**BNFO 601: Integrated Bioinformatics**

**EXAM 1**

**(18 February 2022)**

***The primary purpose of this exam is to serve as an educational tool.*** With this in mind, you should not be surprised if some of the problems go beyond your present abilities and try to connect things that may not yet have been connected. Nonetheless, most of the elements have been drawn from the modules. If you ask yourself, “Have I seen something like this before?” the answer is almost always “Why yes, yes I have!”

**RULES OF THE GAME**

• **Available resources:** This is an open book exam. Most important, it is an open brain exam. That said, you are forbidden from using internet material that specifically arose from this class (*e.g.,* if a former student posted answers from a previous year’s exam, that would be off limits).

**• Unavailable resources:** Needless to say, this is not an open people exam. **Major exception:** You may consult with Paul or Quinn

**• Contact info:** Paul: pfawcett@vcu.edu, Quinn Barrow: barrowqa@vcu.edu

**• What to submit: PAY ATTENTION TO THIS!!**

Electronic submission is *required*. The preferred format is one .zip file containing all of your answer files (**don’t send a whole bunch of separate attachments, *as it will not work* – my email system prevents me from downloading code sent directly as attachments**). When sending files / zipped archives, be sure that the filenames begins with your name so I don’t get a mailbox full of files named “Exam1” (*e.g.,* use something like: last\_name\_BNFO601\_Exam1.zip). When asked to provide a program, submit both the program file, the required input files, and, if possible, at least a sample of the output. The program files should be named as indicated for each question. It’s generally a good idea to submit material that shows your thought processes. Comments in computer programs are a definite "Good Thing" that will help determine if you were on the right track... Items highlighted in red correspond to files that will be a mandatory part of your submission

**• IMPORTANT!** Your submitted folder should include **ALL of the files** I need for your programs to *just run* (even if I was the one that originally provided the files, such as input files, include them in your zip archive if they are needed for your program to run). **Similarly, the file paths that you use in your programs should be relative to the folder name you submit** *(i.e.,* do **NOT** use hard coded paths that are specific to your own computer). I don’t want to mess around copying necessary files into your folder or adjusting path names in order to get your programs to run correctly -- it’s a huge PITA that slows down the grading process. Once again, everything your program needs to run should be included in the folder you submit. ***DO NOT FAIL ME ON THIS.***

• **When to submit:** Everything must be submitted by **11:59 AM, Wednesday, 23 February 2020**, and not later. We will be starting a new module on Monday morning, and I want your full attention on the new material!

**In your submission you must indicate that you have neither received nor given aid during the preparation of your exam answers and agree to abide by the VCU Honor policy (instructor aid excepted). Please include the following text:**

***I have neither received nor given aid on this exam and agree to abide by the VCU Honor Policy***

No collaborations! Infractions of this policy are treated with utmost seriousness.

The honor system policy: <https://students.vcu.edu/studentconduct/vcu-honor-system/>

***The Questions***

**QUESTION 1:** We have already encountered several examples in this course of the FASTA format for DNA and amino acid sequence files. FASTA is commonly used format, and is very popular because it is a very “lightweight” textual file format that is easy to parse and to work with, while maintaining a the characteristic of being readable by humans (note that most binary file formats, while extremely compact, do not directly permit reading by a human -- except perhaps exceptionally geeky ones).

Minimally, the FASTA format just consists of a line of annotation preceded by the character “>”, followed by one or more lines containing either DNA or protein sequence information. For example, check out these two sequences in FASTA format taken from the *E. coli* 104:H4 protein file we mentioned in class (this was the “other” pathogenic *E.coli*):

>gi|340736248|gb|EGR65296.1| 2-isopropylmalate synthase [Escherichia coli O104:H4 str. 01-09591]

YKEXEYNLDNLYDAFLKLADKKGQVFDYDLEALAFIGKQQEEPEHFRLDYFSVQSGSNDIATAAVKLACGEEVKAEAANGNGPVDAVYQAINRITDYNVELVKYSLTAKGHGKDALGQVDIVANYNGRRFHGVGLATDIVESSAKAMVHVLNNIWRAAEVEKELQRKAQHNENNKETV

While FASTA is great for many applications, it brings with it certain drawbacks, most noticeably a significant limitation with respect to how much and what kind of annotation detail can be included. There are therefore several alternatives to FASTA. The most famous and widely used of these is probably the GenPept format that was developed at NCBI for use with their GenBank database. A description of the format can be found here:

https://www.unmc.edu/bsbc/education/courses/gcba815/formats.html

Compare the FASTA entry above for 2-isopropylmalate synthase with its corresponding GenPept format entry:

LOCUS EGR65295 363 aa linear BCT 20-JUL-2011

DEFINITION 3-isopropylmalate dehydrogenase [Escherichia coli O104:H4 str.

01-09591].

ACCESSION EGR65295

VERSION EGR65295.1 GI:340736247

DBLINK BioProject: [PRJNA67931](https://www.ncbi.nlm.nih.gov/bioproject/PRJNA67931)

BioSample: [SAMN02470246](https://www.ncbi.nlm.nih.gov/biosample/SAMN02470246)

DBSOURCE accession [AFPS01000001.1](https://www.ncbi.nlm.nih.gov/nuccore/340736170)

KEYWORDS .

SOURCE Escherichia coli O104:H4 str. 01-09591

ORGANISM [Escherichia coli O104:H4 str. 01-09591](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=1042803)

Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriales;

Enterobacteriaceae; Escherichia.

REFERENCE 1 (residues 1 to 363)

AUTHORS Mellmann,A., Harmsen,D., Cummings,C.A., Zentz,E.B., Leopold,S.R.,

Rico,A., Prior,K., Szczepanowski,R., Ji,Y., Zhang,W.,

McLaughlin,S.F., Henkhaus,J.K., Leopold,B., Bielaszewska,M.,

Prager,R., Brzoska,P.M., Moore,R.L., Guenther,S., Rothberg,J.M. and

Karch,H.

TITLE Prospective Genomic Characterization of the German

Enterohemorrhagic Escherichia coli O104:H4 Outbreak by Rapid Next

Generation Sequencing Technology

JOURNAL PLoS ONE 6 (7), E22751 (2011)

PUBMED [21799941](https://www.ncbi.nlm.nih.gov/pubmed/21799941)

REFERENCE 2 (residues 1 to 363)

AUTHORS Cummings,C.A., Mellmann,A., Harmsen,D., Prior,K., Szczepanowski,R.,

Leopold,B., Leopold,S., Karch,H., Rico,A., Guenther,S.,

Greither,R., Myers,J., Cifuentes,F., Ji,Y., McLaughlin,S. and

Brzoska,P.

TITLE Direct Submission

JOURNAL Submitted (10-JUN-2011) Life Technologies, 850 Lincoln Centre

Drive, Foster City, CA 94404, USA

COMMENT Reads were mapped with TMAP against the publicly available E. coli

55989 chromosome (CU928145.2) and the derived consensus was split

into contigs at zero-coverage regions. These contigs were used as

a 'backbone' for mapping of reads, followed by de novo assembly of

unmapped reads with the MIRA assembler (v 3.2.1). A small number

of de novo and consensus contigs were merged using CAP3. The strain

is available from the German National Consulting Laboratory for

Hemolytic Uremic Syndrome (HUS) at the Institute of Hygiene,

University Hospital Muenster.

Annotation was added by the NCBI Prokaryotic Genomes Automatic

Annotation Pipeline Group. Information about the Pipeline can be

found here:

<http://www.ncbi.nlm.nih.gov/genomes/static/Pipeline.html>. Please be

aware that the annotation is done automatically with little or no

manual curation.

##Genome-Assembly-Data-START##

Assembly Method :: MIRA v. 3.2.1

Assembly Name :: Ecoli.O104:H4.01-09591\_1.0

Genome Coverage :: 24x

Sequencing Technology :: Ion Torrent PGM

##Genome-Assembly-Data-END##

Method: conceptual translation.

FEATURES Location/Qualifiers

source 1..363

/organism="Escherichia coli O104:H4 str. 01-09591"

/strain="01-09591"

/serovar="O104:H4"

/host="Homo sapiens"

/culture\_collection="HUSEC<DEU>:041"

/db\_xref="taxon:[1042803](https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=1042803)"

/country="Germany: Cologne"

/collection\_date="2001"

[Protein](https://www.ncbi.nlm.nih.gov/protein/340736247?from=1&to=363&sat=37&sat_key=76480151) 1..363

/product="3-isopropylmalate dehydrogenase"

/EC\_number="[1.1.1.85](http://www.expasy.org/enzyme/1.1.1.85)"

[Region](https://www.ncbi.nlm.nih.gov/protein/340736247?from=3&to=362&sat=37&sat_key=76480151) 3..362

/region\_name="PRK00772"

/note="3-isopropylmalate dehydrogenase; Provisional"

/db\_xref="CDD:[234832](https://www.ncbi.nlm.nih.gov/Structure/cdd/cddsrv.cgi?uid=234832)"

[CDS](https://www.ncbi.nlm.nih.gov/nuccore/340736170?from=81361&to=82452&sat=37&sat_key=76480151) 1..363

/locus\_tag="HUSEC41\_00380"

/coded\_by="complement(AFPS01000001.1:81361..82452)"

/inference="DESCRIPTION: similar to AA

sequence:ProteinCluster:PRK00772"

/inference="EXISTENCE: similar to AA

sequence:RefSeq:YP\_003227177.1"

/note="COG0473 Isocitrate/isopropylmalate dehydrogenase"

/transl\_table=[11](https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi?mode=c#SG11)

ORIGIN

1 msknyhiavl pgdgigpevm tqalkvldav rnrfamritt shydvggaai dnhgqplppa

61 tvegceqada vlfgsvggpk wehlppdqqp ergallplrk hfklfsnlrp aklyqgleaf

121 cplradiaan gfdilcvrel tggiyfgqpk gregsgqyek afdtevyhrf eieriariaf

181 esarkrrhkv tsidkanvlq ssilwreivn eiateypdve lahmyidnat mqlikdpsqf

241 dvllcsnlfg dilsdecami tgsmgmlpsa slneqgfgly epaggsapdi agknianpia

301 qilslalllr ysldaddaac aierainral eegirtgdla rgaaavstde mgdiiaryva

361 egv

//

There’s a lot more information there! On the other hand, it’s maybe a lot tougher to work with if you are interested mostly just in the raw sequence and some minimal annotation sufficient to get your work done.

**Q1a)** Your mission, should you choose to accept it, is as follows. Use your knowledge of regular expressions to create and turn into me a program that can convert a GenPept format file (which might contain many individual peptides, perhaps even a whole genome’s worth) into a FASTA format file, which is output to disk. This file will be called:

**“Yourname\_convert\_GP\_to\_FASTA.py”**

You can use as a starting point any of the other programs we have worked with in the class, or you can write your own from scratch, perhaps using our previous programs just for inspiration (which might be just as easy). I’m not too fussy about the particular approach you take, but I would like for you to bundle up your various methods in a nice little class, and you should take care to ensure that your program is sufficiently general that it will work with most or all files that are in GenPept format, and is **not** just cooked up to work only with the specific O104:H4 file at hand! To test your program, I might even try running your program with different GenPept input files to see how it handles it.

It would be ideal if your program had a separate class method for returning the sequence in a FASTA format file.

Note that your program will be required to convert the entire *E. coli* O104:H4 peptide file from the GenPept format to the FASTA format. So first we will have to generate a whole-genome GenPept format *E. coli* O104:H4 peptide file. Do this by steering your browser to:

[https://www.ncbi.nlm.nih.gov/protein/?term=txid1042803[Organism:noexp](https://www.ncbi.nlm.nih.gov/protein/?term=txid1042803%5bOrganism:noexp)]

This is the same link we used previously to generate a FASTA file, and can be found also on the class website. But this time, navigate to the “Send To” dropdown on the upper right, and when prompted to choose destination, select “File”, then when prompted for format, select “GenPept” [NOT “GenPept (full)”].

This will create a large GenPept format file. Save it as **“O104\_H4\_GP.txt”**. It is this file that your program will read as input. It should write as output a file named **“O104\_H4\_FASTA.txt”**.Needless to say, the output file should look very similar and should contain the same annotation information as we would have produced had we simply downloaded the genome file from GenBank in FASTA format to begin with. I recommend a strategy where regular expressions are first used to harvest all of the required annotation information, which can then be concatenated together into a single annotation line. Likewise, note that the GenPept sequence information itself is in a little bit of a different format than with FASTA. For a start, it uses numbers to begin each line, which FASTA does not. It also uses spaces as a delimiter, and renders the single letter amino acid code in lower rather than upper case. You will need to address all of these issues, possibly filtering out things like digits and extraneous whitespace!

The good news for you is that the GenPept format is extremely regular in structure… there should, for instance, be no confusion about when one sequence is ending and the annotation for another sequence is beginning. Again, pay close attention to the structure of the problem you are dealing with. Look for and consider special cases!

**QUESTION 2:**

***Preliminaries.*** Restriction enzymes, properly known as restriction endonucleases, are a class of enzymes produced in prokaryotes (*i.e.* bacteria and archae) that cleave DNA following recognition of specific sequences in DNA known as restriction sites. There are a number of classes of restriction endonucleases, each of which has its own weird and wonderful properties. But probably the most commonly encountered are the so-called class II restriction enzymes. The sites recognized by these are typically palindromic and contiguous (*i.e.* there is no spacer region between the palindromes – unlike the 8 bp spacer we saw separating the GAT and TAC components of the canonical NtcA binding site). Each restriction enzyme of this class (and there are hundreds known) recognizes a distinctive cut site, usually between 4 and 8 nucleotides in length. Recognition sites of size six are perhaps the most common. For example, the restriction enzyme EcoRI recognizes the site 5’ - G/AATTC – 3’, (which, naturally, has a palindromic, complementary sequence of 3’ - CTTAA/G – 5’ on the other strand). H***ere the / character interspersed with the nucleotides indicates the position of the phosphodiester bond that is the actual cut site.*** You’ll notice something funny if I write both strands together as:

**EcoRI recognition site:**

5’ – N-N-N-N-G/A-A-T-T-C-N-N-N-N – 3’

| | | | | | | | | | | | | |

3’ – N-N-N-N-C-T-T-A-A/G-N-N-N-N – 5’

Note that the cut sites are not in the same positions on both strands (at least in this case). It turns out that while restriction enzymes cleave phosphodiesters bond of helical DNA, they can either cleave both strands in the same place to yield a blunt end, or at a staggered position leaving overhangs called “sticky ends”, as shown in the products of the EcoRI enzyme digestion:

**Product 1:**

**5’ –** N-N-N-N-**G – 3’**

| | | | |

**3’ –** N-N-N-N-**C-T-T-A-A – 5’**

**Product 2:**

**5’ – A-A-T-T-C-**N-N-N-N **– 3’**

**| | | | |**

**3’ - G-**N-N-N-N **– 5’**

Note that depending on the position of the actual cut site, the products can have 5’ overhangs (as shown here), a blunt end, or a 3’ overhang. In any case, sticky ends turn out to be exceptionally useful for molecular biology experimentation -- if two different DNA molecules have matching ends, they can be very efficiently joined by the enzyme DNA ligase. Blunt ends can be joined together too, albeit less efficiently, and without the selectivity conferred by the complementary overhangs. Alternatively, the sticky ends can be “blunted” using T4 DNA Polymerase, which catalyzes the synthesis of DNA in the 5´→ 3´ direction (i.e. will fill in 5’ overhangs) and also has a 3´→ 5´ exonuclease activity (which will chew back 3’ overhangs).

Because of these useful properties, restriction enzymes are a staple of any modern molecular biology lab.

***Your mission.*** I have provided you with a bare skeleton of a program called

**YOURNAME\_Restriction\_Digestion.py**

Following modifications that you will make, this program should eventually accept a DNA sequence file as input (which I have provided as **mystery\_genome.fasta**), and which will then analyze the sequence for the presence of restriction sites.

Note that when you try and download this file from the Google drive (look for the download icon in the top right of the screen when you click on the file), it will give you a scare message that the file type might possibly be dangerous. This warning may safely be ignored.

Take a goodly amount of time before starting trying to understand what HAS been provided in the program. Note that if you use the default arguments of the initializer of recognition\_sequence=r'GAATTC', bond\_position=1, this will correspond to the restriction enzyme EcoRI as shown above. *However, in our actual analysis, we are going to use the restriction enzyme HindIII, which was isolated from the bacterium Haemophilus influenzae, and which has a recognition sequence 5' - A/A G-C-T-T – 3'* (*p.s.* don’t change the defaults, just make sure to call the initializer with the correct argument).

I have given you the skeleton, but much of the required functionality is missing in the provided program. But once you have filled in the blanks, your Python program must do the following:

1. Produce a list of the positions of every cut site in the input file. Since restriction enzyme recognition sites are very predictable, might I suggest that this is a perfect job for (pretty simple!) regular expressions?

2. Infer from the cut positions the DNA fragments that would be produced by “digestion” of that DNA sequence with the restriction enzyme (*i.e.* the fragments that would be produced if every site was cut).

3. Produce a list of Fragment objects (I have, very kindly indeed, provided a Fragment class for you in the code) that describe each fragment, including:

a. fragment number

b. fragment length

c. start position in original sequence (remember biological sequences begin at residue 1, not residue 0)

d. end position in original sequence

e. the actual DNA sequence (using the usual 5’->3’ single strand convention)

4. The fragments above should be output to an output file that will be in FASTA format (*i.e.* each annotation line is preceded with a ‘>’ character followed by the annotation). The annotation line should consist of the fragment number, followed by the beginning and ending position. The data line(s) will consist of the actual DNA sequence corresponding to that fragment

5. There should be a method that calculates the number and average length of the fragments that were found. This method should ALSO calculate the number of EXPECTED fragments and the average length of the expected fragments, based on both the composition of the input sequence and on the composition of the recognition site.

If you calculate both of these it should report the **ratio** of the two as well. The concept of a ratio of actually observed vs. expected (given some model) is something that will reoccur frequently during the course. What is the significance if there is a deviation from the expectation under some model or set of assumptions?

**2A.** Your code will be turned in as

**YOURNAME\_Restriction\_Digestion.py**

**2B.** How many fragments did you actually observe resulting from your *in silico* digestion of the mystery genome with HindIII?? How does this number strike you? High? Low? About right? How would you even calibrate your thinking about how many fragments should be produced? Explain any basic intuition you might have.

**2C.** What was the average length of the observed fragments? Again, explain your intuition about this number?

**2D.** How many fragments did you EXPECT to find? Is the number of fragments that you actually found higher or lower than the number you expected to find in the input sequence? Explain your reasoning.

**2E.** What was the EXPECTED average size of the fragments that you found? How did the number found correspond to what you expected? Again, explain your reasoning.

**2F.** Can you identify the mystery genome? If so, what is it? Describe the approach you took.

**2G.** If we pause to think carefully about the process that we have carried out and the results we have obtained, the identity of the mystery genome actually poses a significant conundrum. Put on your biologist’s cap for a moment, and describe for me the nature of this conundrum.

**2H.** How is this conundrum resolved? You may need to do some further reading on restriction enzymes and their associated systems in prokaryotes. Consider not only

what restriction enzymes do, but what they are FOR in the natural world, and the selective pressures that have led to their evolution.