Practical 1

Q1. Write a Perl Script to store DNA sequence in Scalar variable entered by user and display an OUTPUT

CODE:

print("Enter your DNA seq: ");

$seq = <stdin>;

print("This is the DNA seq you entered: $seq");

OUTPUT:



Q2. Write a Perl script to ask user to enter RNA sequence using an array without loops

CODE:

@sequence;

print("Enter the first RNA sequence: ");

$sequence[0] = <stdin>;

print("Enter the Second RNA sequence: ");

$sequence[1] = <stdin>;

print("Enter the Third RNA sequence: ");

$sequence[2] = <stdin>;

print("Enter the fourth RNA sequence: ");

$sequence[3] = <stdin>;

print("Enter the fifth RNA sequence: ");

$sequence[4] = <stdin>;

print("The sequences you entered are: \n");

print("Sequence 1: $sequence[0]");

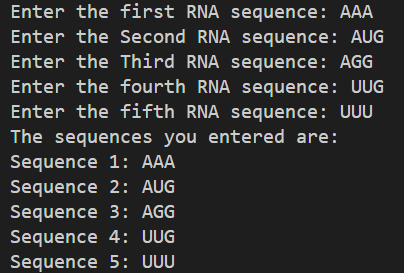
print("Sequence 2: $sequence[1]");

print("Sequence 3: $sequence[2]");

print("Sequence 4: $sequence[3]");

print("Sequence 5: $sequence[4]");

OUTPUT:



Q3. Write a perl script to store codon using hash varibles

CODE:

%codons = (1 => AUG, 2 => AAA, 3 => UUU, 4 => AGG);

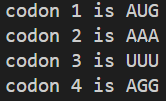
print("codon 1 is $codons{1}\n");

print("codon 2 is $codons{2}\n");

print("codon 3 is $codons{3}\n");

print("codon 4 is $codons{4}\n");

OUTPUT:



Practical 2

Q1. Write a perl script to ask user to enter a number and check whether entering number is even or odd

CODE:

print("Enter a number: ");

$num = <stdin>;

if($num % 2 == 0){

print("Number is Even\n");

}else{

print("Number is Odd\n");

}

OUTPUT:

Q2. Write a perl script to ask user to enter number to display Fibonacci series

CODE:

print("How many digits of fibonacci are to be printed: ");

$limit = <stdin>;

$cur\_num = 1;

$prev\_num = 0;

print("$prev\_num, $cur\_num");

for($i=1; $i<=($limit-2); $i++){

$ans = $cur\_num + $prev\_num;

$prev\_num = $cur\_num;

$cur\_num = $ans;

print("$ans ");

}

OUTPUT:



Q3. Write a Perl script to ask user to entera number and check where entered number is negative or positive

CODE:

print("Enter a number: ");

$number = <stdin>;

if($number % 2 == 0){

print("Number is even\n");

}else{

print("Number is odd\n");

}

OUTPUT:

Q4. Write a Perl script to ask user to enter RNA sequence using an array with for and foreach loops

CODE:

@rna\_seq;

for($i=0;$i<5;$i++){

print("Enter sequence #$i: ");

$rna\_seq[$i] = <stdin>;

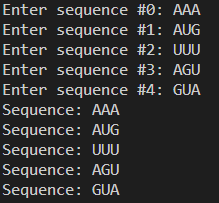
}

foreach $seq(@rna\_seq){

print("Sequence: $seq");

}

OUTPUT:



Q5. Write a Perl script to ask user to enter DNA sequence using a hash and display keys and values separately

CODE:

%dna\_seq;

for($i=0;$i<5;$i++){

$num = $i+1;

print("Enter sequence $num: ");

$dna\_seq{$i} = <stdin>;

}

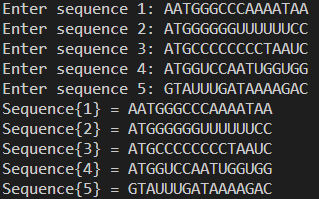
for($i=0; $i<5;$i++){

$num = $i+1;

print("Sequence{$num} = $dna\_seq{$i}");

}

OUTPUT:



Q6. Write a Perl script to ask user to enter number and find factorial of an entered number

CODE:

print("Enter a number: ");

$number = <stdin>;

$ans = 1;

for($i=$number;$i>=1;$i--){

$ans = $i\*$ans;

}

print("Factorial of $number is $ans\n");

OUTPUT:



Q7. Write a Perl script to store DNA sequence and check entered sequence is DNA or not

use Switch;

$seq = <stdin>;

$is\_DNA = false;

for($i=0;$i<length($seq)-1;$i++){

$check = substr($seq, $i, 1);

switch($check){

case "A" {$is\_DNA = true}

case "T" {$is\_DNA = true}

case "G" {$is\_DNA = true}

case "C" {$is\_DNA = true}

default : print "It is not a DNA Sequence";

exit;

}

}

if($is\_DNA == true){

print("It is a DNA sequence");

}

OUTPUT:





Q8. Perl script to display the following pattern

A A A A A

A A A A

A A A

A A

A

CODE:

$num = 5;

for($i=0;$i<6;$i++){

for($j=$num;$j>0;$j--){

print("A ");

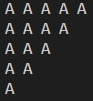
}

print("\n");

$num--;

}

OUTPUT:



Practical 3

Q1. Write a Perl script accept two number and a string and perform the following operations

1. Perl Arithmetic Operators
2. Miscellaneous Operators

CODE:

print("Enter 1st Number: ");

$num1 = <stdin>;

print("Enter 2nd Number: ");

$num2 = <stdin>;

print("Enter 1st String: ");

$string1 = <stdin>;

print("Enter 2nd String: ");

$string2 = <stdin>;

print("\nExecuting arithmetic operators...\n");

print("Addition: ");

print($num1+$num2);

print("\n");

print("Subtraction: ");

print($num1-$num2);

print("\n");

print("Division: ");

print($num/$num2);

print("\n");

print("Multiplication: ");

print($num\*$num2);

print("\n");

print("Modulo ");

print($num%$num2);

print("\n");

print("Exponent ");

print($num\*\*$num2);

print("\n");

print("\nExecuting miscellenous...\n");

print("Concatenate: ");

print("$string1.$string2");

print("\n");

print("Repetition ");

print("$string1"x3);

print("\n");

print("Range ");

print($num1..$num2);

print("\n");

print("Autoincrement(num1): ");

print($num1++);

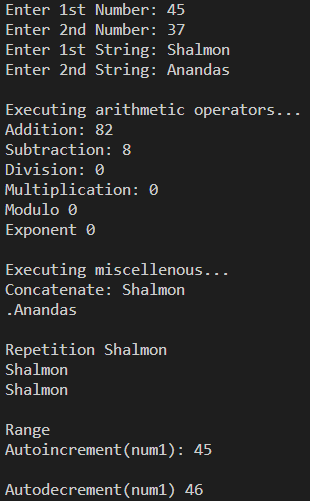
print("\n");

print("Autodecrement(num1) ");

print($num1--);

print("\n");

OUTPUT:



Q2. Write a perl script to accept three number and display smallest number

CODE:

print("Enter #1: ");

$a = <stdin>;

print("Enter #2: ");

$b = <stdin>;

print("Enter #3: ");

$c = <stdin>;

if($a < $b && $a < $c){

print("Biggest number is $a");

}elsif($b < $a && $b < $c){

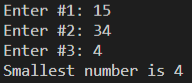
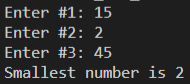
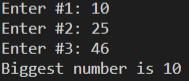
print("Smallest number is $b");

}elsif($c < $a && $c < $b){

print("Smallest number is $c");

}

OUTPUT:



Q3. Write a perl script to enter two string and check wheter its equal or not

CODE:

print("Enter 1st String: ");

$string1 = <stdin>;

print("Enter 2nd String: ");

$string2 = <stdin>;

if($string1 eq $string2){

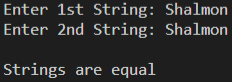
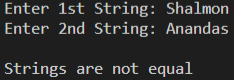
print("\nStrings are equal\n");

}else{

print("\nStrings are not equal\n");

}

OUTPUT:

Q4. Write a Perl script to store elements in an array and perform the following

1. Find length of the array
2. Add one element at end of an array
3. Remove one element at beginning of an array
4. Add one element at beginning of an array
5. Remove on element at end of an array

CODE:

@array = (45,6,3,42,35,22,67,54,23);

print("Array is @array\n");

print("\nLength of the array is $#array\n");

print("\nAdding 25 to the end of the array...\n");

push(@array, 25);

print("Array after adding 25 is @array\n");

print("\nRemoving an element from beginning of the array...\n");

shift(@array);

print("Array after removing element from beginning is @array\n");

print("\nAdding 25 to the beginning of the array...\n");

unshift(@array, 25);

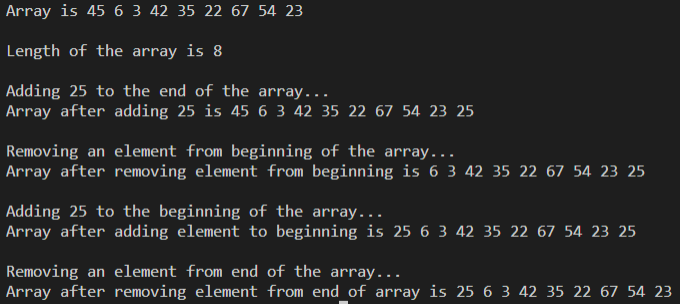
print("Array after adding element to beginning is @array\n");

print("\nRemoving an element from end of the array...\n");

pop(@array);

print("Array after removing element from end of array is @array\n");

OUTPUT:



Q5. Write a perl script create elements in an array like ATGCA, ATTG, AATGC, AAAT and perform the following:

1. Find length of an array
2. Add one element i.e., ATGC at bottom of an array
3. Remove one element at beginning of an array
4. Add one element i.e., ATGCC at top of an array
5. Remove one element at end of an array

CODE:

@element=("ATGCA","ATTG","AATGC","AAAT");

@elem = qw/ATGCA ATTG AATGC AAAT/;

print "The length of the array is $#element\n";

push(@element,"ATGC"); #add elem at the end

print"\nThe array after adding is

@element\n";

shift(@element); #remove elem from start

print"\nArray after removing one element from start

@element\n";

unshift(@element, "ATGCC"); #add elem from start

print"\nArray after adding one element from start

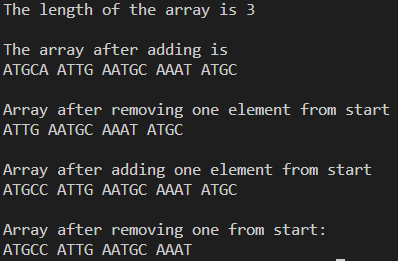
@element\n";

pop(@element); #remove one elem from end

print"\nArray after removing one from start:

@element\n";

OUTPUT:



Q6. Write a perl script to create an array and perform following operations such as merge, reverse and sorting

CODE:

@array1= (1,2,3,4,5,7);

@array2= (8,9,11,10,13,12);

@array3= (@array1, @array2);

@chr= qw/a b d s e f g y z/;

print"@array3\n";

@sorted=sort {$a <=> $b} @array3;

print("@sorted\n");

@rev = reverse(@sorted);

print("@rev\n");

OUTPUT:



Q7. Display array in descending order:

CODE:

@array1= (1,2,3,4,5,7,8,9,10,11,12);

Print(“@array1\n”);

@sorted=sort {$b <=> $a} @array1;

print("@sorted\n");

OUTPUT:



Q8. Write a perl script to store string of an array and display index number 3,4,5 at once

CODE:

@strings = ("ATGU", "ATTG", "ATCG", "TGAC", "TCGA", "TACG", "GCTA");

print(@strings[3..5]);

OUTPUT:



Q9. Write a perl script to demonstrate splice operator

CODE:

@arr = (0..9);

print("Complete array @arr\n");

@replacement = splice(@arr, 3, 4, a..d);

print("Added Elements @arr\n");

OUTPUT:



Q10. Write a perl script to sort hashes using keys

CODE:

%data = ('b' => 2, 'a' => 1, 'e' => 5, 'd' => 4, 'c' => 3);

@data\_sorted = sort(%data);

print(@data\_sorted);

OUTPUT:



Q11. Write a Perl program to determine the frequency of nucleotide bases in given nucleotide sequence using nested if else

CODE:

$seq = <stdin>;

$a = 0;

$t = 0;

$g = 0;

$c = 0;

for($i=0; $i<length($seq); $i++){

$n = substr($seq, $i, 1);

if($n eq "a"){

$a++;

}elsif($n eq "t"){

$t++;

}elsif($n eq "g"){

$g++;

}elsif($n eq "c"){

$c++;

}

}

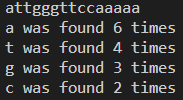
print("a was found $a times\n");

print("t was found $t times\n");

print("g was found $g times\n");

print("c was found $c times\n");

OUTPUT:



Practical 4

Q1. Write a perl program to create a subroutine named calculate and find area and perimeter of rectangle

CODE:

sub Calculate{

print("Give length of the rectangle: ");

$len = <stdin>;

print("Give breadth of the rectangle: ");

$bre = <stdin>;

$area = $len \* $bre;

$peri = 2\*($len + $bre);

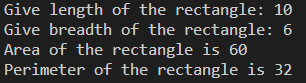
print("Area of the rectangle is $area\n");

print("Perimeter of the rectangle is $peri\n");

}

Calculate();

OUTPUT:



Q2. Write a perl program to create a subroutine named calculate and find area and perimeter of a rectangle with parameters

CODE:

sub Calculate{

my($l, $b) = @\_;

$area = $l\*$b;

$peri = 2\*$l + 2\*$b;

print("Area of the Rectangle is $area\n");

print("Perimeter of the Rectangle is $peri\n");

}

print("Enter length of the Rectangle: ");

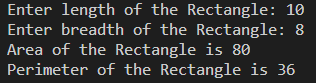
$l = <stdin>;

print("Enter breadth of the Rectangle: ");

$b = <stdin>;

Calculate($l,$b);

OUTPUT:



Practical 5

Q1. Write a perl script to accept a number and create reference of scalar variable and display a value using dereferencing

CODE:

print("Enter a number: ");

$num = <stdin>;

$ref\_num = \$num;

print("Number entered and stored in Reference is ",${$ref\_num},"\n");

OUTPUT:



Q2. Write a perl script to store an array and use reference and dereference

CODE:

@arr = qw/Biology Zoology Mathematics Physics Chemistry/;

$ref\_arr = \@arr;

print("Array entered and stored in Refenrence is ", @{$ref\_arr},"\n");

OUTPUT:



Q3. Write a perl script to store a hash and use reference and derefencing

CODE:

%subjects = (1=>"Biology", 2=>"Zoology", 3=>"Mathematics", 4=>"Physics", 5=>"Chemistry");

$ref\_hash = \%subjects;

print("Hash entered and stored in Refenrence is ",%{$ref\_hash},"\n");

OUTPUT:



Q4. Write a perl script to create a subroutine and use reference and dereferencing

CODE:

sub default{

print("This is a subroutine\n");

}

$sub\_ref = \&default;

print("The reference will be called now\n\n");

&{$sub\_ref;}

OUTPUT:



Q5. Write a perl script to store number in global variable and demonstrate the scope of it

CODE:

sub g\_pr{

$global\_num = 70;

print("Printing global variable inside the subroutine it was declared in: $global\_num\n");

}

g\_pr();

print("Printing global variable outside the subroutine it was declared in: $global\_num\n");

OUTPUT:



Q6. Write a perl script to store a number in private variable and demonstrate the scope of it

CODE:

sub pr\_var{

my $private\_var = 70;

print("Printing Private variable inside the subroutine it was declared in: $private\_var\n");

}

pr\_var();

print("Priting private variable outside the subroutine it was declared in: $private\_var\n");

OUTPUT:



# NOTDONE

Q7. Write a perl script to store a number in state variable and display an OUTPUT

CODE:

use feature 'state';

state $static\_number = 70;

print("Printing Static variable $static\_number\n");

print("Trying to change value of static variable from 70 to 100\n");

$static\_number = "shalmon";

print("Printing Static variable after changing $static\_number\n");

OUTPUT:

