BIOL-550 Bioinformatics – Take home exam, Fall 2022 (100 pts; 13 questions; 9 pages)

######## General instructions ##

Put all answers inside your **~/EXAM_F22/** directory **on Mozart** unless specified otherwise. Use proper subdirectories (Q001, Q002, Q003…) for each question ## See question 1.

Submit your answers on Blackboard as a single gzipped Tar archive (.tar.gz) that includes the full content of your **~/EXAM_F22/** directory. Make sure to use a descriptive name for your archive, e.*g.* PombertJF_F22exam.tar.gz.

Input files, when required, are in the ~/EXAM_F22/Inputs/ subdirectory. Feel free to copy them to your current working directory for ease of use.

To connect to Mozart by SSH/SFTP, if you are connecting from outside the campus, make sure to activate the IIT VPN first.

######## Questions ##

Objectives – Bash loops, data structure and file permissions
A sane data structure is always a good idea when working on computers, especially when working on multiple projects. Let's create subdirectories in your ~/EXAM_F22/ folder to properly store your data/answers.

Question 1 **5 pts. In your ~/EXAM_F22/ folder:**

a. Create a single Bash loop that does the following commands:
   i. Creates a folder for each question of this exam (*i.e.*, Q001, Q002, Q003…).
   ii. Copies the file ~/EXAM_F22/Inputs/Q01/data_template.txt into each of these folders as data_001.txt in Q001/, data_002.txt in Q002/, data_003.txt in Q003/, and so forth…
b. Run your bash loop.
c. Write your Bash loop in ~/EXAM_F22/F22_myanswers.sh. Feel free to use comments as necessary if you want to add details to your answers.

Question 2 **5 pts. In your ~/EXAM_F22/ folder:**

a. Set recursively the permissions of all subdirectories starting with Q00 to the octal permissions 750.
b. What would be the octal permission (*i.e.* numbers) associated with the **dr-x--x-wx** triplets?
c. Write your command/answers in ~/EXAM_F22/F22_myanswers.sh.

Objective – Transferring files from/to servers back and forth (The basics).
To work remotely efficiently, we must be able to transfer files back and forth between remote servers/workstations and computers acting as terminals whenever needed.

Question 3 **5 pts.** Let's make sure that you can do this:

a. Download this PDF file (EXAM_F22.pdf) from Blackboard to your computer (using any web browser of your choosing), then use the sftp command line tool to transfer EXAM_F22.pdf from your computer to your ~/EXAM_F22/Q002 folder on Mozart
sftp is available from MS Windows, MacOS and Linux terminals.
b. Using sftp, list the content of the remote to your ~/EXAM_F22/Inputs directory
Note that sftp does not load the environment variables from Bash; ~ and $HOME
will not be interpolated; you must use either the absolute or relative paths
c. Using sftp, show the current working directory of the local computer
d. Using sftp, download the file ~/EXAM_F22/Inputs/Q03/sftp.txt from Mozart to your local computer.
e. Write your commands/answers in ~/EXAM_F22/F22_myanswers.sh. Feel free to use comments as necessary if you want to add details to your answers.

Objectives – Using and writing simple bash scripts

A Bash script is a text file that can be used to specify a series of commands for the shell to execute. This script can also contain variables defined by the user. A good shell script can help simplify your work. It also helps with reproducibility by enabling code reuse.

Question 4 **10 pts. In your ~/EXAM_F22/Q004 folder,** create a Bash script titled get_PDBs.pl that does all the following:

a. Uses a proper shebang at the top ## Always a good idea to set this up
b. Creates a variable named BASE_URL with 'https://files.rcsb.org/download' as value
c. Creates a variable named MESSAGE with the following string as value 'All PDB files downloaded!'.
d. Creates a subdirectory ~/EXAM_F22/Q004/PDBs
e. Iterates through a list of PDB entries (7R98,7B3Y,7U09, and 6YI3) with a for loop, and uses either wget or curl to download the corresponding PDB entry from the BASE_URL variable (*e.g.* 'https://files.rcsb.org/download/7R98.pdb) to the ~/EXAM_F22/Q004/PDBs subdirectory.
Note: while you can hard code the PDB entries in your loop, you may also want to
use **$@** instead to feed a list of command line arguments to your loop (this makes
your code more flexible; feel free to look it up!)
f. Creates a single gzipped tar archive named PDBs.tar.gz containing the ~/EXAM_F22/Q004/PDBs subdirectory and its content
g. Displays the message from the MESSAGE variable in the shell once completed.

Make your get_PDBs.pl shell script executable, then run it. Write your commands and the content of your script in ~/EXAM_F22/F22_myanswers.sh.

The RCSB PDB databank is a database of protein structures determined experimentally; we'll
see more about it later this semester.

Objectives – Understanding environment variables and how to install software locally

UNIX-like operating systems use the **PATH** environment variable to store a list of absolute paths. This list is searched by the operating system when an executable is invoked from the command line. This enables us to bypass writing the absolute/relative path on the command line every time we invoke an executable.

Question 5 8 pts. In your ~/EXAM_F22/Q005 folder:

a. Use git to download the scripts from the following GitHub repository: https://github.com/PombertLab/Misc.git
b. Add the corresponding directory ~/EXAM_F22/Q005/Misc to your PATH variable by modifying your ~/.bash_profile accordingly.
c. Source it.
d. Test your configuration by typing read_len_plot.py. ## You should see a short HOWTO if your PATH variable was modified properly.
e. Copy your ~/.bash_profile to ~/EXAM_F22/Q005/my_bashprofile.
f. Write your commands in ~/EXAM_F22/F22_myanswers.sh.

Objectives – Understanding aliases and symbolic links

Long commands can be shortened to simpler ones by creating aliases. This can also be useful to add commonly used flags to a command.

Question 6 7 pts. In your ~/EXAM_F22/Q006 folder:

a. Create an alias lls (for long listing) that will use ls with the -l, -a and -h flags when invoked.
b. Add your alias to your ~/.bashrc configuration file, then source it.
c. Test your newly created alias on your home directory (~) to ensure that it works.
d. Copy your ~/.bashrc configuration file to ~/EXAM_F22/Q006/my_rcprofile
e. Write your commands in ~/EXAM_F22/F22_myanswers.sh.

Symbolic links (symlinks) can be used to create shortcuts to folders and files. This can be very useful with large files, as copying them all over the place would waste a lot of storage space.

Question 7 **5 pts. In your ~/EXAM_F22/Q007 folder**:

a. Create a symlink titled Scripts that points to ~/EXAM_F22/Q005/Misc from Question 5.
b. Write your command in ~/EXAM_F22/F22_myanswers.sh.

*Objective – RTFM: Read the ^f.*g$ manual (Don't know something? No sweat; look it up!).*

Many command-line programs in Unix-like systems come with manuals (known as man pages) accessible from the eponym man program. This can come in handy when you want to see what options are available to you (especially if your Internet connection is down).

The program du is a useful command line tool that can give you an estimate of the space occupied by a folder and its content.

Question 8 **5 pts.**

a. Read the du man page, then use du to show the count for all files in ~/EXAM_F22/Inputs using the human-readable format.
b. Write your command in ~/EXAM_F22/F22_myanswers.sh.

Objectives – Understanding the flow (stream) of information, redirections, and pipes

Many command-line programs in Unix-like systems will default their output data to the standard output stream. We can redirect this stream (or flow) of information with redirection operators and/or pipes.

Question 9 **5 pts. In your ~/EXAM_F22/Q009 folder**:

a. Use zcat to display the content of all .fasta.gz files in ~/EXAM_F22/Inputs/Q09 and redirect its output to ~/EXAM_F22/Q009/concatenated.fasta ## Zcat is like cat but for gzipped files!
b. In a single command line, use grep to search for all lines containing the word Streptococcus, count the number of lines with wc, and write the output of this search to ~/EXAM_F22/Q009/counts.txt.
c. In a single command line, use grep to search for all lines containing the word Staphylococcus, count the number of lines with wc, and append the output of this search to ~/EXAM_F22/Q009/counts.txt.
d. Write your command in ~/EXAM_F22/F22_myanswers.sh.

Objective – Mastering regular expressions

Regular expressions are useful to sift through large amounts of data and capture patterns of interest. Many programming languages use Perl Compatible Regular Expressions (PCRE) built from the Perl 5 regular expression implementation. Learning to use PCRE will be useful not only for Perl also but for other PCRE-compliant programming languages such as Python.

You have received a genome annotation file in GFF format from a collaborator. Your collaborator is only interested in genes coding for tRNA or rRNA features that are located on the forward (+) strand.

Question 10 10 pts. In your ~/EXAM_F22/Q010/ folder

- **a.** Create a regular expression that will find all lines with tRNA or rRNA features (3rd column) and with a positive (+) strandedness (7th column)
  ## See image below for example
- **b.** Test your regex with grep on ~/EXAM_F22/Inputs/Q10/genome.gff. Write the output to ~/EXAM_F22/Q010/Q10_myOutput.tsv. You can compare your results with the expected output in ~/EXAM_S22/Inputs/Q10/Q10_desired_output.tsv using **diff** and/or **wc -l**.
- **c.** Write your regular expression in ~/EXAM_F22/F22_myanswers.sh.

Feature Strandedness/phase

```
111  CP075158.1  Genbank  gene   40831  41718  .  -  .  ID=gene-GPK93_01g00270;Name=GPK93_01g00270;e
112  CP075158.1  Genbank  mRNA   40831  41718  .  -  .  ID=rna-gnl|IITBIO|GPK93_01g00270_mRNA;Parent
113  CP075158.1  Genbank  exon   40831  41718  .  -  .  ID=exon-gnl|IITBIO|GPK93_01g00270_mRNA-1;Par
114  CP075158.1  Genbank  CDS    40831  41718  .  -  0  ID=cds-UTX44519.1;Parent=rna-gnl|IITBIO|GPK93_01
115  CP075158.1  Genbank  gene   41810  42652  .  -  .  ID=gene-GPK93_01g00280;Name=GPK93_01g00280;e
116  CP075158.1  Genbank  mRNA   41810  42652  .  -  .  ID=rna-gnl|IITBIO|GPK93_01g00280_mRNA;Parent
117  CP075158.1  Genbank  exon   41810  42652  .  -  .  ID=exon-gnl|IITBIO|GPK93_01g00280_mRNA-1;Par
118  CP075158.1  Genbank  CDS    41810  42652  .  -  0  ID=cds-UTX44520.1;Parent=rna-gnl|IITBIO|GPK93_01
119  CP075158.1  Genbank  gene   42690  43097  .  -  .  ID=gene-GPK93_01g00290;Name=GPK93_01g00290;e
120  CP075158.1  Genbank  mRNA   42690  43097  .  -  .  ID=rna-gnl|IITBIO|GPK93_01g00290_mRNA;Parent
121  CP075158.1  Genbank  exon   42690  43097  .  -  .  ID=exon-gnl|IITBIO|GPK93_01g00290_mRNA-1;Par
122  CP075158.1  Genbank  CDS    42690  43097  .  -  0  ID=cds-UTX44521.1;Parent=rna-gnl|IITBIO|GPK93_01
123  CP075158.1  Genbank  gene   43179  44405  .  -  .  ID=gene-GPK93_01g00300;Name=GPK93_01g00300;e
124  CP075158.1  Genbank  mRNA   43179  44405  .  -  .  ID=rna-gnl|IITBIO|GPK93_01g00300_mRNA;Parent
125  CP075158.1  Genbank  exon   43179  44405  .  -  .  ID=exon-gnl|IITBIO|GPK93_01g00300_mRNA-1;Par
126  CP075158.1  Genbank  CDS    43179  44405  .  -  0  ID=cds-UTX44522.1;Parent=rna-gnl|IITBIO|GPK93_01
127  CP075158.1  Genbank  gene   44592  44663  .  +  .  ID=gene-GPK93_01g00310;Name=GPK93_01g00310;g
128  CP075158.1  Genbank  tRNA   44592  44663  .  +  .  ID=rna-GPK93_01g00310;Parent=gene-GPK93_01g0
129  CP075158.1  Genbank  exon   44592  44663  .  +  .  ID=exon-GPK93_01g00310-1;Parent=rna-GPK93_01
130  CP075158.1  Genbank  gene   44775  46790  .  +  .  ID=gene-GPK93_01g00320;Name=GPK93_01g00320;e
131  CP075158.1  Genbank  mRNA   44775  46790  .  +  .  ID=rna-gnl|IITBIO|GPK93_01g00320_mRNA;Parent
132  CP075158.1  Genbank  exon   44775  46790  .  +  .  ID=exon-gnl|IITBIO|GPK93_01g00320_mRNA-1;Par
133  CP075158.1  Genbank  CDS    44775  46790  .  +  0  ID=cds-UTX44523.1;Parent=rna-gnl|IITBIO|GPK93_01
134  CP075158.1  Genbank  gene   46995  47705  .  +  .  ID=gene-GPK93_01g00330;Name=GPK93_01g00330;e
135  CP075158.1  Genbank  mRNA   46995  47705  .  +  .  ID=rna-gnl|IITBIO|GPK93_01g00330_mRNA;Parent
136  CP075158.1  Genbank  exon   46995  47705  .  +  .  ID=exon-gnl|IITBIO|GPK93_01g00330_mRNA-1;Par
137  CP075158.1  Genbank  CDS    46995  47705  .  +  0  ID=cds-UTX44524.1;Parent=rna-gnl|IITBIO|GPK93_01
```

Objective – Understanding, using, and writing Perl scripts.

Trying to perform bioinformatic analyses without working knowledge of a scripting language is simply inefficient. Many of the analyses we perform in bioinformatics require either parsing text in some way or that we automate processes to reduce the number of commands lines that we must enter manually. In the next section, we will test your working knowledge of Perl.

FIXING ERRORS. One common challenge when writing scripts is to find the errors that crept in while typing. These can be many things from minor typos to the use of wrong functions. Being able to spot and fix those errors, *i.e.* the debugging process, is a must.

Question 11 10 pts. In your ~/EXAM_F22/Q011/ folder:

a. Copy the file fastQ_to_Oops.pl from ~/EXAM_F22/Inputs/ to ~/EXAM_F22/Q011/fastQ_to_A.pl
b. Find and correct the five (5) errors in fastQ_to_A.pl.
   ## The errors are in the code not the comments!
c. Test your corrected code using ~/EXAM_S22/Inputs/Q11/nanopore.fastq as input
   The desired output file (nanopore.fasta) can be found in ~/EXAM_S22/Inputs/Q11/. You can compare your output to the desired one with **diff** and **wc -l**

AUTOMATING ANALYSES. Doing the same thing over and over manually is both time consuming and error prone. Let's create a simple Perl script to help automate a typical analysis in bioinformatics.

Mapping sequencing data to a reference genome is a common analysis in bioinformatics. This process is known as read mapping and is often used to infer genetic diversity between isolates (we'll learn more about this later this semester). Many tools exist to perform this task and they often differ in their command lines. Automating the use of these tools can help prevent errors and reduce user hands-on time.

Winnowmap (https://github.com/marbl/Winnowmap) is a new read mapping tool for sequencing data produced by long read sequencing platforms (*e.g.* Oxford Nanopore, Pacific Biosciences). This tool is optimized to handle repetitive sequences like those found in telomeres and is based on another read mapper (minimap2).

Using this tool involves two main steps. In the first, repeats are estimated based on kmers (we'll learn about those soon). In the second, reads are mapped to the genome while accounting for the repeats identified in the first step, producing a SAM (sequence alignment map) file.

Typical ***winnowmap*** command lines are:

```
## Setting a few variables
reference=reference.fasta
kmers=repetitive_k15.txt
fastq=~/EXAM_S22/Inputs/Q12/nanopore_1.fastq.gz
output=output_1.sam
```

```
## Repeat estimation step; the command line is broken with \ for readability
meryl count \
  k=15 \
  output merylDB $reference

meryl print \
  greater-than distinct=0.9998 \
  merylDB > $kmers

## Read mapping step; the command line is broken with \ for readability
winnowmap \
  -W $kmers \
  -ax map-ont \
  $reference $fastq > $output
```

As you can imagine, running these steps on several FASTQ files can quickly become tedious and error prone! Let's make a simple Perl script to automate this process.

IMPORTANT INFORMATION: Pathing was not implemented properly in meryl; absolute paths will not work (it is a bug in its code; relative paths are fine). To prevent problems, you can copy ~/EXAM_S22/Inputs/Q12/reference.fasta to your working directory, create a symlink, or use a relative path. Feel free to ask us about it if you struggle with this issue

```
# Meryl using a full path will stop and throw an error message:
meryl count k=15 output merylDB ~/EXAM_S22/Inputs/Q12/reference.fasta
# Can't interpret '/home/jpombert/EXAM_S22/Inputs/Q12/reference.fasta': not a meryl
# command, option, or recognized input file.

# Meryl using a relative path runs without issue:
meryl count k=15 output merylDB ./Inputs/Q12/reference.fasta
```

Question 12 10 pts. In your ~/EXAM_F22/Q012/ folder, create a Perl script named run_winnowmap.pl that does the following:

a. Takes a list of FASTQ files from the @ARGV and stores it in @FASTQ.
```
### Winnowmap supports both .fastq files and gzipped fastq files (.fastq.gz):
## no need to decompress the provided .fastq.gz files
# Optional: you can use GetOptions(); to create command line switches if you want;
# it is a great way to parse the command line arguments stored in @ARGV
```

b. Defines variables for the reference genome (reference.fasta) and kmers text file (repetitive_k15.txt) to be used in the steps below.
```
## You can either hard code these variables, grab them from @ARGV, or even use
## GetOptions(); to create command line switches, TIMTOWTDI !
## Make sure to use a relative path for reference.fasta to prevent issues with meryl
```

c. Tells the operating system to run the meryl count step.

d. Tells the operating system to run the meryl print step.

e. Iterates through each of the .fastq.gz files (hint: loop) and tells the operating system to run the read mapping step on the current .fastq.gz file.
   ## Make sure that the output SAM files have different names between iterations
   ## to prevent them from be overwritten by new ones!

f. Prints a message in the shell stating that all the alignments were performed once completed.

Test your code on the .fastq.gz files provided in ~/EXAM_S22/Inputs/Q12/. As usual, you can compare your output files with the expected ones in ~/EXAM_S22/Inputs/Q12/ using **diff** and **wc -l**.

```
## Note that the SAM file contains the command line as part of the output. If your command
## line differs from the provided desired outputs, you will likely see a difference with the
## following lines below. This is fine.
##
## 2c2
## < @PG   ID:Winnowmap    PN:Winnowmap    VN:2.03 CL:winnowmap -W repetitive_k15.txt -
## ax map-ont ../Inputs/Q12/reference.fasta ../Inputs/Q12/nanopore_1.fastq.gz
## ---
## > @PG   ID:Winnowmap    PN:Winnowmap    VN:2.03 CL:winnowmap -W repetitive_k15.txt -
## ax map-ont reference.fa Inputs/Q12/nanopore_1.fastq.gz
```

PARSING TEXT. Bioinformatic analyses often rely on the use of several tools sequentially, with the output of one tool fed to another. Sometimes, this output must first be reformatted into a format that is recognized by the next tool. Being able to parse text to reformat it into another data structure is often very useful.

BLAST (https://blast.ncbi.nlm.nih.gov/), short for Basic Local Alignment Search Tool, is used to search for homology between nucleotide or amino acid sequences. This tool can be used via a web portal or from a command line interface, either locally or on remote computers.

Your PI (principal investigator) has provided you with the output of a BLAST search performed recently. This output file (queries.blastp.6) is a standard tab-delimited file featuring a total of 12 columns (https://www.metagenomics.wiki/tools/blast/blastn-output-format-6). While useful, this format does not provide you with the molecular functions of the matches found.

Your PI has asked you to parse the content of this file to remove extraneous columns and to add the molecular functions associated with the corresponding matches. These functions are in another tab-delimited file named (products.txt). ## Both files are in ~/EXAM_S22/Inputs/Q13/

Question 13 **15 pts. In your ~/EXAM_F22/Q013/ folder,** create a Perl script named blast_parser.pl that does the following:

a. ***STEP 1:*** Takes the queries.blastp.6 and products.txt files together with an output file (parsed_output.tsv) from the list of argument variables (@ARGV) and creates filehandles to read from queries.blastp.6 and products and to write to parsed_output.tsv. ## You can use positional arguments, or alternatively create command line switches with GetOptions();

b. ***STEP 2:*** Initializes a hash named %products.

c. Iterates through the content of the parsed_output.tsv file line per line and grabs the protein name and its associated function using either a regular expression or split(); while ignoring comments (if any). ## See below

```
1  ### CHROMOSOME 01 ###
2  A3770_01p00010  hypothetical protein
3  A3770_01p00020  cytochrome P450
4  A3770_01p00030  hypothetical protein
5  A3770_01p00040  coiled-coil domain-containing protein
```

comment to be ignored

Protein names Associated molecular functions

d. Adds the protein name (key) and its function (value) to %products.

e. ***STEP 3:*** iterates line per line through the content of queries.blastp.6 and grabs the query (qseqid; 1st column), the match (sseqid; 2nd column) and the evalue (evalue; 11th column).

```
1  HOP50_01g00150  A3770_01p00230  100.000  228  0  0  1  228  1  228  1.75e-168  461
2  HOP50_01g00150  A3770_02p12540  40.574  244  114  9  3  221  7  244  4.35e-45  149
3  HOP50_01g00150  A3770_01p00710  40.000  225  120  5  3  217  12  231  2.71e-44  147
4  HOP50_01g00150  A3770_12p66310  40.271  221  96  6  34  220  47  265  2.19e-43  145
5  HOP50_01g00150  A3770_05p37580  41.451  193  101  4  35  217  52  242  2.49e-42  144
```

query match evalue

f. Prints the query, match, evalue and the function associated with this match (from %products) to the output file using a tab-delimited format
See Q11_desired_output.tsv for details.

As usual you can use **diff** and **wc -l** to compare your output to the desired one.