INF5380 / INF9380 - High Performance Computing in Bioinformatics
Final exam, Spring 2022

General information about the exam:

- This is a practical home exam.
- All materials and tools are allowed.
- The exam must be solved individually.
- Requirements for assignments at the Department of Informatics must be followed. The rules can be found online here: https://www.uio.no/english/studies/examinations/compulsory-activities/mn-ifi-mandatory.html
- For INF5380 (master students): Only the first part of the exam must be completed. The exam will be graded from A to F.
- For INF9380 (PhD students): Both parts of the exam must be completed. The exam will be graded as passed or failed.
- Read through all exam questions first and make sure you understand all of it before you start working. Contact the course responsible with any questions.
- Write a report where you describe how you have solved each task. Include all command lines, scripts, code and config files used, so that your procedure may be replicated in detail. Show all relevant results (a brief excerpt from those that take up too much space), and answer all questions.
- The report may be written in English or Norwegian.
- The exam should be anonymous, do not include your name in the report.
- The report in the form of a single PDF document must be submitted no later than Wednesday 25 May 2022 at 17.00 through Inspera (https://uio.inspera.no/).
- Contact person regarding the exam, for technical question or problems, or if there is anything unclear about the exam questions:
  Torbjørn Rognes, email torognes@ifi.uio.no, mobile phone 907 555 87.

Data and tools to be used in the exam

Files available on Fox:
/projects/ec34/inf9380/exam_2022/

```
data/
        control.fq.gz
        subject.fq.gz
programs/
        compare.py
ref/
        ref.fa
        TruSeq3-SE.fa
```

The files are also available on FileSender at this URL:
https://filesender2.uio.no/?s=download&token=df3fa88f-604a-407c-921b-f62a15176f57

The Docker image "arvindsundaram/vc_norbis" containing the tools FastQC, Trimmomatic, bwa, bcftools, etc is available on DockerHub at the URL below. You may use this Docker image when solving the problems in this exam.
https://hub.docker.com/r/arvindsundaram/vc_norbis

Part 1: Variant calling workflow on Fox (everyone)

Use your existing account on Fox to solve this part. You will need to run the tools using Singularity. Feel free to use the NREC resources if required.

Write a report where you describe how you have solved each exam problem. Include all command lines, scripts, code and config files used, so that your procedure may be replicated in detail. Show all relevant results (a brief excerpt from those that take up too much space), and answer all questions.

Find the number of variants found in chr 18 (ref.fa) in the two mice datasets: control.fq.gz and subject.fq.gz (single end sequence data), and report the variants specific to the sample subject. Write Slurm scripts to perform each step (a-f) in the workflow below. Reserve 4 cores on 1 node on Fox and run the tools using multiple threads where possible.

a) We are going to run several tools, including FastQC, Trimmomatic, BWA and bcftools. Why is it necessary to use Singularity to run these tools on Fox? How can you most easily run these tools using Singularity?
b) First perform a quality check of the input fastq files (data/control.fq.gz and data/subject.fq.gz) using FastQC.
c) Clean the fastq files using Trimmomatic (Use SE option; ref/TruSeq3-SE.fa adapter file).
d) Create the bwa index for the reference (ref/ref.fa) (mouse chromosome 18).
e) Map the fastq files to the reference using bwa mem (default parameters; change the number of threads though).
f) Use BCFtools mpileup to calculate genotype likelihoods and BCFtools call to find variants.
g) Use the tool provided in programs/compare.py to calculate subject specific variants ($ python compare.py <ONE.vcf> <TWO.vcf>; variants found only in <TWO.vcf> appear in the output).

Part 2: Variant calling workflow using the NREC cloud (PhD students only)

In this part you will create and run a variant calling workflow similar to the one described in part 1, but this time using NREC resources and Docker as demonstrated during the course. Usernames (`studentXX@inf9380`) and passwords for the NREC API were distributed by email on 15 March and the NREC ssh key pair files (`inf9380-2022-ssh` and `inf9380-2022-ssh.pub`) on 24 March. All nodes have been reset after the course. Log in to the admin node with the IP address allocated to you and distributed by email on 25 April.

Write a report where you describe how you have solved each exam problem. Include all command lines, scripts, code and config files used, so that your procedure may be replicated in detail. Show all relevant results (a brief excerpt from those that take up too much space), and answer all questions.

a) Explain the possible advantages and disadvantages of using a cloud resource (like NREC) compared to a HPC resource with a queuing system (like Fox).

b) Set up your admin machine with ssh keys etc. Show the commands used.

c) What is the admin machine used for? Why has an admin machine been created on NREC for you to use? What would you do if you did not have this admin machine to work from?

d) Install Terraform, the Openstack client and other necessary or useful software packages on the admin machine. Show the commands needed to do this.

e) Set up Openstack authentication and create the `~/keystonerc` file. Show the contents of this file in your report.

f) Create the `basic.tf` terraform configuration file. Use one single compute node of the `large` flavor (with 2 cpus, 8 GB RAM and 20 GB disk). Use the `GOLD CentOS Stream 8` image. Show the contents of the configuration file.

g) Create resources with Terraform. Show the commands and their output.

h) Log in to the compute node and copy the data from Fox to a new folder on the compute node using the following command (replace `ec-username` with your own username):

```
scp -r ec-username@fox.educloud.no:/projects/ec34/inf9380/exam_2022 .
```

i) Install and set up Docker on the compute node. Show the commands needed.

j) Adapt the scripts you wrote for each step in the workflow in part 1 to run using Docker on the compute node, taking advantage of both cpus. This time you will not use Slurm. Show the new scripts.

k) Run the workflow and compare the results to what you obtained in part 1. Did you expect the results to be identical or different? Why? Discuss any differences in the results from your expectations.