

BIOINFORMATICS INSITUTE OF KENYA

PERL HACKATHON

1. **What is perl?** Perl is a loosely typed scripting programming language created by Larry Wall is 1987. Perl stands for *Practical Extraction and Reporting Language*.
2. **What are the features of Perl programming?**
 - i. Supports unicode
 - ii. Easily intergrates with databases via database intergration interface.
 - iii. Can be embedded in other systems.
 - iv. Works effeciently with markup languages
3. **What are the benefits of Perl programming in using it in biological applications?**
 - i. Easy to program and low learning curve.
 - ii. Rapid prototyping. Fast turn around time in creating applications as compared to compiled languages
 - iii. It is stable because it has undergone years of development.
 - iv. Portable: It can run on most systems. It ships preinstalled in OSX and *nix based systems.
4. **Is perl a case sensitive language?** Perl is case sensitive.
5. **What is a perl identifier?** Is a name used to identify a variable, function, class, module, or other object.
6. **What are data types that perl supports?**
 - i. Scalars
 - ii. Arrays
 - iii. Hashes
7. **What are scalar data types in perl?** This is a single unit of data. Usually starts with \$.


```
#!/usr/bin/env perl
```



```
my $age = 50; # This is a scalar  
my $name = "Albert"; # This is another scalar
```
8. **What are Arrays in perl?** This is a variable that has an ordered list of scalars. They start with (@) sign.


```
#!/usr/bin/env perl
```

```
@person1 = ("Albert", 50); # This is an array
```

9. **What are Hashes in perl?** This is a set of key value pairs.

```
#!/usr/bin/env perl
```

```
my %person1 = ('Name' => "Albert", 'Age' => 50); # This is an hash
```

10. **How will you declare a variable in perl?** Using the *my* keyword.

```
#!/usr/bin/env perl
```

```
my @variable1 = 50; # This is a scalar
```

```
my @variable2 = ("Albert", 50); # This is an array
```

```
my %variable3 = ('Name' => "Albert", 'Age' => 50); # This is an hash
```

11. **What is variable context in perl?** Variables are treated differently in perl based on the data type they are being copied.

12. **What is scalar context?** Assignment to a scalar variable evaluates the right-hand side in a scalar context.

13. **What is a list context?** Assignment to an array or a hash evaluates the right-hand side in a list context.

14. **What is a boolean context?** Boolean context is simply any place where an expression is being evaluated to see whether it's true or false.

15. **What is void context?** This context not only doesn't care what the return value is, it doesn't even want a return value.

16. **What is interpolative context?** This context only happens inside quotes, or things that work like quotes.

17. **What is the difference between single quoted string and double quoted string?** Single quotes will not resolve variables and escapes whereas double quotes will resolve variables, and escape characters.

18. **What is V-Strings?** Also known as version strings. They are used to print out version numbers.

19. **What is the purpose of a `__FILE__` literal?** Used to represent the current file name in a program.

20. **What is the purpose of a `__LINE__` literal?** Used to represent the current line number in a program.

21. **What is the purpose of a `__PACKAGE__` literal?** Used to represent the current package name in a program.

22. **How will you access an element of a perl array?** Prefix the variable with a dollar sign (\$) and then append the element index within the square brackets after the name of the variable.

```
#!/usr/bin/env perl
```

```
my @person1 = ("Albert", 50); # This is an array
```

```
print "$person1[0]\n"; # Accessing first element
```

```
print "$person1[1]\n"; # Accessing second element
```

23. **What is range operator?** Used as a shorthand way to set up arrays.

```
@array = (1..10);
```

24. **How will you get size of an array?** Using the scalar context on the array.

```
#!/usr/bin/env perl
```

```
my @array = (1,2,3);
```

```
print "Size: ",scalar @array,"\n";
```

25. **How will you add an element to an end of an array?** Using *push* keyword.

```
#!/usr/bin/env perl
```

```
my @array = (1,2,3);
```

```
push(@array, 4);
```

26. **How will you add an element to an beginning of an array?** Using *unshift* keyword.

```
#!/usr/bin/env perl
```

```
my @array = (1,2,3);
```

```
unshift(@array, 0);
```

27. **How will you remove an element to an end of an array?** Using *pop* keyword.

```
#!/usr/bin/env perl
```

```
my @array = (1,2,3);
```

```
pop(@array);
```

28. **How will you remove an element to an beginning of an array?** Using *shift* keyword.

```
#!/usr/bin/env perl
```

```
my @array = (1,2,3);
shift(@array);
```

29. **How will you get slice from an array?** One can uses the indices of the various elements in the array to slice it.

```
#!/usr/bin/env perl
my @array = (1..10);
my @newarray = @array[1,3,5,7]; # Slicing using indices.
```

30. **How will you get replace elements of an array?** Using *splice* keyword.

```
#!/usr/bin/perl

@array = (1..20);
print "Before - @array\n"; # Before - 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

splice(@nums, 5, 5, 21..25);
print "After - @nums\n"; # After - 1 2 3 4 5 21 22 23 24 25 11 12 13 14 15 16 17 18 19 20
```

31. **How will you convert a string to an array?** Using the `split()` subroutine one can convert strings to arrays using regular expression.

32. **How will convert an array to string?** The `join()` subroutine can be used to convert a array to a string.

33. **How will you sort an array?** The `sort` keyword is used to sort arrays

34. **What is the purpose of `$[` variable?** Used to set the start of the array to 1.

35. **How will you merge two array?** *push* subroutine will help in jioning two arrays.

36. **How will you create Hashes in perl?** Using `%` followed by a name of the variable then assign keys and value such the code snippet below.

```
#!/usr/bin/env perl

my %person1 = ('Name' => "Albert", 'Age' => 50); # This is a hash
```

37. **How will you get elements from Hashes in perl?** A scalar and reference to the key is used to access the element of a hash

```
#!/usr/bin/env perl
```

```
my %person1 = ('Name' => "Albert", 'Age' => 50); # This is an hash
print "$person{'Name'}\n"; # Accessing a value from the hash
```

38. **How will you get all the keys from Hashes in perl?** The *keys* keyword is used to access the keys from a hash.

```
#!/usr/bin/env perl
```

```
my %person1 = ('Name' => "Albert", 'Age' => 50); # This is an hash
@thekeys = keys %person1; # getting the keys
```

39. **How will you get all the values from Hashes in perl?** The *values* keyword is used to access the keys from a hash.

```
#!/usr/bin/env perl
```

```
my %person1 = ('Name' => "Albert", 'Age' => 50); # This is an hash
@thevalues = values %person1; # getting the keys
```

40. **How will you check if key exists in a hash or not?** Using conditional *if* one can check if a key exists in a hash.
41. **How will you get the size of hash?** The size of the hash can be obtained by changing the list context to scalar context.
42. **How will you add an element to a hash?** Assign a new key and value using the assignment operator will add a new element to the hash.
43. **How will you remove an element from a hash?** *delete* keyword is used here together with the scalar-key reference to remove an element from a hash.
44. **What is the purpose of the next statement?**
45. **What is the purpose of the last statement?** Loop is immediately terminated and the program control resumes at the next statement following the loop.
46. **What is the purpose of the continue statement?** The loop jumps to the next iteration.
47. **What is the purpose of the redo statement?** Restarts the loop block without evaluating the conditional again.
48. **What is the purpose of the goto Label statement?** It jumps to the statement labeled with LABEL and resumes normal execution from there.
49. **What is the purpose of the gote Expr statement?** It is a generalization of goto LABEL. The expression returns a label name

and then jumps to that labeled statement.

50. **What is the purpose of the goto &NAME statement?** For currently running subroutine it substitutes a call to the named subroutine.
51. ****** is exponentiation. $2^{**}2$ will result to 4
52. **<=>** Numeric comparison, returning -1, 0, or 1
53. **lt** Less than
54. **gt** Greater than
55. **le** less than or equal.
56. **ge** greater than or equal.
57. **eq** For comparing string equality.
58. **ne** For comparing non equality of strings
59. **cmp** String comparison, returning -1, 0, or 1
60. ****=** The exponentiation assignment operator.
61. **q{ }** Single Quotes, does not allow interpolation.
62. **qq{ }** Double Quotes, allow interpolation.
63. **qx{ }** Backquote, executes external command inside backquotes.
64. **.** (dot) Concatenation of two strings.
65. **x** Used for repetition.
66. **..** (two dots) Used to depict range. (1..10) will result in numbers from 1 to 10.
67. **++** Increment of values
68. **-** decrement of values
69. **->** Used in referencing
70. **localtime()** This subroutine returns values of current date and time.

SECTION 2

1. Look for the protein HKDC1
 - a. What is the function of this protein? **Catalyses the rate limiting of glucose metabolism.**
 - b. find the orthologs of the protein

HKDC1 - hexokinase domain containing 1

This gene encodes a member of the hexokinase protein family. The encoded protein is involved in glucose metabolism, and reduced expression may be associated with gestational diabetes mellitus. High expression of this gene may also be associated with poor prognosis in hepatocarcinoma. [provided by RefSeq, Sep 2016]

[Genes similar to HKDC1](#)

NCBI Orthologs [How was this calculated?](#)

0 item

Genes Literature

SEARCH THE TAXONOMY TREE

Enter taxonomic name

- Teleostomi
 - birds
 - alligators and others
 - turtles
 - lizards
 - mammals
 - coelacanth
 - bony fishes

180 genes for: *Teleostomi*

☐ 0 selected.

Previous Next

Species	Gene	Architecture	aa	
<input type="checkbox"/> <i>Homo sapiens</i> human	HKDC1 hexokinase domain containing 1		917	▼
<input type="checkbox"/> <i>Mus musculus</i> house mouse	Hkdc1 hexokinase domain containing 1		915	▼

- Do a multiple sequence alignment of the protein with the orthologs to identify a non-conserved site
 - Using perl, write a script that gets out the non-conserved region and inserts a conserved sequence to make any of the 2 orthologs similar.
 - How is this similar to cloning?
 - How can you apply this principle in real life?
- Get the DNA sequence of human UCP2 gene. Write a perl script that:
 - Calculate the number of base pairs in the gene.

```
#!/usr/bin/env perl
```

```
$dna_filename = "sequence.txt";
chomp $dna_filename;
unless ( open(DNAFILE, $dna_filename) )
{
    print "Sorry the file does not exist!!! \n";
    print "Cannot open file \"$dna_filename\" \n";
    die;
}
@DNA = <DNAFILE>;
close DNAFILE;
$DNA = join( ' ', @DNA);
```

```

print " \n The original DNA file is:\n  $DNA \n";
$DNA =~ s/^>//g;
$DNA =~ s/^\s*#//g;
$DNA =~ s/\s//g;
@DNA = split( ' ', $DNA );
$count_of_A = 0;
$count_of_C = 0;
$count_of_G = 0;
$count_of_T = 0;
$errors      = 0;
foreach $base (@DNA) {
    if ( $base eq 'A' ) {
        ++$count_of_A;
    } elsif ( $base eq 'C' ) {
        ++$count_of_C;
    } elsif ( $base eq 'G' ) {
        ++$count_of_G;
    } elsif ( $base eq 'T' ) {
        ++$count_of_T;
    }
    else {
        print "Error - Unknown base: $base\n";
        ++$errors;
    }
}
print "Adenine = $count_of_A\n";
print "Cytosine = $count_of_C\n";
print "Guanine = $count_of_G\n";
print "Thymine = $count_of_T\n";

if ($errors) {
    print "There were $errors unrecognized bases.\n";
}

```

b. Cuts the sequence halfway to give 2 equal fragments

```

#!/usr/bin/env perl

$dna_filename = "sequence.txt";
chomp $dna_filename;
unless ( open(DNAFILE, $dna_filename) )
{
    print "Sorry the file does not exist!!! \n";
    print "Cannot open file \"$dna_filename\"\n";
    die;
}

```



```

}
@DNA = <DNAFILE>;
close DNAFILE;
$DNA = join( ' ', @DNA);
$DNA =~ s/\s//g;
$first_portion = length($DNA)/2;
print "$first_portion\n";
print length($DNA);
$DNA_first = substr $DNA, 0, $first_portion;
$DNA_last = substr $DNA, $first_portion, $first_portion;

print "$first_portion";
print "===== First Portion =====";
print "$DNA_first\n\n";

print "===== Last Portion =====\n";
print "$DNA_last";

```

4. Is the sequence ACCTAGGT palindromic?
 - a. Justify your answer with the aid of a perl script.
 - b. Write pseudocode for the script in (a) above