# **CEN0336 - Introdução a Programação de Computadores Aplicada a Ciências Biológicas**

# **Prova II**

Nome:

No USP:

## **Instruções Gerais**

Vocês terão desde o dia 3 de dezembro as 15:00 e até o dia 15 de dezembro as 23:59 para resolver as atividades da prova.

São 4 perguntas/atividades, cada uma precisa de uma resposta neste arquivo de word, que tem que ser carregado no sistema e-disciplinas até o dia e horário limite. Além disso algumas perguntas precisam que vocês desenvolvam scripts e carregam num repositório github especifico para a prova (indicado na pergunta 1). No dia 16 de dezembro as 00:01 eu descarregarei todos os repositórios para avaliar seus scripts. Os critérios de avaliação dos scripts são:

Eles têm que rodar sem erros de sintaxe (30%)

Eles têm que ser logicamente corretos (40%)

Eles têm que ter comentários explicando o que é feito em cada passo (20%)

Eles têm que ter controle da entrada dos usuários (10%)

## **Perguntas/Atividades**

1. Crie um repositório no github para esta prova. Escreva o endereço completo do seu repositório github como resposta desta pergunta.

Lembre-se que o repositório tem que ter o nome segundo o padrão:

P2\_NNNNNNN-CEN036

Onde NNNNNNN é seu número USP.

A resposta tem que ter o endereço completo, iniciando com https://"

O professor irá clonar seu repositório, com as respostas para os outros pontos da prova.

<https://github.com/Garibodi/P2_10319115-CEN036.git>

1. Crie um script de python com o nome script\_getORF.py que realize as seguintes operações:
2. Ler um arquivo multifasta da linha de comandos, usando o modulo sys.
3. Para cada registro fasta identificar o Open Reading Frame (Fase de Leitura Aberta: <https://pt.wikipedia.org/wiki/Fase_de_leitura_aberta> ou <https://en.wikipedia.org/wiki/Open_reading_frame>) mais comprido dentre as 6 fases de leitura possíveis. Os ORFs neste caso não têm introns. Repare que uma fase de leitura aberta tem que iniciar com um codon de início da tradução e terminar com um códon de terminação da tradução (<https://pt.wikipedia.org/wiki/C%C3%B3digo_gen%C3%A9tico>), é seu comprimento tem que ser múltiplo de três (3)
4. Daquele ORF mais comprido identificado, gerar o peptídeo codificado.
5. Imprimir o ORF mais comprido no arquivo ORF.fna. O formato da sequência tem que ser fasta, e seu identificador, tem que ser o identificador original, adicionando o string: “\_frameX\_START\_END”, onde X é a Fase de leitura (1,2,3,4,5 ou 6), e START e END são as coordenadas na sequencia
6. Imprimir o peptídeo correspondente ao ORF mais comprido no arquivo ORF.faa. O formato da sequência tem que ser fasta, e seu identificador, tem que ser o identificador original, adicionando o string: “\_frameX\_START\_END”, onde X é a Fase de leitura (1,2,3,4,5 ou 6), e START e END são as coordenadas na sequencia
7. Não esquece que tem que processar todos os registros do arquivo fasta original, i.e., tem que ter o mesmo número de entradas nos arquivos ORF.faa, ORF.fna que no arquivo original.
8. Carregue seu script, e os arquivos ORF.faa e ORF.fna no seu repositório.
9. Calcule o md5sum do seu script e coloque esse número como resposta desta pergunta.
10. Explique a diferença entre os dois sets:

**set1=set('accggttaata')  
set2={'accggttaata'}**

**Ambos são sets, porém a primeira linha *(set1=set('accggttaata')* se refere à um set construído com a função *“set constructor: set()”.* Utilizando este método com a sequência “accggttaata”, o set retornará os valores {'t', 'g', 'a', 'c'}, já que foi criado um set com esta string e cada letra assume uma posição no set, sem aceitar duplicatas, e em qualquer ordem já que o set não é ordenado nem indexado.**

**Já o segundo argumento *(set2={'accggttaata'})* cria um set adicionando a string *'accggttaata'* a um uma variável do tipo *set* com uma posição, a qual a sequência assume.**

**Quando rodamos o script com os seguintes argumentos para ambos, temos o seguinte:**

**Código:**

set1=set('accggttaata')

set2={'accggttaata'}

print(type(set1))

print(type(set2))

print(set1)

print(set2)

Output:

<class 'set'>

<class 'set'>

{'t', 'g', 'a', 'c'}

{'accggttaata'}

1. Descreva com sus próprias palavras o que o seguinte pseudocódigo faz. Faz uma avaliação desse pseudocódigo e relate quaisquer problemas que possa ter.

Defina TOTAL com zero

Defina CONTADOR NOTAS como zero

While CONTADOR NOTAS é menor ou igual a dez

      Entre (input) a seguinte nota

      Some a nota ao TOTAL

Defina a média da disciplina como o TOTAL dividido por dez (TOTAL/10)

Imprima na tela a média da disciplina

Implemente esse pseudocódigo num script de python e carregue ele no seu repositório da prova com o nome notas.py. Calcule o md5sum do seu script e coloque esse número também como resposta desta pergunta.

**Md5sum: b5a1e2c474d45720ed02892b3cf9165a**

**O pseudocódigo define uma variável para guardar a soma das notas e outra para ser uma variável contadora. Em seguida, insere um ciclo while em que o usuário é inquirido por uma nota 10 vezes, e essa nota é somada à variável total.**

**No caso, o pseudocódigo não avança pois não foi adicionado um argumento para adicionar uma unidade ao contador a cada ciclo do while. Portanto, o código perguntará indefinidamente notas ao usuário e o contador nunca chegará à 10.**

**Após, o pseudocódigo calcularia a média das notas utilizando o total/10 e imprimiria na tela a média da disciplina.**

**Outro problema com o código é que o while loop tem que ser <=9 ou <10, caso contrário assumirá 11 notas e não 10.**

**O pseudocódigo estaria correto se fosse escrito como:**

**Defina TOTAL como zero**

**Defina CONTADOR NOTAS como zero**

**Enquanto CONTADOR NOTAS é menor que dez**

**Defina NOTA como input: “Entre a seguinte nota”**

**Some NOTA ao TOTAL**

**Some 1 a CONTADOR**

**Defina a média da disciplina como o TOTAL dividido por dez (TOTAL/10)**

**Imprima na tela a média da disciplina**