG' => '1', 'Geeks' => '2'};

• A reference to an anonymous array can be created using the square brackets [].

Example:

# creating reference to an anonymous array

$ref\_to\_anonymous\_array = [20, 30, ['G', 'F', 'G']];

Dereferencing

Now, after we have made the reference, we need to use it to access the value. Dereferencing is the way of accessing the value in the memory pointed by the reference. In order to dereference, we use the prefix $, @, % or & depending on the type of the variable(a reference can point to a array, scalar, or hash etc).

Example:

# Dereferencing of a Scalar

# defining a scalar

$scalar = 1234;

# creating an reference to scalar variable

$reference\_scalar = \$scalar;

# Dereferencing

# printing the value stored

# at $reference\_scalar by prefixing

# $ as it is a Scalar reference

print $$reference\_scalar;

Output:

1234

Q12) Elaborate on controlled statement used in perl

Loop control statements change the execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Perl supports the following control statements. Click the following links to check their detail.

Sr.No. Control Statement & Description

1. next statement

It causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

1. last statement

Terminates the loop statement and transfers execution to the statement immediately following the loop.

1. continue statement

A continue BLOCK, it is always executed just before the conditional is about to be evaluated again.

1. redo statement

The redo command restarts the loop block without evaluating the conditional again. The continue block, if any, is not executed.

1. goto statement

Perl supports a goto command with three forms: goto label, goto expr, and goto &name

Unit 2

Q1) what is regexs How regexs works give its application

Regexs: Regular Expression (Regex or Regexp or RE) in Perl is a special text string for describing a search pattern within a given text.

A regular expression is a string of characters that defines the pattern or patterns you are viewing. The syntax of regular expressions in Perl is very similar to what you will find within other regular expression.supporting programs, such as sed, grep, and awk.

The basic method for applying a regular expression is to use the pattern binding operators =~ and !~. The first operator is a test and assignment operator.

There are three regular expression operators within Perl.

• Match Regular Expression - m//

• Substitute Regular Expression - s///

• Transliterate Regular Expression - tr///

The forward slashes in each case act as delimiters for the regular expression (regex) that you are specifying. If you are comfortable with any other delimiter, then you can use in place of forward slash.

￼

Application:

• It can be used to count the number of occurrence of a specified pattern in a string.

• It can be used to search for a string which matches the specified pattern.

• It can also replace the searched pattern with some other specified string.

Q2) Elaborate on types of regular expression operators.

There are three regular expression operators within Perl.

• Match Regular Expression - m//

• Substitute Regular Expression - s///

• Transliterate Regular Expression - tr///

The forward slashes in each case act as delimiters for the regular expression (regex) that you are specifying. If you are comfortable with any other delimiter, then you can use in place of forward slash.

1. The Match Operator

The match operator, m//, is used to match a string or statement to a regular expression. For example, to match the character sequence "foo" against the scalar $bar, you might use a statement like this −

Example:

$bar = "This is foo and again foo";

if ($bar =~ m/foo/) {

print "First time is matching\n";

} else {

print "First time is not matching\n";

}

Output:

First time is matching

2. The Substitution Operator

The substitution operator, s///, is really just an extension of the match operator that allows you to replace the text matched with some new text. The basic form of the operator is −

s/PATTERN/REPLACEMENT/;

The PATTERN is the regular expression for the text that we are looking for. The REPLACEMENT is a specification for the text or regular expression that we want to use to replace the found text with. For example, we can replace all occurrences of dog with cat using the following regular expression −

Example:

$string = "The cat sat on the mat";

$string =~ s/cat/dog/;

print "$string\n";

Output:

The dog sat on the mat

3. The Translation Operator

Translation is similar, but not identical, to the principles of substitution, but unlike substitution, translation (or transliteration) does not use regular expressions for its search on replacement values. The translation operators are −

tr/SEARCHLIST/REPLACEMENTLIST/cds

y/SEARCHLIST/REPLACEMENTLIST/cds

The translation replaces all occurrences of the characters in SEARCHLIST with the corresponding characters in REPLACEMENTLIST. For example, using the "The cat sat on the mat." string we have been using in this chapter −

Example:

$string = 'The cat sat on the mat';

$string =~ tr/a/o/;

print "$string\n";

Output:

The cot sot on the mot.

Q3) what is match operator and explain its modifiers.

The Match Operator

The match operator, m//, is used to match a string or statement to a regular expression. For example, to match the character sequence "foo" against the scalar $bar, you might use a statement like this −

Example:

$bar = "This is foo and again foo";

if ($bar =~ /foo/) {

print "First time is matching\n";

} else {

print "First time is not matching\n";

}

Output:

First time is matching

Match Operator Modifiers

The match operator supports its own set of modifiers. The /g modifier allows for global matching. The /i modifier will make the match case insensitive. Here is the complete list of modifiers

Sr.No. Modifier & Description

1. i

Makes the match case insensitive.

1. m

Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary.

1. o

Evaluates the expression only once.

1. s

Allows use of . to match a newline character.

1. x

Allows you to use white space in the expression for clarity.

1. g

Globally finds all matches.

1. cg

Allows the search to continue even after a global match fails

Q4) what is meta characters? List down Meta characters used in perl and explain any seven. With example.

Metacharacters

A metacharacter is a character that has a special meaning during pattern processing. You use metacharacters in regular expressions to define the search criteria and any text manipulations.

Metacharacters Description Example

^ This character is used to match an expression to its right at the start of a string. ^a is an expression match to the string which starts with 'a' such as "aab", "a9c", "apr", "aaaaab", etc.

$ The $sign is used to match an expression to its left at the end of a string. r$ is an expression match to a string which ends with r such as "aaabr", "ar", "r", "aannn9r", etc.

. This character is used to match any single character in a string except the line terminator, i.e. /n. b.x is an expression that match strings such as "bax", "b9x", "bar".

| It is used to match a particular character or a group of characters on either side. If the character on the left side is matched, then the right side's character is ignored. A|b is an expression which gives various strings, but each string contains either a or b.

\ It is used to escape a special character after this sign in a string.

A It is used to match the character 'A' in the string. This expression matches those strings in which at least one-time A is present. Such strings are "Amcx", "mnAr", "mnopAx4".

Ab It is used to match the substring 'ab' in the string. This expression matches those strings in which 'Ab' is present at least one time. Such strings are "Abcx", "mnAb", "mnopAbx4"

Q5) explain quantifiers used in regex.

Quantifiers

The quantifiers are used in the regular expression for specifying the number of occurrences of a character.

Perl provides several numbers of regular expression quantifiers which are used to specify how many times a given character can be repeated before matching is done. This is mainly used when the number of characters going to be matched is unknown.

Characters Description Example

+ This character specifies an expression to its left for one or more times. s+ is an expression which gives "s", "ss", "sss", and so on.

? This character specifies an expression to its left for 0 (Zero) or 1 (one)times. aS? is an expression which gives either "a" or "as", but not "ass".

1. This character specifies an expression to its left for 0 or more times Br\* is an expression which gives "B", "Br", "Brr", "Brrr", and so on…

{x} It specifies an expression to its left for only x times. Mab{5} is an expression which gives the following string which contains 5 b's:

"Mabbbbb"

{x, } It specifies an expression to its left for x or more times. Xb{3, } is an expression which gives various strings containing at least 3 b's. Such strings are "Xbbb", "Xbbbb", and so on.

{x,y} It specifies an expression to its left, at least x times but less than y times. Pr{3,6}a is an expression which provides two strings.

Both strings are as follows:

"Prrrr" and "Prrrrr

Q6) write a short note on

1. split function:

split() is a string function in Perl which is used to split or you can say to cut a string into smaller sections or pieces. There are different criteria to split a string, like on a single character, a regular expression(pattern), a group of characters or on undefined value etc.. The best thing about this function that user can specify how many sections to split the string into.

Syntax:

split /Pattern/, Expression, Limit

In the above syntax, Pattern is specified a regular expression which provides the criteria to split the string. The Expression is the string which is to be split. The Limit is kind of restriction which stops the splitting at (n-1)th pattern found in the string.

Example:

@fields = split(/:/, "1:2:3:4:5");

print "Field values are: @fields\n";

When above code is executed, it produces the following result −

Field values are: 1 2 3 4 5

2. Join function:

join() function in Perl combines the elements of LIST into a single string using the value of VAR to separate each element. It is effectively the opposite of split.

Note that VAR is only placed between pairs of elements in the LIST; it will not be placed either before the first element or after the last element of the string.

Syntax:

join(VAR, LIST)

Example:

# Joining string with a separator

$string = join( "-", "Geeks", "for", "Geeks" );

print"Joined String is $string\n";

# Joining string without a separator

$string = join( "", "Geeks", "for", "Geeks" );

print"Joined String is $string\n";

Output:

Joined String is Geeks-for-Geeks

Joined String is GeeksforGeeks

3. Substring function:

substr() in Perl returns a substring out of the string passed to the function starting from a given index up to the length specified. This function by default returns the remaining part of the string starting from the given index if the length is not specified. A replacement string can also be passed to the substr() function if you want to replace that part of the string with some other substring.

Syntax:

substr(string, index, length, replacement)

Example:

$temp = substr("okay", 2);

print "Substring valuye is $temp\n";

$temp = substr("okay", 1,2);

print "Substring valuye is $temp\n";

Output:

Substring valuye is ay

Substring valuye is ka

Q7) what is perl formating explain its syntax, field holder with example

Perl uses a writing template called a 'format' to output reports. To use the format feature of Perl, you have to define a format first and then you can use that format to write formatted data.

Define a Format

Following is the syntax to define a Perl format −

format FormatName =

fieldline

value\_one, value\_two, value\_three

fieldline

value\_one, value\_two

.

Here FormatName represents the name of the format. The fieldline is the specific way, the data should be formatted. The values lines represent the values that will be entered into the field line. You end the format with a single period.

Example

format EMPLOYEE =

===================================

@<<<<<<<<<<<<<<<<<<<<<< @<<

$name $age

@#####.##

$salary

===================================

.

select(STDOUT);

$~ = EMPLOYEE;

@n = ("Ali", "Raza", "Jaffer");

@a = (20,30, 40);

@s = (2000.00, 2500.00, 4000.000);

$i = 0;

foreach (@n) {

$name = $\_;

$age = $a[$i];

$salary = $s[$i++];

write;

}

Output:

===================================

Ali 20

2000.00

===================================

===================================

Raza 30

2500.00

===================================

===================================

Jaffer 40

4000.00

===================================

Fieldholders:

 – Value lines denotes/describes the values which will be entered into the fieldlines. – Format is ended by a single period (.) – Fieldholders have the space for the data which will be entered later.

Syntax for Fieldholders:

@<<<<<<< left-justified (with 7 field spaces by counting '@' and '<')

@||||||| center-justified

@###.#### numeric fieldholder

@\* multiline fieldholder

Unit 3

Q1) Difference between SQL and Mongo DB

￼

What are MongoDB and MySQL?

MongoDB MySQL

MongoDB is an open-source database developed by MongoDB, Inc. MongoDB stores data in JSON-like documents that can vary in structure. It is a popular NoSQL database. MySQL is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation.

How Data is Stored?

MongoDB MySQL

In MongoDB, each individual records are stored as ‘documents’. In MySQL, each individual records are stored as ‘rows’ in a table.

HIERARCHICAL UPPER OF A RECORD

MongoDB MySQL

Documents belonging to a particular class or group as

stored in a ‘collection’.

Example: collection of users. A ‘table’ is used to store rows (records) of similar type.

SQL or NoSQL

MongoDB MySQL

MongoDB is what is called a NoSQL database. This means that pre-defined structure for the incoming data can be defined and adhered to but also, if required different documents in a collection can have different structures. It has a dynamic schema. MySQL as the name suggests uses Structured Query Language (SQL) for database access. The schema can not be changed. The inputs following the given schema are only entered.

Example: In a table, if there are columns for name, address and there is need to enter ‘age’ in new column in one of the entries, it will not take it as the column is not defined in schema. This can be achieved in MongoDB, any new field can be inserted irrespective of the schema and is thus known to have dynamic schema.

SALIENT FEATURES

MongoDB MySQL

MongoDB was designed with high availability and scalability in mind, and includes out-of-the-box replication and sharding. MySQL concept does not allow efficient replication and sharding but in MySQL one can access associated data using joins which minimizes duplication

Q2) introduction to Mongo DB

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Q3) History of Mongo DB

MongoDB was founded in 2007 by Dwight Merriman, Eliot Horowitz and Kevin Ryan – the team behind DoubleClick. At the Internet advertising company DoubleClick (now owned by Google), the team developed and used many custom data stores to work around the shortcomings of existing databases.

Q4) Features of Mongo DB

These are some important features of MongoDB:

1. Support ad hoc queries

In MongoDB, you can search by field, range query and it also supports regular expression searches.

1. Indexing

You can index any field in a document.

1. Replication

MongoDB supports Master Slave replication.

A master can perform Reads and Writes and a Slave copies data from the master and can only be used for reads or back up (not writes)

1. Duplication of data

MongoDB can run over multiple servers. The data is duplicated to keep the system up and also keep its running condition in case of hardware failure.

1. Load balancing

It has an automatic load balancing configuration because of data placed in shards.

6. Uses JavaScript instead of Procedures.

7. It is a schema-less database written in C++.

8. Provides high performance.

9. Stores files of any size easily without complicating your stack.

10. Easy to administer in the case of failures.

Q5) advantages and disadvantages of Mongo DB

Advantages of MongoDB over RDBMS

• Schema less − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.

• Structure of a single object is clear.

• No complex joins.

• Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

• Tuning.

• Ease of scale-out − MongoDB is easy to scale.

• Conversion/mapping of application objects to database objects not needed.

• Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Why Use MongoDB?

• Document Oriented Storage − Data is stored in the form of JSON style documents.

• Index on any attribute

• Replication and high availability

• Auto-Sharding

• Rich queries

• Fast in-place updates

• Professional support by MongoDB

Where to Use MongoDB?

• Big Data

• Content Management and Delivery

• Mobile and Social Infrastructure

• User Data Management

• Data Hub

Disadvantages

• MongoDB uses high memory for data storage.

• There is a limit for document size, i.e. 16mb.

• There is no transaction support in MongoDB.

Q6) Commands of Mongo DB

The use Command

MongoDB use DATABASE\_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of use DATABASE statement is as follows −

use DATABASE\_NAME

Example

If you want to use a database with name <mydb>, then use DATABASE statement would be as follows −

>use mydb

switched to db mydb

To check your currently selected database, use the command db

>db

mydb

If you want to check your databases list, use the command show dbs.

>show dbs

local 0.78125GB

test 0.23012GB

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

>db.movie.insert({"name":"tutorials point"})

>show dbs

local 0.78125GB

mydb 0.23012GB

test 0.23012GB

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

The dropDatabase() Method

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax

Basic syntax of dropDatabase() command is as follows −

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, show dbs.

>show dbs

local 0.78125GB

mydb 0.23012GB

test 0.23012GB

>

If you want to delete new database <mydb>, then dropDatabase() command would be as follows −

>use mydb

switched to db mydb

>db.dropDatabase()

>{ "dropped" : "mydb", "ok" : 1 }

>

MongoDB commands

mongo

Enter the MongoDB client

show dbs

List all database. Should have at least on record to display the db in list.

db

Display active database name

db.stats()

Show the database name, number of collection and documents in the database, etc.

use db\_name

To switch / create database

db.dropDatabase( )

Drop database

\*\*CURD\*\* OPERATION

Database in MongoDB

❖ To create a database use company;

Collections

Tables in MongoDB is called as collections

❖ To create a collection

➢ db.createCollection(name,

options)

➢ Eg : db.createCollection(‘Employees’, options)

❖ Drop a collection

➢ db.collection\_name.drop()

Name

Collection name

Options

❖ capped(boolean) -fixed size collecction that automatically Overwrite oldest entries if collection size is reached.

❖ autoindexId(boolean) - Automatically index the \_id field

❖ size(number) - Maximum size of the collection in bytes if capped = true

❖ max - Maximum number of documents allowed in collection

Unit 4

Q1) what is package, explain with syntax how to create and use perl package with example.

A package is a collection or code group in Perl. Meanwhile, a module in Perl is a package defined in a file with the name “package” and the the extension .pm.

We can only define the package once in a program; whereas, more than one different module can have the same name in a single code file.

A package name must always be at the start of a Perl program.

Perl provides a mechanism to protect different sections of code from inadvertently tampering with each other's variables. In fact, apart from certain magical variables, there's really no such thing as a global variable in Perl. Code is always compiled in the current package. The initial current package is package main, but at any time you can switch the current package to another one using the package declaration. The current package determines which symbol table is used for name lookups (for names that aren't otherwise package-qualified). The notion of "current package" is both a compile-time and run-time concept. Most name lookups happen at compile-time, but run-time lookups happen when symbolic references are dereferenced, and also when new bits of code are parsed under eval. In particular, eval operations know which package they were invoked in, and propagate that package inward as the current package of the evaluated code. (You can always switch to a different package within the eval string, of course, since an eval string counts as a block, as does a file loaded in with do, require, or use.)

Syntax:

Package Package\_Name;

Example:

# Package definition

package Vehicle;

sub Car{

print "This is a car subroutine!\n";

}

sub Motorbike{

print "This is a motorbike subroutine!\n";

}

# Using package Vehicle

use Vehicle;

# Function Car of vehicle

Vehicle::Car();

# Function Motorbike of vehicle

Vehicle::Motorbike();

Q2) what do you mean by file handling? Explain file operation used in file handling.

File handling is the most important part in any programming language. A filehandle is an internal Perl structure that associates with a file name.

Perl File handling is important as it is helpful in accessing file such as text files, log files or configuration files.

Perl filehandles are capable of creating, reading, opening and closing a file.

In Perl, a FileHandle associates a name to an external file, that can be used until the end of the program or until the FileHandle is closed. In short, a FileHandle is like a connection that can be used to modify the contents of an external file and a name is given to the connection (the FileHandle) for faster access and ease.

The three basic FileHandles in Perl are STDIN, STDOUT, and STDERR, which represent Standard Input, Standard Output, and Standard Error devices respectively.

File Handling is usually done through the open function.

Syntax: open(FileHandle, Mode, FileName);

Parameters:

• FileHandle- The reference to the file, that can be used within the program or until its closure.

• Mode- Mode in which a file is to be opened.

• FileName- The name of the file to be opened.

The FileHandle is closed using the close function.

Syntax: close(FileHandle); Parameters:

• FileHandle- The FileHandle to be closed.

Q3) what do you mean by directory explain directory operation used in file handling.

Perl is universal and cross-platform programming language mainly used for text manipulation and used in developing many software application like web development, graphical user interface application etc. It is preferred over other programming languages as it is faster, powerful and Perl has a lot of shortcuts which helps in writing a quick script making it take less time for writing. A directory is used in programming languages to store values in the form of lists. A directory is quite similar to a file. Just like a file, the directory also allows performing several operations on it. These operations are used for the modification of an existing directory or creation of a new one.

Different operating systems have different commands to look at the files list within a directory. For example, 'li' command is used for Linux and 'dir' command is used for Windows.

But we can also open a directory in Perl using Perl directory functions. Here, Directory handling is quite similar to file handling.

Different operations that can be performed on a Directory are:

1. Creation of a new Directory

2. Opening an existing Directory

3. Reading content of a Directory

4. Changing a Directory path

5. Closing a Directory

6. Removing a Directory

Following are the standard functions used to play with directories.

opendir DIRHANDLE, EXPR # To open a directory

readdir DIRHANDLE # To read a directory

rewinddir DIRHANDLE # Positioning pointer to the begining

telldir DIRHANDLE # Returns current position of the dir

seekdir DIRHANDLE, POS # Pointing pointer to POS inside dir

closedir DIRHANDLE # Closing a directory.

Q4) elaborate on minus D. And globe function.

Perl functions and subroutines are used to reuse a code in a program. You can use a function at several places in your application with different parameters.

There is only one difference in function and subroutine, subroutine is created with sub keyword and it returns a value. You can divide your code into separate subroutines. Logically each function in each division should perform a specific task.

glob() function in Perl is used to print the files present in a directory passed to it as an argument. This function can print all or the specific files whose extension has been passed to it.

Syntax: glob(Directory\_name/File\_type);

Parameter: path of the directory of which files are to be printed.

Returns: the list of files present in the given directory.

Example 1: Printing names of all the files in the directory

#!/usr/bin/perl

# To store the files

# from the directory in the array

@files = glob('C:/Users/GeeksForGeeks/Folder/\*');

# Printing the created array

print "@files\n"

Above example will print all the files of the requested directory.

Q5) explain object oriented rules in PERL.

Object-oriented programming: As the name suggests, Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Object-oriented programming: As the name suggests, Object-Oriented Programming or OOPs refers to languages that uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

 OOPs Concepts:

• Class

• Object

• Method

• Polymorphism

• Inheritance

• Encapsulation

• Abstraction

1. Class: A class is a user defined blueprint or prototype from which objects are created.  It represents the set of properties or methods that are common to all objects of one type.

• Class name: The name should begin with a initial letter (capitalized by convention).

• Superclass(if any): The name of the class’s parent (superclass), if any, preceded by the keyword ‘use’.

• Constructors(if any):Constructors in Perl subroutines returns an object which is an instance of the class. In Perl, the convention is to name the constructor “new”.

• Body: The class body surrounded by braces, { }.

2. Object: It is a basic unit of Object Oriented Programming and represents the real life entities.  A typical Perl program creates many objects, which as you know, interact by invoking methods. An object consists of :

• State : It is represented by attributes of an object. It also reflects the properties of an object.

• Behavior : It is represented by methods of an object. It also reflects the response of an object with other objects.

• Identity : It gives a unique name to an object and enables one object to interact with other objects.

3. Method: A method is a collection of statements that perform some specific task and return result to the caller. A method can perform some specific task without returning anything. Methods are time savers and help us to reuse the code without retyping the code.

4. Polymorphism: Polymorphism refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Perl with the help of the signature and declaration of these entities. Polymorphism in Perl are mainly of 2 types:

• Overloading in Perl

• Overriding in Perl

5. Inheritance: Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in perl by which one class is allowed to inherit the features(fields and methods) of another class. Important terminology:

• Super Class: The class whose features are inherited is known as superclass(or a base class or a parent class).

• Sub Class: The class that inherits the other class is known as subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

• Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

A class can be created in perl by using packages and can be inherited by using the ‘use’ keyword.

Syntax: use package\_name

6. Encapsulation: Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

• Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.

• As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

• Encapsulation can be achieved by: Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

7. Abstraction: Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.

Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes, etc in the car. This is what abstraction is.

Q7) What is bioperl why bio PERL used in bio informatics give its advantages and disadvantages.

￼

BioPerl is a collection of Perl modules that facilitate the development of Perl scripts for bioinformatics applications. It has played an integral role in the Human Genome Project.

Bioperl provides access to data stores such as GenBank and SwissProt via a flexible series of sequence input/output modules, and to the emerging common sequence data storage format of the Open Bioinformatics Database Access project.

Why use in bioinformatics

Bioperl provides an easy-to-use, stable, and consistent programming interface for bioinformatics application programmers. The Bioperl modules have been successfully and repeatedly used to reduce otherwise complex tasks to only a few lines of code.

Perl makes it easy to process and manipulate long sequences such as DNA and proteins. Perl makes it convenient to write a program that controls one or more other programs. As a final example, Perl is used to put biology research labs, and their results, on their own dynamic web sites.

Advantages:

BioPerl was one of the first biological module repositories that increased its usability. It has very easy to install modules, along with a flexible global repository. BioPerl uses good test modules for a large variety of processes.

Searching for similar sequences. Creating and manipulating sequence alignments. Searching for genes and other structures on genomic DNA. Developing machine readable sequence annotations.

Disadvantages:

There are many ways to use BioPerl, from simple scripting to very complex object programming. This makes the language not clear and sometimes hard to understand. For as many modules that BioPerl has, some do not always work the way they are intended.

Q8) write short note on

Explain function and useful functions

1. BioSeq module:

￼

￼

￼

￼

2. SeqIO:

￼

￼

3. AlignIo:

AlignIO is patterned on the object and its commands have many of the same names as the commands in . Just as in the object can be created with -file and -format

Example:

use Bio::AlignIO; my $io = Bio::AlignIO->new(

1. file => "receptors.aln",
2. format => "clustalw" );

Bio::AlignIO reads many formats but does not write in every format (the same is true for Bio::SeqIO). AlignIO currently supports output in these formats:

• fasta

• mase

• selex

• clustalw

• msf

• phylip

• po

• stockholm

• XMFA

• metafasta

Bio::AlignIO objects can be produced by bioperl-run alignment creation objects (e.g. Clustalw.pm, BLAST’s bl2seq, TCoffee.pm, and Lagan.pm or they can be read in from files of multiple-sequence alignments in various formats using AlignIO.

Q6) write a short note on

1. Inheritance:

Inheritance: Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in perl by which one class is allowed to inherit the features(fields and methods) of another class. Important terminology:

• Super Class: The class whose features are inherited is known as superclass(or a base class or a parent class).

• Sub Class: The class that inherits the other class is known as subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

• Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

A class can be created in perl by using packages and can be inherited by using the ‘use’ keyword.

2. Bioperl:

￼

BioPerl is a collection of Perl modules that facilitate the development of Perl scripts for bioinformatics applications. It has played an integral role in the Human Genome Project.

Bioperl provides access to data stores such as GenBank and SwissProt via a flexible series of sequence input/output modules, and to the emerging common sequence data storage format of the Open Bioinformatics Database Access project.

Perl in bioinformatics

Perl makes it easy to process and manipulate long sequences such as DNA and proteins. Perl makes it convenient to write a program that controls one or more other programs. As a final example, Perl is used to put biology research labs, and their results, on their own dynamic web sites.