# 2015 Computational Bioscience Program Preliminary Exam Day 3

## General instructions

You are free to choose whatever programming language you like, but **in general you may not use any bioinformatics toolkits or libraries**. You *are* allowed to use libraries for calculation of common statistical tests and p-values (like t-test, Wilcoxon, Kolmogorov Smirnoff, etc), but you must fully cite the source. If there is any question about the eligibility of a specific statistical test, do not hesitate to mail sonia.leach@gmail.com. Also, if you are programming in C++, the use of the Standard Template Libraries is allowed. If you are programming in python, you may use scipy and the numpy multidimensional array library. If there are questions, please send an email to sonia.leach@gmail.com; reasonable accommodations will be made with the goal of standardizing available data structures across languages that students may choose.

This portion of the exam is a test of your individual algorithmic design and programming skills. *Do not communicate with anyone else about the exam*. If you want clarification about anything else, you may email sonia.leach@gmail.com.

## Submission

You must submit your exam, which consists of the following components, by **8:00 a.m., Wednesday, June 10** via email to Anis.Karimpour-Fard@ucdenver.edu and Sonia.Leach@gmail.com:

1. **Written report**
2. **Algorithm pseudocode**
3. **Documented program code**
4. **Program Output (gzipped)**

Provide in your submission a README.txt file that includes detailed, concise instructions on how to install your program and its dependencies within a basic environment. To ease the burden on the graders (Anis and Sonia), we ask that you be available to troubleshoot should there be difficulties with installation in the remainder of the week following the exam submission. **Better yet would be to email Anis a 'Hello World' type program written in your language of choice before your submission (say by Friday June 5) so we can work out any installation issues.**  If you use additional data sources, you must upload and fully describe these data sets as well. If the submission becomes too big for email, please send the report and pseudocode via email and provide a dropbox link shared to Anis and Sonia that contains the remainder.

## Grading

Your work will be graded based upon three components: a written report (25%), algorithm pseudocode (25%), and implementation (50%). **The written report** should describe and justify your strategy, define the input files, present the algorithm overview, define any scoring methods, and detail the expected output files. The report should also contain an analysis of the final results and discussion. You should cite the appropriate scientific literature where appropriate. **The pseudocode** should describe and contain comments for each step of the process. **The implementation** must include a working program with extensive comments describing each step, as well as describing all input and output files.

## Programming problem

There are many applications wherein a researcher would like to know the sequence context around a given query sequence they suspect exists in their sample. For example, gene targeting may be used to create a knock-out model and the researcher would like to verify that the target vector was incorporated into the right place in the genome. Alternatively, a researcher might wish to fully identify suspected contaminating sequences that would indicate the presence and/or source of unclean sample handling procedures in the laboratory, such as a specific PCR primer contamination.

**Create a program that takes as input the set of all next-generation sequencing reads identified in a sample and an initial query sequence and returns the largest sequence contig that can be constructed from the reads that contains the initial query sequence.**

**DETAILS:**

**Input: (download from dropbox link provided with distribution email)**

**QUERY.fasta** - fasta file containing initial query sequence (size: 1 KB)

**READS.fasta.gz** - gzipped fasta file of sequencer reads (size 5 MB, gunzip before use)

*Note: The files provided are data from a* ***real*** *sequencing run, with all inherent* ***errors and artifacts****.*

**Output:**

**ALLELES.fasta** - fasta file of the largest constructed contig (allele) containing the initial query **ALLELES.aln** -

tab-delimited text file describing alignment of sequence reads to contig(s) in ALLELES.fasta with the following columns (see example further below):

|  |  |
| --- | --- |
| sseqid | : name of sequencing read (from READS.fastq.gz) |
| qseqid | : name of contig matched (from ALLELES.fasta) |
| sstart | : starting coordinate in sequencing read sseqid that matches qseq |
| send | : ending coordinate in sequencing read sseqid that matches qseq |
| qstart | : starting coordinate in contig that matches sseq |
| qend | : ending coordinate in contig that matches sseq |

Coordinates should respect that the contig in ALLELES.fasta is in the forward orientation, such that if the sequencing read sseq aligns in the forward direction with respect to the contig, send > sstart, whereas if sseq aligns in the reverse direction with respect to the contig, send < sstart (see italics in example below). It is not necessary to include identifiers for sequences from READS.fastq.gz that do not align to the sequence in ALLELES.fasta. Additional informative columns may be added to the text file, as you deem useful or appropriate.

Example output.aln file:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sseqid | qseqid | sstart | send | qstart | qend |
| 2S43D:08461:04180 | contig1 | 13 | 40 | 1 | 64 |
| 2S43D:07701:07310 | contig1 | 20 | 112 | 240 | 332 |
| 2S43D:07489:10315 | contig1 | *123* | *90* | 20 | 53 |
| 2S43D:04035:14719 | contig1 | *105* | *41* | 10 | 74 |

*Hint: at the most basic level, you need a function to compare (parts of) two sequences.*

*Bonus: Assuming the organism was diploid and the data resulted from a gene targeting knockout-model, describe how you would use/ modify your program to determine whether the organism was a heterozygote or a homozygote for the knock-out.*