B-Cell Epitope Prediction using Attention-Based LSTM

Group 7 Members:

- Donaire, Rudnick James
- Gonzales, Ryan Joseph
- Moncayo, Ethan Andrew
- Pajaro, Randall Joseph

Project Overview

B-Cells / B-Lymphocytes are a type of white blood cells that functions as a cell that produces antibody molecules which are responsible for humoral immune response (fluid-related substances).

Epitopes are regions in a protein / peptide that the antigen recognizes. These are areas in which the immune response of our body will recognize and are susceptible for antibodies to lock with (think of it as a lock and key process).

Epitope Prediction is a process in bioinformatics in which it identifies regions of a certain cell where antibodies can latch onto and neutralize its functions. This process can be used to develop epitope-based vaccine of various viruses such as SARS-CoV or H1V1.

The use of Attention-based LSTM to identify these epitope regions in a particular protein sequence requires the task to identify important points within the protein sequence alongside the chemical and structural features of that particular protein and peptide sequeunces.

This project is based on the study of Toshiaki, N. et al [1]

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sgt import SGT

import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Concatenate, LSTM, Dense, Activation, BatchN
from tensorflow.keras.callbacks import EarlyStopping

from attention import Attention

from sklearn.ensemble import ExtraTreesClassifier
```

```
from sklearn.decomposition import PCA

from sklearn.metrics import classification_report
```

Loading the Dataset

In [2]:

```
sars_csv = pd.read_csv('input_sars.csv')
bcell_csv = pd.read_csv('input_bcell.csv')

sars = sars_csv.copy()
b_cell = bcell_csv.copy()

df = pd.concat([sars,b_cell], ignore_index=True)
df.head()
```

Out[2]:

	parent_protein_id	protein_seq	start_position	end_position
0	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	1	17
1	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	1	15
2	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	2	10
3	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	6	20
4	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	9	25

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14907 entries, 0 to 14906
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   parent_protein_id    14907 non-null  object
 1   protein_seq          14907 non-null  object
 2   start_position       14907 non-null  int64
 3   end_position         14907 non-null  int64
 4   peptide_seq          14907 non-null  object
 5   chou_fasman          14907 non-null  float64
 6   emini                14907 non-null  float64
 7   kolaskar_tongaonkar  14907 non-null  float64
 8   parker               14907 non-null  float64
 9   isoelectric_point    14907 non-null  float64
 10  aromaticity          14907 non-null  float64
 11  hydrophobicity       14907 non-null  float64
 12  stability            14907 non-null  float64
 13  target               14907 non-null  int64
dtypes: float64(8), int64(3), object(3)
memory usage: 1.6+ MB
```

In [4]:

```
df.describe()
```

Out[4]:

	start_position	end_position	chou_fasman	emini	kolaskar_tongaonkar	parker	isoe
	14907.000000	14907.000000	14907.000000	14907.000000	14907.000000	14907.000000	

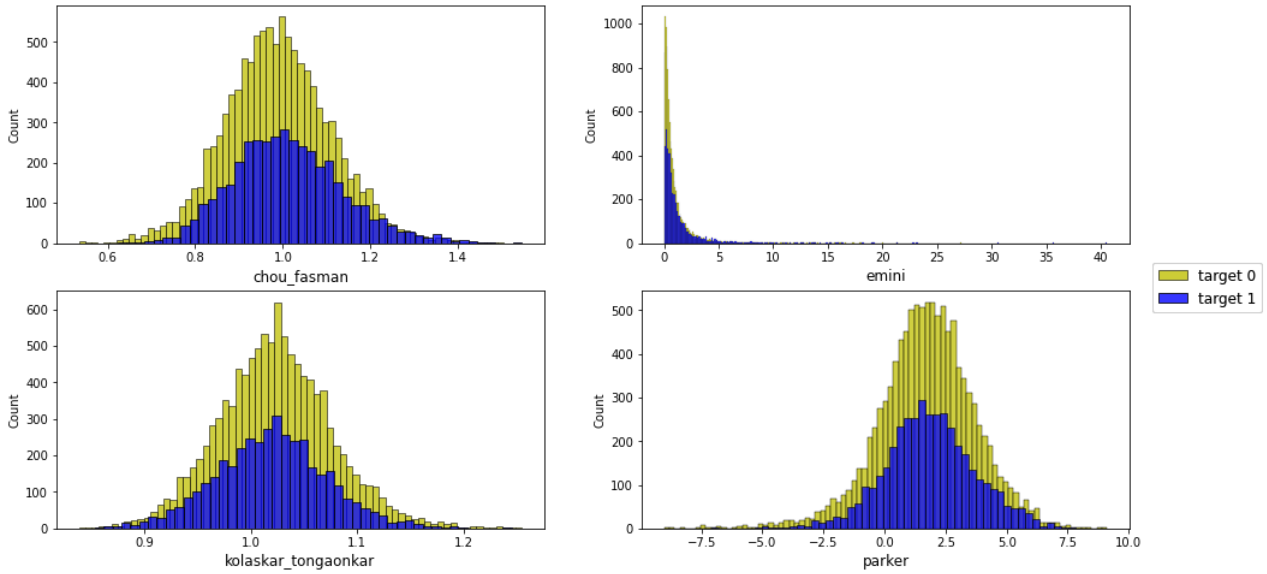
	start_position	end_position	chou_fasman	emini	kolaskar_tongaonkar	parker	iso
mean	308.845173	319.519420	0.994906	1.082811	1.021808	1.750098	
std	358.433563	358.647859	0.123656	1.826098	0.053430	1.954424	
min	1.000000	6.000000	0.534000	0.000000	0.838000	-9.029000	
25%	86.000000	96.000000	0.913000	0.244000	0.987000	0.600000	
50%	197.000000	208.000000	0.991000	0.551000	1.021000	1.775000	
75%	400.000000	411.000000	1.073000	1.208500	1.055000	2.960000	
max	3079.000000	3086.000000	1.546000	40.605000	1.255000	9.120000	

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
parent_protein_id      0
protein_seq            0
start_position         0
end_position           0
peptide_seq            0
chou_fasman            0
emini                  0
kolaskar_tongaonkar    0
parker                 0
isoelectric_point      0
aromaticity            0
hydrophobicity         0
stability              0
target                 0
dtype: int64
```

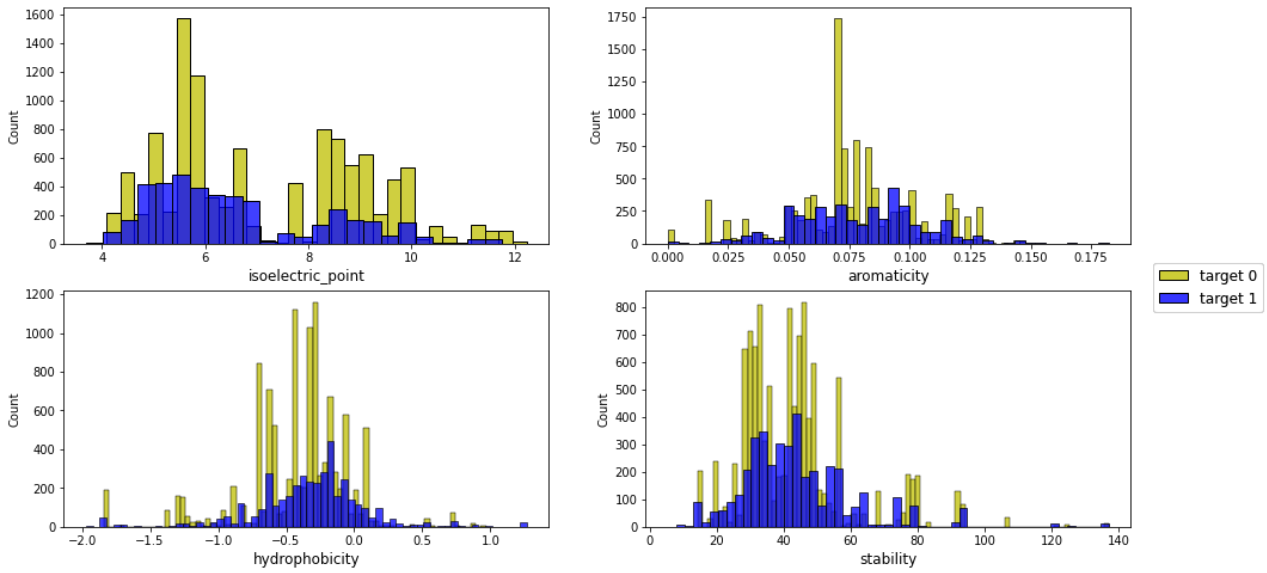
Exploratory Data Analysis

In [6]:
```
epitopes = df['target'].astype("bool").values
fig, ax = plt.subplots(2, 2, figsize=(16,8))

ax = [x for a in ax for x in a]

for i,name in enumerate(["chou_fasman","emini","kolaskar_tongaonkar","parker"]):
    value = df[name]
    sns.histplot(value[~epitopes],
                 ax = ax[i],
                 color = 'y')
    sns.histplot(value[epitopes],
                 ax = ax[i],
                 color = 'b')
    ax[i].set_xlabel(name,
                     fontsize=12)
    fig.legend(labels = ["target 0", "target 1"],
               loc = "right",
               fontsize=12)
```



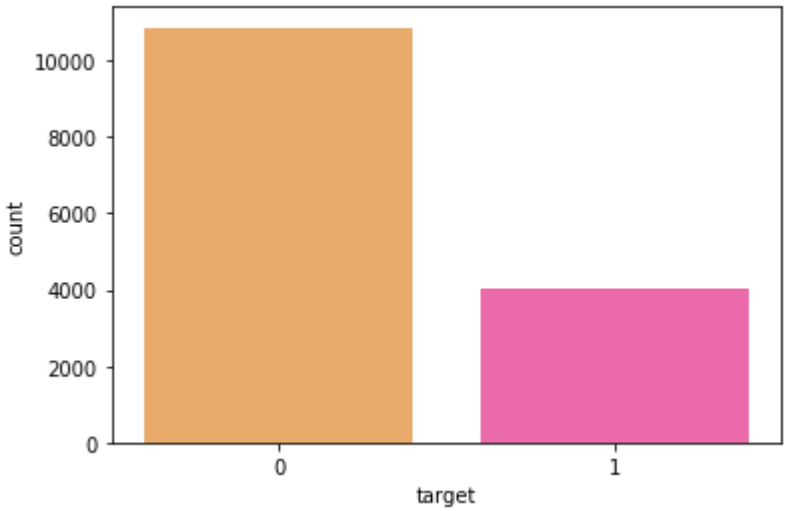
In [7]:

```
epitopes = df['target'].astype("bool").values
fig, ax = plt.subplots(2, 2, figsize=(16,8))

ax = [x for a in ax for x in a]

for i,name in enumerate(["isoelectric_point","aromaticity","hydrophobicity","stability"
    value = df[name]
    sns.histplot(value[~epitopes],
                 ax = ax[i],
                 color = 'y')
    sns.histplot(value[epitopes],
                 ax = ax[i],
                 color = 'b')
    ax[i].set_xlabel(name,
                     fontsize=12)
    fig.legend(labels = ["target 0", "target 1"],
               loc = "right",
               fontsize=12)
```



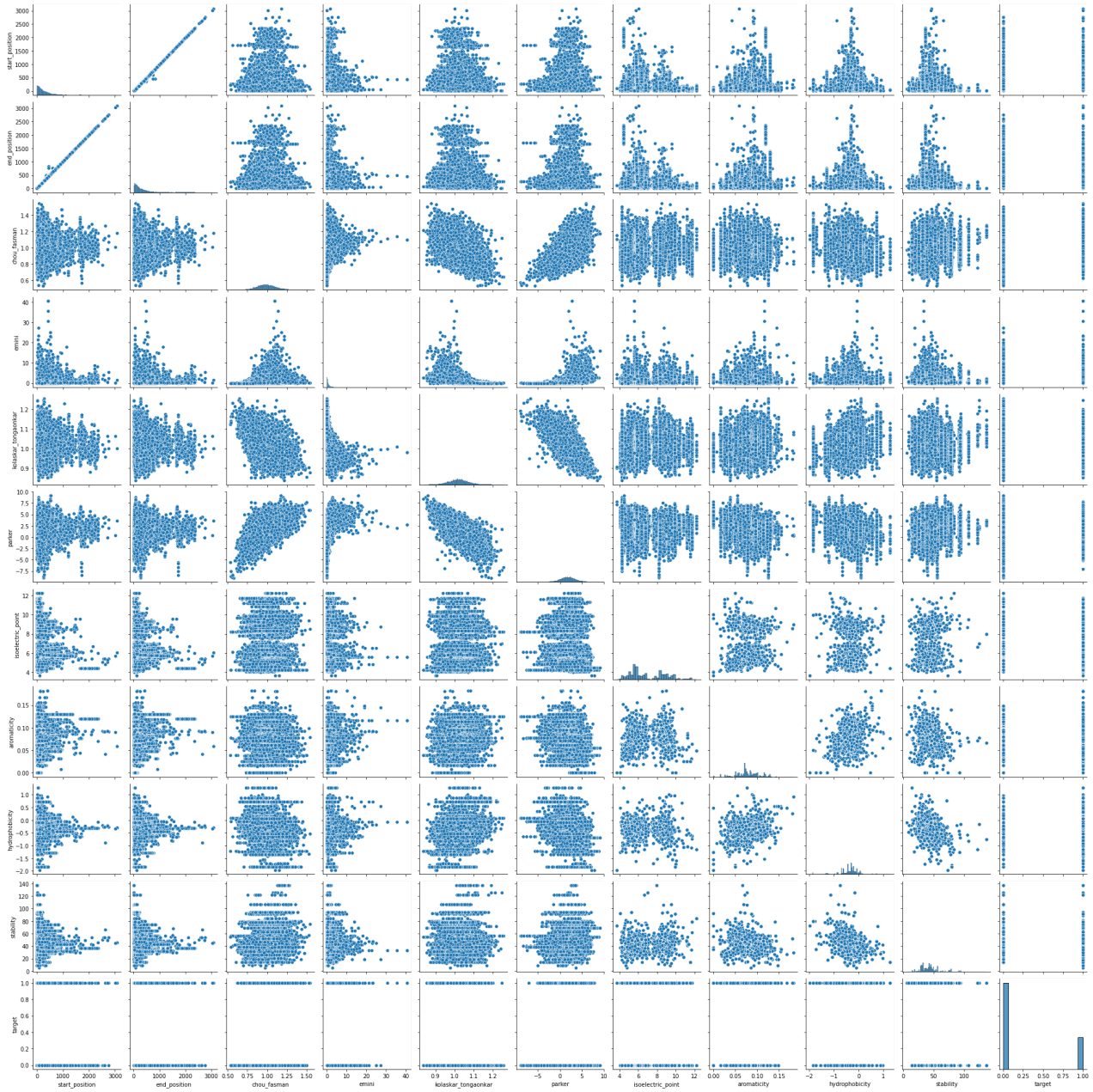
As what is shown in here, the distribution of the class per chemical features of a protein and the

In [8]:
```
sns.countplot(data = df,
              x = 'target',
              palette = 'spring_r')
plt.show()
```



The reason for the imbalance number of labels of epitope regions is due to the fact to the total number of proteins present in the dataset. Different proteins have different lengths that may affect the number of eptiope regions present in that sequence (hypothetically).

In [9]:
```
sns.pairplot(data = df)
plt.show()
```



This pairplot represents the correlation of each chemical and structural features of proteins and peptides in the dataset.

Model Creation

In [10]:
```
# simplifying the dataset

# protein sequence
sequence = df[['parent_protein_id', 'protein_seq', 'peptide_seq']].copy()

# features
features = df.drop(['protein_seq', 'peptide_seq'], axis = 1).copy()

# target
target = df[['parent_protein_id', 'target']].copy()
```

We will be dividng the dataset into portions: sequences of the proteins present in the dataset (for [illegible] features of proteins and peptides, and the target class

or the ground truth labels of epitope regions.

In [11]:
```
sequence.head()
```

Out[11]:

	parent_protein_id	protein_seq	peptide_seq
0	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	MFIFLLFLTLTSGSDLD
1	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	MFIFLLFLTLTSGSD
2	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	FIFLLFLTL
3	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	LFLTLTSGSDLDRCT
4	AAU93319	MFIFLLFLTLTSGSDLDRCTTFDDVQAPNYTQHTSSMRGVYYPDEI...	TLTSGSDLDRCTTFDDV

In [12]:
```
features.head()
```

Out[12]:

	parent_protein_id	start_position	end_position	chou_fasman	emini	kolaskar_tongaonkar	parker	isc
0	AAU93319	1	17	0.887	0.040	1.056	-2.159	
1	AAU93319	1	15	0.869	0.047	1.056	-2.500	
2	AAU93319	2	10	0.621	0.042	1.148	-7.467	
3	AAU93319	6	20	1.021	0.230	1.049	0.927	
4	AAU93319	9	25	1.089	0.627	1.015	3.165	

In [13]:
```
target.head()
```

Out[13]:

	parent_protein_id	target
0	AAU93319	0
1	AAU93319	0
2	AAU93319	0
3	AAU93319	0
4	AAU93319	0

Embedding of Protein Sequences

The embedding of protein sequences is important for this method. This method is similar to how embedding works in NLP. In this case, the embedding sequence will be based off on the amino acids present in the sequence. These amino acids can be seen in the image below:

Each letter present in the sequences represents the amino acid shown in the picture.

In [14]:

```
corpus = sequence.drop_duplicates(subset = ['parent_protein_id']).reset_index().drop('i
corpus = corpus[['parent_protein_id', 'protein_seq']].copy()
corpus['protein_seq'] = corpus['protein_seq'].map(list)
corpus.rename(columns = {'parent_protein_id': 'id', 'protein_seq': 'sequence'}, inplace

corpus
```

Out[14]:

	id	sequence
0	AAU93319	[M, F, I, F, L, L, F, L, T, L, T, S, G, S, D, ...
1	A2T3T0	[M, D, V, L, Y, S, L, S, K, T, L, K, D, A, R, ...
2	F0V2I4	[M, T, I, H, K, V, A, I, N, G, F, G, R, I, G, ...
3	O75508	[M, V, A, T, C, L, Q, V, V, G, F, V, T, S, F, ...
4	O84462	[M, T, N, S, I, S, G, Y, Q, P, T, V, T, T, S, ...
...	...	...
756	Q5F6I1	[M, T, K, Q, L, K, L, S, A, L, F, V, A, L, L, ...
757	Q7T9D9	[M, G, G, L, S, L, L, Q, L, P, R, D, K, F, R, ...
758	Q81871	[M, R, P, R, P, I, L, L, L, L, L, M, F, L, P, ...
759	Q91DE1	[M, D, R, G, T, R, R, I, W, V, S, Q, N, Q, G, ...
760	Q9QZS0	[M, H, S, K, T, A, P, R, F, L, V, F, L, L, L, ...

761 rows × 2 columns

In [15]:

```
sgt = SGT(kappa = 10,
          lengthsensitive = False)
embedding = sgt.fit_transform(corpus)
embedding.set_index('id', inplace = True)
embedding
```

Out[15]:

id	(A, A)	(A, C)	(A, D)	(A, E)	(A, F)	(A, G)	(A, H)	(A, I)	(A, K)	
AAU93319	0.206373	0.188073	0.197921	0.205757	0.188173	0.197756	0.076780	0.197342	0.182637	0.1
A2T3T0	0.083440	0.094843	0.236770	0.213567	0.085801	0.236914	0.241964	0.230914	0.091894	0.0
F0V2I4	0.240478	0.092911	0.210327	0.213199	0.240545	0.225934	0.232824	0.229365	0.213850	0.2
O75508	0.211541	0.229098	0.266633	0.036549	0.240204	0.250589	0.260957	0.231432	0.275536	0.2
O84462	0.224873	0.030106	0.186145	0.172689	0.186601	0.190224	0.206436	0.170815	0.186824	0.1
...	...	...	...	...	...	...	...	...	...	
				227828	0.097518	0.218990	0.031116	0.087373	0.244486	0.2

id	(A, A)	(A, C)	(A, D)	(A, E)	(A, F)	(A, G)	(A, H)	(A, I)	(A, K)	
Q7T9D9	0.207897	0.031050	0.194902	0.211794	0.242260	0.205042	0.081445	0.193283	0.199705	0.2
Q81871	0.210301	0.000528	0.202582	0.211396	0.191542	0.214938	0.213521	0.223836	0.010433	0.1
Q91DE1	0.200040	0.270063	0.181221	0.200017	0.212107	0.217553	0.198275	0.215787	0.207465	0.1
Q9QZS0	0.173567	0.068235	0.069460	0.069456	0.065576	0.195506	0.198585	0.205200	0.198557	0.1

761 rows × 400 columns

We will then apply PCA in order to reduce the dimensions of the embedded sequence into a 256-dimension vector.

In [16]:
```
pca = PCA(n_components = 256)
pca_components = pca.fit_transform(embedding)
```

In [17]:
```
pca_df = pd.DataFrame(pca_components,
                      columns = ['vector {0}'.format(i + 1) for i in range(256)],)
pca_df['parent_protein_id'] = corpus['id']
pca_cols = pca_df.columns.tolist()
pca_cols = pca_cols[-1:] + pca_cols[:-1]
pca_df = pca_df[pca_cols]
pca_df
```

Out[17]:

	parent_protein_id	vector 1	vector 2	vector 3	vector 4	vector 5	vector 6	vector 7	vect
0	AAU93319	-0.857650	0.224306	-0.168453	0.007793	-0.073999	-0.027826	-0.023339	0.00
1	A2T3T0	-0.217129	-0.305087	0.132407	0.237793	0.038140	-0.289895	-0.051760	0.05
2	F0V2I4	0.056037	-0.195550	-0.121842	0.208691	-0.401362	0.049826	0.204539	-0.09
3	O75508	0.497248	0.516343	0.186537	0.421227	-0.169212	-0.204186	0.072387	-0.04
4	O84462	-0.126953	-0.325322	-0.042521	0.052661	0.093575	0.003862	0.262025	-0.27
...	...	...	...	...	...	...	...	...	
756	Q5F6I1	0.429767	-0.230684	-0.026051	-0.134467	-0.198217	0.060654	0.084130	-0.05
757	Q7T9D9	-0.616245	0.141540	-0.013215	0.077081	-0.110705	0.140049	0.107811	-0.14
758	Q81871	-0.449741	-0.075700	0.393529	0.182626	0.054733	-0.022799	-0.051002	0.06
759	Q91DE1	-0.496172	-0.408107	0.172195	-0.027547	0.086922	-0.033614	0.000653	0.25
760	Q9QZS0	-0.232793	0.343720	-0.001349	-0.077699	-0.048350	-0.086221	0.174852	-0.00

In [18]:
```
merged = pd.merge(features, pca_df, how = 'inner', on = 'parent_protein_id')

# separating the dataset to two inputs: vectors and features

# vectors
columns = list(features.columns)
vectors_input = merged.drop(columns, axis = 1)

# features
features_input = merged[columns].copy()
features_input.drop('target', axis = 1, inplace = True)
```

We will scale the chemical and structural features of the proteins via StandardScaler

In [19]:
```
# scaling features_input values
scaler = StandardScaler()

scaling = features_input.drop(['parent_protein_id', 'start_position', 'end_position'],

scaled = scaler.fit_transform(scaling)

x = pd.DataFrame(scaled)
x.insert(0, 'start_position', features_input['start_position'])
x.insert(1, 'end_position', features['end_position'])

rename = list(features.columns)[2:]

for i in range(8):
    x.rename(columns = {i: rename[i]}, inplace = True)

y = merged['target']
```

This will be the model that is going to be used for the Attention-based LSTM architecture. It will consist of:

- an LSTM layer
- an Attention layer
- concatenated to a FCL that accepts the vectors of protein sequences and the chemical and structural features of the protein

In [20]:
```
# model architecture

# input layer
vector_input = Input((256, 1))
feature_input = Input((10,))

# lstm layer
lstm_layer_1 = LSTM(128, return_sequences = True)(vector_input)
lstm_dropout = Dropout(0.6)(lstm_layer_1)
lstm_attention = Attention(32)(lstm_dropout)

# fork layer
                                e_input, lstm_attention])
```

```
# fully-connected layer
dense_1 = Dense(200, kernel_initializer = 'normal', activation = 'relu')(concat_layer)
batch_normal_1 = BatchNormalization(momentum = 0.6)(dense_1)
dropout_1 = Dropout(0.3)(batch_normal_1)
dense_2 = Dense(100, kernel_initializer = 'uniform', activation = 'relu')(dropout_1)
batch_normal_2 = BatchNormalization(momentum = 0.6)(dense_2)
dropout_2 = Dropout(0.3)(batch_normal_2)
dense_3 = Dense(40, kernel_initializer = 'uniform', activation = 'relu')(dropout_2)
batch_normal_3 = BatchNormalization(momentum = 0.6)(dense_3)
droput_3 = Dropout(0.3)(batch_normal_3)
output = Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid')(droput_3)

# defining model
model = Model(inputs = [vector_input, feature_input], outputs = output)
model.compile(loss = "binary_crossentropy", optimizer = "adam", metrics = ['accuracy'])
```

In [21]:

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_1 (InputLayer)            [(None, 256, 1)]     0
__________________________________________________________________________________________
lstm (LSTM)                     (None, 256, 128)     66560       input_1[0][0]
__________________________________________________________________________________________
dropout (Dropout)               (None, 256, 128)     0           lstm[0][0]
__________________________________________________________________________________________
last_hidden_state (Lambda)      (None, 128)          0           dropout[0][0]
__________________________________________________________________________________________
attention_score_vec (Dense)     (None, 256, 128)     16384       dropout[0][0]
__________________________________________________________________________________________
attention_score (Dot)           (None, 256)          0           last_hidden_state[0][0]
                                                                 attention_score_vec[0]
[0]
__________________________________________________________________________________________
attention_weight (Activation)   (None, 256)          0           attention_score[0][0]
__________________________________________________________________________________________
context_vector (Dot)            (None, 128)          0           dropout[0][0]
                                                                 attention_weight[0][0]
__________________________________________________________________________________________
attention_output (Concatenate)  (None, 256)          0           context_vector[0][0]
                                                                 last_hidden_state[0][0]
__________________________________________________________________________________________
input_2 (InputLayer)            [(None, 10)]         0
__________________________________________________________________________________________
attention_vector (Dense)        (None, 128)          32768       attention_output[0][0]
__________________________________________________________________________________________
concatenate (Concatenate)       (None, 138)          0           input_2[0][0]
```

```
                                                                 attention_vector[0][0]
__________________________________________________________________________________________
__________
dense (Dense)                   (None, 200)          27800       concatenate[0][0]
__________________________________________________________________________________________
__________
batch_normalization (BatchNorma (None, 200)          800         dense[0][0]
__________________________________________________________________________________________
__________
dropout_1 (Dropout)             (None, 200)          0           batch_normalization[0]
[0]
__________________________________________________________________________________________
__________
dense_1 (Dense)                 (None, 100)          20100       dropout_1[0][0]
__________________________________________________________________________________________
__________
batch_normalization_1 (BatchNor (None, 100)          400         dense_1[0][0]
__________________________________________________________________________________________
__________
dropout_2 (Dropout)             (None, 100)          0           batch_normalization_1
[0][0]
__________________________________________________________________________________________
__________
dense_2 (Dense)                 (None, 40)           4040        dropout_2[0][0]
__________________________________________________________________________________________
__________
batch_normalization_2 (BatchNor (None, 40)           160         dense_2[0][0]
__________________________________________________________________________________________
__________
dropout_3 (Dropout)             (None, 40)           0           batch_normalization_2
[0][0]
__________________________________________________________________________________________
__________
dense_3 (Dense)                 (None, 1)            41          dropout_3[0][0]
==========================================================================================
==========
Total params: 169,053
Trainable params: 168,373
Non-trainable params: 680
__________________________________________________________________________________________
__________
```

Model Testing

In [22]:

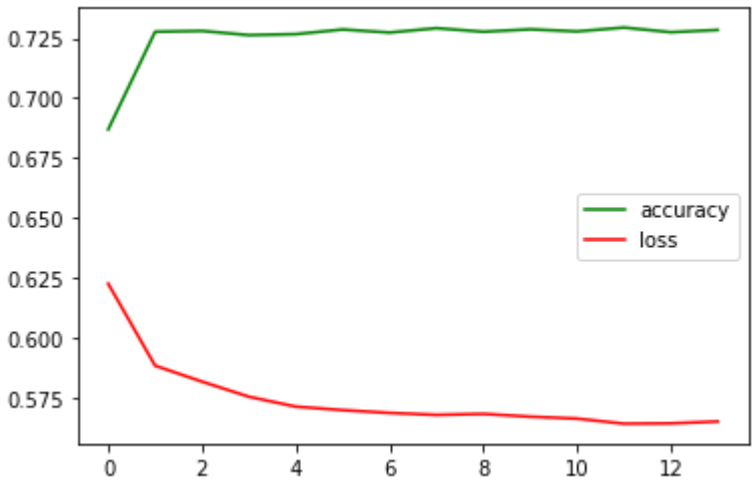
```
early_stop = EarlyStopping(monitor = 'loss',
                           patience = 2,
                           verbose = 1)

hist = model.fit([vectors_input, x], y,
                 epochs = 100,
                 batch_size = 32,
                 callbacks = early_stop)
```

```
Epoch 1/100
466/466 [==============================] - 80s 157ms/step - loss: 0.6541 - accuracy: 0.6
306
Epoch 2/100
466/466 [==============================] - 73s 156ms/step - loss: 0.5901 - accuracy: 0.7
283
Epoch 3/100
466/466 [==============================] - 75s 162ms/step - loss: 0.5847 - accuracy: 0.7
271
```

```
Epoch 4/100
466/466 [==============================] - 73s 156ms/step - loss: 0.5730 - accuracy: 0.7
293
Epoch 5/100
466/466 [==============================] - 74s 158ms/step - loss: 0.5746 - accuracy: 0.7
266
Epoch 6/100
466/466 [==============================] - 73s 157ms/step - loss: 0.5688 - accuracy: 0.7
297
Epoch 7/100
466/466 [==============================] - 81s 174ms/step - loss: 0.5640 - accuracy: 0.7
315
Epoch 8/100
466/466 [==============================] - 79s 169ms/step - loss: 0.5668 - accuracy: 0.7
318
Epoch 9/100
466/466 [==============================] - 79s 169ms/step - loss: 0.5729 - accuracy: 0.7
231
Epoch 10/100
466/466 [==============================] - 79s 170ms/step - loss: 0.5666 - accuracy: 0.7
291
Epoch 11/100
466/466 [==============================] - 79s 169ms/step - loss: 0.5644 - accuracy: 0.7
289
Epoch 12/100
466/466 [==============================] - 80s 171ms/step - loss: 0.5678 - accuracy: 0.7
268
Epoch 13/100
466/466 [==============================] - 79s 169ms/step - loss: 0.5635 - accuracy: 0.7
311
Epoch 14/100
466/466 [==============================] - 78s 168ms/step - loss: 0.5679 - accuracy: 0.7
242
Epoch 00014: early stopping
```

In [23]:

```
plt.plot(hist.history['accuracy'],
         label = 'accuracy',
         color = 'green')
plt.plot(hist.history['loss'],
         label = 'loss',
         color = 'red')
plt.legend()
plt.show()
```



d by the model is around 72%, which is somewhat close

to the study that this project was based on.

Model Predictions

We will be using the model to a dataset that contains a protein sequence of the Covid with no particular labels to it.

In [24]:
```
covid_data = pd.read_csv('input_covid.csv')
covid_data
```

Out[24]:

	parent_protein_id	protein_seq	start_position	end_po
0	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1	
1	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	2	
2	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	3	
3	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	4	
4	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	5	
...	...	...	...	
20307	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1258	
20308	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1259	
20309	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1260	
20310	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1261	
20311	6VYB_A	MGILPSPGMPALLSLVSLLSVLLMGCVAETGTQCVNLTTRTQLPPA...	1262	

20312 rows × 13 columns

In [25]:
```
corpus_copy = corpus.copy()

sequence_pred = covid_data[['parent_protein_id', 'protein_seq', 'peptide_seq']].copy()

corpus_prediction = sequence_pred.drop_duplicates(subset = ['parent_protein_id']).reset
corpus_prediction = corpus_prediction[['parent_protein_id', 'protein_seq']].copy()
corpus_prediction['protein_seq'] = corpus_prediction['protein_seq'].map(list)
corpus_prediction.rename(columns = {'parent_protein_id': 'id', 'protein_seq': 'sequence

corpus_combine = pd.concat([corpus_copy, corpus_prediction], ignore_index = True)
corpus_combine
```

Out[25]:

	id	sequence
0	AAU93319	[M, F, I, F, L, L, F, L, T, L, T, S, G, S, D, ...
1	A2T3T0	[M, D, V, L, Y, S, L, S, K, T, L, K, D, A, R, ...
2	F0V2I4	[M, T, I, H, K, V, A, I, N, G, F, G, R, I, G, ...
3	Q75F08	[M, V, A, T, G, L, Q, V, V, G, F, V, T, S, F, ...

	id	sequence
4	O84462	[M, T, N, S, I, S, G, Y, Q, P, T, V, T, T, S, ...
...	...	...
757	Q7T9D9	[M, G, G, L, S, L, L, Q, L, P, R, D, K, F, R, ...
758	Q81871	[M, R, P, R, P, I, L, L, L, L, L, M, F, L, P, ...
759	Q91DE1	[M, D, R, G, T, R, R, I, W, V, S, Q, N, Q, G, ...
760	Q9QZS0	[M, H, S, K, T, A, P, R, F, L, V, F, L, L, L, ...
761	6VYB_A	[M, G, I, L, P, S, P, G, M, P, A, L, L, S, L, ...

762 rows × 2 columns

In [26]:
```
embedding_pred = sgt.fit_transform(corpus_combine)
embedding_pred.set_index('id', inplace = True)
embedding_pred
```

Out[26]:

id	(A, A)	(A, C)	(A, D)	(A, E)	(A, F)	(A, G)	(A, H)	(A, I)	(A, K)	
AAU93319	0.206373	0.188073	0.197921	0.205757	0.188173	0.197756	0.076780	0.197342	0.182637	0.1
A2T3T0	0.083440	0.094843	0.236770	0.213567	0.085801	0.236914	0.241964	0.230914	0.091894	0.0
F0V2I4	0.240478	0.092911	0.210327	0.213199	0.240545	0.225934	0.232824	0.229365	0.213850	0.2
O75508	0.211541	0.229098	0.266633	0.036549	0.240204	0.250589	0.260957	0.231432	0.275536	0.2
O84462	0.224873	0.030106	0.186145	0.172689	0.186601	0.190224	0.206436	0.170815	0.186824	0.1
...	...	...	...	...	...	...	...	...	...	
Q7T9D9	0.207897	0.031050	0.194902	0.211794	0.242260	0.205042	0.081445	0.193283	0.199705	0.2
Q81871	0.210301	0.000528	0.202582	0.211396	0.191542	0.214938	0.213521	0.223836	0.010433	0.1
Q91DE1	0.200040	0.270063	0.181221	0.200017	0.212107	0.217553	0.198275	0.215787	0.207465	0.1
Q9QZS0	0.173567	0.068235	0.069460	0.069456	0.065576	0.195506	0.198585	0.205200	0.198557	0.1
6VYB_A	0.196096	0.072897	0.201719	0.200558	0.069365	0.199422	0.179675	0.198010	0.170752	0.2

762 rows × 400 columns

In [27]:
```
pca_two = PCA(n_components = 256)
pca_components_two = pca_two.fit_transform(embedding_pred)
```

In [28]:
```
pca_df_two = pd.DataFrame(pca_components_two,
                          columns = ['vector {0}'.format(i + 1) for i in range(256)],)
pca_df_two['parent_protein_id'] = corpus_combine['id']
pca_cols_two = pca_df_two.columns.tolist()
pca_cols_two = pca_cols_two[-1:] + pca_cols_two[:-1]
vectors_test = pca_df_two[pca_cols_two]
vectors_test
```

Out[28]:

	parent_protein_id	vector 1	vector 2	vector 3	vector 4	vector 5	vector 6	vector 7	vect
0	AAU93319	-0.857394	0.224392	-0.168704	0.007802	-0.074178	-0.027566	-0.022688	0.00
1	A2T3T0	-0.216444	-0.305076	0.131948	0.237808	0.037897	-0.289526	-0.050432	0.05
2	F0V2I4	0.056537	-0.195339	-0.122498	0.208714	-0.401636	0.050397	0.205174	-0.09
3	O75508	0.498165	0.516291	0.186693	0.421231	-0.169205	-0.204092	0.072657	-0.04
4	O84462	-0.126038	-0.325361	-0.042634	0.052664	0.093481	0.004306	0.262125	-0.27
...	...	...	...	...	...	...	...	...	
757	Q7T9D9	-0.615274	0.141318	-0.012800	0.077078	-0.110638	0.140102	0.107496	-0.14
758	Q81871	-0.448200	-0.076245	0.394367	0.182613	0.054974	-0.023162	-0.051551	0.06
759	Q91DE1	-0.494827	-0.408481	0.172668	-0.027557	0.087076	-0.033785	0.000366	0.25
760	Q9QZS0	-0.231725	0.343501	-0.000753	-0.077711	-0.048164	-0.086266	0.174309	-0.00
761	6VYB_A	-0.751478	0.072437	-0.128949	0.002420	-0.034391	0.033462	0.060117	-0.00

762 rows × 257 columns

In [29]:
```
covid_features = covid_data.drop(['protein_seq', 'peptide_seq'], axis = 1).copy()

merged_input = pd.merge(vectors_test, covid_features, how = 'inner', on = 'parent_prote

covid_vectors = merged_input[vectors_test.columns.tolist()].copy()
covid_vectors.drop('parent_protein_id', axis = 1, inplace = True)

covid_features = merged_input[covid_features.columns.tolist()].copy()
covid_features.drop('parent_protein_id', axis = 1, inplace = True)
```

In [30]:
```
predictions = model.predict([covid_vectors, covid_features])
```

In [31]:
```
results = [1 if value > 0.5 else 0 for value in predictions]
```

In [32]:
```
                                                    eturn_counts = True)
```

In [33]:
```
covid_results = covid_data.copy()
covid_results['predictions'] = results

print('Peptide Sequences of Covid Strain in identifying locations of interest:\n')
for peptide in covid_results[covid_results['predictions'] == 1]['peptide_seq']:
    print('\t-{0}'.format(peptide))
```

```
Peptide Sequences of Covid Strain in identifying locations of interest:

	-MGILP
	-GILPS
	-ILPSP
	-LPSPG
	-PSPGM
	-SPGMP
	-PGMPA
	-GMPAL
	-MPALL
	-PALLS
	-ALLSL
	-LLSLV
	-LSLVS
	-SLVSL
	-LVSLL
	-VSLLS
	-SLLSV
	-LLSVL
	-LSVLL
	-SVLLM
	-VLLMG
	-LLMGC
	-LMGCV
	-MGCVA
	-GCVAE
	-CVAET
	-VAETG
	-AETGT
	-ETGTQ
	-TGTQC
	-GTQCV
	-TQCVN
	-QCVNL
	-CVNLT
	-VNLTT
	-NLTTR
	-LTTRT
	-TTRTQ
	-TRTQL
	-RTQLP
	-TQLPP
	-QLPPA
	-LPPAY
	-PPAYT
	-PAYTN
	-AYTNS
	-YTNSF
	-TNSFT
	-NSFTR
	-SFTRG
	-FTRGV
	-TRGVY
	-RGVYY
```

```
-YYPDK
-YPDKV
-PDKVF
-DKVFR
-KVFRS
-VFRSS
-FRSSV
-RSSVL
-SSVLH
-SVLHS
-VLHST
-LFLPF
-MGILPS
-GILPSP
-ILPSPG
-LPSPGM
-PSPGMP
-SPGMPA
-PGMPAL
-GMPALL
-MPALLS
-PALLSL
-ALLSLV
-LLSLVS
-LSLVSL
-SLVSLL
-LVSLLS
-VSLLSV
-SLLSVL
-LLSVLL
-LSVLLM
-SVLLMG
-VLLMGC
-LLMGCV
-LMGCVA
-MGCVAE
-GCVAET
-CVAETG
-VAETGT
-AETGTQ
-ETGTQC
-TGTQCV
-GTQCVN
-TQCVNL
-QCVNLT
-CVNLTT
-VNLTTR
-NLTTRT
-LTTRTQ
-TTRTQL
-TRTQLP
-RTQLPP
-TQLPPA
-QLPPAY
-LPPAYT
-PPAYTN
-PAYTNS
-AYTNSF
-YTNSFT
-TNSFTR
-NSFTRG
-SFTRGV
-FTRGVY
```



```
-RGVYYP
```

```
-GVYYPD
-VYYPDK
-YYPDKV
-YPDKVF
-PDKVFR
-DKVFRS
-KVFRSS
-VFRSSV
-FRSSVL
-RSSVLH
-SSVLHS
-LFLPFF
-MGILPSP
-GILPSPG
-ILPSPGM
-LPSPGMP
-PSPGMPA
-SPGMPAL
-PGMPALL
-GMPALLS
-MPALLSL
-PALLSLV
-ALLSLVS
-LLSLVSL
-LSLVSLL
-SLVSLLS
-LVSLLSV
-VSLLSVL
-SLLSVLL
-LLSVLLM
-LSVLLMG
-SVLLMGC
-VLLMGCV
-LLMGCVA
-LMGCVAE
-MGCVAET
-GCVAETG
-CVAETGT
-VAETGTQ
-AETGTQC
-ETGTQCV
-TGTQCVN
-GTQCVNL
-TQCVNLT
-QCVNLTT
-CVNLTTR
-VNLTTRT
-NLTTRTQ
-LTTRTQL
-TTRTQLP
-TRTQLPP
-RTQLPPA
-TQLPPAY
-QLPPAYT
-LPPAYTN
-PPAYTNS
-PAYTNSF
-AYTNSFT
-YTNSFTR
-TNSFTRG
-NSFTRGV
-SFTRGVY
-FTRGVYY
```



```
-RGVYYPD
```

```
-GVYYPDK
-VYYPDKV
-YYPDKVF
-YPDKVFR
-PDKVFRS
-DKVFRSS
-KVFRSSV
-VFRSSVL
-FRSSVLH
-RSSVLHS
-MGILPSPG
-GILPSPGM
-ILPSPGMP
-LPSPGMPA
-PSPGMPAL
-SPGMPALL
-PGMPALLS
-GMPALLSL
-MPALLSLV
-PALLSLVS
-ALLSLVSL
-LLSLVSLL
-LSLVSLLS
-SLVSLLSV
-LVSLLSVL
-VSLLSVLL
-SLLSVLLM
-LLSVLLMG
-LSVLLMGC
-SVLLMGCV
-VLLMGCVA
-LLMGCVAE
-LMGCVAET
-MGCVAETG
-GCVAETGT
-CVAETGTQ
-VAETGTQC
-AETGTQCV
-ETGTQCVN
-TGTQCVNL
-GTQCVNLT
-TQCVNLTT
-QCVNLTTR
-CVNLTTRT
-VNLTTRTQ
-NLTTRTQL
-LTTRTQLP
-TTRTQLPP
-TRTQLPPA
-RTQLPPAY
-TQLPPAYT
-QLPPAYTN
-LPPAYTNS
-PPAYTNSF
-PAYTNSFT
-AYTNSFTR
-YTNSFTRG
-TNSFTRGV
-NSFTRGVY
-SFTRGVYY
-FTRGVYYP
-TRGVYYPD
-RGVYYPDK
```



```
-VYYPDKVF
```

```
-YYPDKVFR
-YPDKVFRS
-PDKVFRSS
-DKVFRSSV
-KVFRSSVL
-VFRSSVLH
-FRSSVLHS
-MGILPSPGM
-GILPSPGMP
-ILPSPGMPA
-LPSPGMPAL
-PSPGMPALL
-SPGMPALLS
-PGMPALLSL
-GMPALLSLV
-MPALLSLVS
-PALLSLVSL
-ALLSLVSLL
-LLSLVSLLS
-LSLVSLLSV
-SLVSLLSVL
-LVSLLSVLL
-VSLLSVLLM
-SLLSVLLMG
-LLSVLLMGC
-LSVLLMGCV
-SVLLMGCVA
-VLLMGCVAE
-LLMGCVAET
-LMGCVAETG
-MGCVAETGT
-GCVAETGTQ
-CVAETGTQC
-VAETGTQCV
-AETGTQCVN
-ETGTQCVNL
-TGTQCVNLT
-GTQCVNLTT
-TQCVNLTTR
-QCVNLTTRT
-CVNLTTRTQ
-VNLTTRTQL
-NLTTRTQLP
-LTTRTQLPP
-TTRTQLPPA
-TRTQLPPAY
-RTQLPPAYT
-TQLPPAYTN
-QLPPAYTNS
-LPPAYTNSF
-PPAYTNSFT
-PAYTNSFTR
-AYTNSFTRG
-YTNSFTRGV
-TNSFTRGVY
-NSFTRGVYY
-SFTRGVYYP
-FTRGVYYPD
-TRGVYYPDK
-RGVYYPDKV
-GVYYPDKVF
-VYYPDKVFR
-YYPDKVFRS
```



```
-PDKVFRSSV
```

```
-DKVFRSSVL
-KVFRSSVLH
-VFRSSVLHS
-FRSSVLHST
-MGILPSPGMP
-GILPSPGMPA
-ILPSPGMPAL
-LPSPGMPALL
-PSPGMPALLS
-SPGMPALLSL
-PGMPALLSLV
-GMPALLSLVS
-MPALLSLVSL
-PALLSLVSLL
-ALLSLVSLLS
-LLSLVSLLSV
-LSLVSLLSVL
-SLVSLLSVLL
-LVSLLSVLLM
-VSLLSVLLMG
-SLLSVLLMGC
-LLSVLLMGCV
-LSVLLMGCVA
-SVLLMGCVAE
-VLLMGCVAET
-LLMGCVAETG
-LMGCVAETGT
-MGCVAETGTQ
-GCVAETGTQC
-CVAETGTQCV
-VAETGTQCVN
-AETGTQCVNL
-ETGTQCVNLT
-TGTQCVNLTT
-GTQCVNLTTR
-TQCVNLTTRT
-QCVNLTTRTQ
-CVNLTTRTQL
-VNLTTRTQLP
-NLTTRTQLPP
-LTTRTQLPPA
-TTRTQLPPAY
-TRTQLPPAYT
-RTQLPPAYTN
-TQLPPAYTNS
-QLPPAYTNSF
-LPPAYTNSFT
-PPAYTNSFTR
-PAYTNSFTRG
-AYTNSFTRGV
-YTNSFTRGVY
-TNSFTRGVYY
-NSFTRGVYYP
-SFTRGVYYPD
-FTRGVYYPDK
-TRGVYYPDKV
-RGVYYPDKVF
-GVYYPDKVFR
-VYYPDKVFRS
-YYPDKVFRSS
-YPDKVFRSSV
-PDKVFRSSVL
-DKVFRSSVLH
```



```
-VFRSSVLHST
```

```
-MGILPSPGMPA
-GILPSPGMPAL
-ILPSPGMPALL
-LPSPGMPALLS
-PSPGMPALLSL
-SPGMPALLSLV
-PGMPALLSLVS
-GMPALLSLVSL
-MPALLSLVSLL
-PALLSLVSLLS
-ALLSLVSLLSV
-LLSLVSLLSVL
-LSLVSLLSVLL
-SLVSLLSVLLM
-LVSLLSVLLMG
-VSLLSVLLMGC
-SLLSVLLMGCV
-LLSVLLMGCVA
-LSVLLMGCVAE
-SVLLMGCVAET
-VLLMGCVAETG
-LLMGCVAETGT
-LMGCVAETGTQ
-MGCVAETGTQC
-GCVAETGTQCV
-CVAETGTQCVN
-VAETGTQCVNL
-AETGTQCVNLT
-ETGTQCVNLTT
-TGTQCVNLTTR
-GTQCVNLTTRT
-TQCVNLTTRTQ
-QCVNLTTRTQL
-CVNLTTRTQLP
-VNLTTRTQLPP
-NLTTRTQLPPA
-LTTRTQLPPAY
-TTRTQLPPAYT
-TRTQLPPAYTN
-RTQLPPAYTNS
-TQLPPAYTNSF
-QLPPAYTNSFT
-LPPAYTNSFTR
-PPAYTNSFTRG
-PAYTNSFTRGV
-AYTNSFTRGVY
-YTNSFTRGVYY
-TNSFTRGVYYP
-NSFTRGVYYPD
-SFTRGVYYPDK
-FTRGVYYPDKV
-TRGVYYPDKVF
-RGVYYPDKVFR
-GVYYPDKVFRS
-VYYPDKVFRSS
-YYPDKVFRSSV
-YPDKVFRSSVL
-PDKVFRSSVLH
-DKVFRSSVLHS
-KVFRSSVLHST
-MGILPSPGMPAL
-GILPSPGMPALL
-ILPSPGMPALLS
```



```
-PSPGMPALLSLV
```

```
-SPGMPALLSLVS
-PGMPALLSLVSL
-GMPALLSLVSLL
-MPALLSLVSLLS
-PALLSLVSLLSV
-ALLSLVSLLSVL
-LLSLVSLLSVLL
-LSLVSLLSVLLM
-SLVSLLSVLLMG
-LVSLLSVLLMGC
-VSLLSVLLMGCV
-SLLSVLLMGCVA
-LLSVLLMGCVAE
-LSVLLMGCVAET
-SVLLMGCVAETG
-VLLMGCVAETGT
-LLMGCVAETGTQ
-LMGCVAETGTQC
-MGCVAETGTQCV
-GCVAETGTQCVN
-CVAETGTQCVNL
-VAETGTQCVNLT
-AETGTQCVNLTT
-ETGTQCVNLTTR
-TGTQCVNLTTRT
-GTQCVNLTTRTQ
-TQCVNLTTRTQL
-QCVNLTTRTQLP
-CVNLTTRTQLPP
-VNLTTRTQLPPA
-NLTTRTQLPPAY
-LTTRTQLPPAYT
-TTRTQLPPAYTN
-TRTQLPPAYTNS
-RTQLPPAYTNSF
-TQLPPAYTNSFT
-QLPPAYTNSFTR
-LPPAYTNSFTRG
-PPAYTNSFTRGV
-PAYTNSFTRGVY
-AYTNSFTRGVYY
-YTNSFTRGVYYP
-TNSFTRGVYYPD
-NSFTRGVYYPDK
-SFTRGVYYPDKV
-FTRGVYYPDKVF
-TRGVYYPDKVFR
-RGVYYPDKVFRS
-GVYYPDKVFRSS
-VYYPDKVFRSSV
-YYPDKVFRSSVL
-YPDKVFRSSVLH
-PDKVFRSSVLHS
-DKVFRSSVLHST
-MGILPSPGMPALL
-GILPSPGMPALLS
-ILPSPGMPALLSL
-LPSPGMPALLSLV
-PSPGMPALLSLVS
-SPGMPALLSLVSL
-PGMPALLSLVSLL
-GMPALLSLVSLLS
-MPALLSLVSLLSV
```



```
-ALLSLVSLLSVLL
```

```
-LLSLVSLLSVLLM
-LSLVSLLSVLLMG
-SLVSLLSVLLMGC
-LVSLLSVLLMGCV
-VSLLSVLLMGCVA
-SLLSVLLMGCVAE
-LLSVLLMGCVAET
-LSVLLMGCVAETG
-SVLLMGCVAETGT
-VLLMGCVAETGTQ
-LLMGCVAETGTQC
-LMGCVAETGTQCV
-MGCVAETGTQCVN
-GCVAETGTQCVNL
-CVAETGTQCVNLT
-VAETGTQCVNLTT
-AETGTQCVNLTTR
-ETGTQCVNLTTRT
-TGTQCVNLTTRTQ
-GTQCVNLTTRTQL
-TQCVNLTTRTQLP
-QCVNLTTRTQLPP
-CVNLTTRTQLPPA
-VNLTTRTQLPPAY
-NLTTRTQLPPAYT
-LTTRTQLPPAYTN
-TTRTQLPPAYTNS
-TRTQLPPAYTNSF
-RTQLPPAYTNSFT
-TQLPPAYTNSFTR
-QLPPAYTNSFTRG
-LPPAYTNSFTRGV
-PPAYTNSFTRGVY
-PAYTNSFTRGVYY
-AYTNSFTRGVYYP
-YTNSFTRGVYYPD
-TNSFTRGVYYPDK
-NSFTRGVYYPDKV
-SFTRGVYYPDKVF
-FTRGVYYPDKVFR
-TRGVYYPDKVFRS
-RGVYYPDKVFRSS
-GVYYPDKVFRSSV
-VYYPDKVFRSSVL
-YYPDKVFRSSVLH
-YPDKVFRSSVLHS
-PDKVFRSSVLHST
-DKVFRSSVLHSTQ
-MGILPSPGMPALLS
-GILPSPGMPALLSL
-ILPSPGMPALLSLV
-LPSPGMPALLSLVS
-PSPGMPALLSLVSL
-SPGMPALLSLVSLL
-PGMPALLSLVSLLS
-GMPALLSLVSLLSV
-MPALLSLVSLLSVL
-PALLSLVSLLSVLL
-ALLSLVSLLSVLLM
-LLSLVSLLSVLLMG
-LSLVSLLSVLLMGC
-SLVSLLSVLLMGCV
-LVSLLSVLLMGCVA
```



```
-SLLSVLLMGCVAET
```

```
-LLSVLLMGCVAETG
-LSVLLMGCVAETGT
-SVLLMGCVAETGTQ
-VLLMGCVAETGTQC
-LLMGCVAETGTQCV
-LMGCVAETGTQCVN
-MGCVAETGTQCVNL
-GCVAETGTQCVNLT
-CVAETGTQCVNLTT
-VAETGTQCVNLTTR
-AETGTQCVNLTTRT
-ETGTQCVNLTTRTQ
-TGTQCVNLTTRTQL
-GTQCVNLTTRTQLP
-TQCVNLTTRTQLPP
-QCVNLTTRTQLPPA
-CVNLTTRTQLPPAY
-VNLTTRTQLPPAYT
-NLTTRTQLPPAYTN
-LTTRTQLPPAYTNS
-TTRTQLPPAYTNSF
-TRTQLPPAYTNSFT
-RTQLPPAYTNSFTR
-TQLPPAYTNSFTRG
-QLPPAYTNSFTRGV
-LPPAYTNSFTRGVY
-PPAYTNSFTRGVYY
-PAYTNSFTRGVYYP
-AYTNSFTRGVYYPD
-YTNSFTRGVYYPDK
-TNSFTRGVYYPDKV
-NSFTRGVYYPDKVF
-SFTRGVYYPDKVFR
-FTRGVYYPDKVFRS
-TRGVYYPDKVFRSS
-RGVYYPDKVFRSSV
-GVYYPDKVFRSSVL
-VYYPDKVFRSSVLH
-YYPDKVFRSSVLHS
-YPDKVFRSSVLHST
-PDKVFRSSVLHSTQ
-MGILPSPGMPALLSL
-GILPSPGMPALLSLV
-ILPSPGMPALLSLVS
-LPSPGMPALLSLVSL
-PSPGMPALLSLVSLL
-SPGMPALLSLVSLLS
-PGMPALLSLVSLLSV
-GMPALLSLVSLLSVL
-MPALLSLVSLLSVLL
-PALLSLVSLLSVLLM
-ALLSLVSLLSVLLMG
-LLSLVSLLSVLLMGC
-LSLVSLLSVLLMGCV
-SLVSLLSVLLMGCVA
-LVSLLSVLLMGCVAE
-VSLLSVLLMGCVAET
-SLLSVLLMGCVAETG
-LLSVLLMGCVAETGT
-LSVLLMGCVAETGTQ
-SVLLMGCVAETGTQC
-VLLMGCVAETGTQCV
-LLMGCVAETGTQCVN
```



```
-MGCVAETGTQCVNLT
```

```
-GCVAETGTQCVNLTT
-CVAETGTQCVNLTTR
-VAETGTQCVNLTTRT
-AETGTQCVNLTTRTQ
-ETGTQCVNLTTRTQL
-TGTQCVNLTTRTQLP
-GTQCVNLTTRTQLPP
-TQCVNLTTRTQLPPA
-QCVNLTTRTQLPPAY
-CVNLTTRTQLPPAYT
-VNLTTRTQLPPAYTN
-NLTTRTQLPPAYTNS
-LTTRTQLPPAYTNSF
-TTRTQLPPAYTNSFT
-TRTQLPPAYTNSFTR
-RTQLPPAYTNSFTRG
-TQLPPAYTNSFTRGV
-QLPPAYTNSFTRGVY
-LPPAYTNSFTRGVYY
-PPAYTNSFTRGVYYP
-PAYTNSFTRGVYYPD
-AYTNSFTRGVYYPDK
-YTNSFTRGVYYPDKV
-TNSFTRGVYYPDKVF
-NSFTRGVYYPDKVFR
-SFTRGVYYPDKVFRS
-FTRGVYYPDKVFRSS
-TRGVYYPDKVFRSSV
-RGVYYPDKVFRSSVL
-GVYYPDKVFRSSVLH
-VYYPDKVFRSSVLHS
-YYPDKVFRSSVLHST
-YPDKVFRSSVLHSTQ
-PDKVFRSSVLHSTQD
-MGILPSPGMPALLSLV
-GILPSPGMPALLSLVS
-ILPSPGMPALLSLVSL
-LPSPGMPALLSLVSLL
-PSPGMPALLSLVSLLS
-SPGMPALLSLVSLLSV
-PGMPALLSLVSLLSVL
-GMPALLSLVSLLSVLL
-MPALLSLVSLLSVLLM
-PALLSLVSLLSVLLMG
-ALLSLVSLLSVLLMGC
-LLSLVSLLSVLLMGCV
-LSLVSLLSVLLMGCVA
-SLVSLLSVLLMGCVAE
-LVSLLSVLLMGCVAET
-VSLLSVLLMGCVAETG
-SLLSVLLMGCVAETGT
-LLSVLLMGCVAETGTQ
-LSVLLMGCVAETGTQC
-SVLLMGCVAETGTQCV
-VLLMGCVAETGTQCVN
-LLMGCVAETGTQCVNL
-LMGCVAETGTQCVNLT
-MGCVAETGTQCVNLTT
-GCVAETGTQCVNLTTR
-CVAETGTQCVNLTTRT
-VAETGTQCVNLTTRTQ
-AETGTQCVNLTTRTQL
-ETGTQCVNLTTRTQLP
```



```
-GTQCVNLTTRTQLPPA
```

```
-TQCVNLTTRTQLPPAY
-QCVNLTTRTQLPPAYT
-CVNLTTRTQLPPAYTN
-VNLTTRTQLPPAYTNS
-NLTTRTQLPPAYTNSF
-LTTRTQLPPAYTNSFT
-TTRTQLPPAYTNSFTR
-TRTQLPPAYTNSFTRG
-RTQLPPAYTNSFTRGV
-TQLPPAYTNSFTRGVY
-QLPPAYTNSFTRGVYY
-LPPAYTNSFTRGVYYP
-PPAYTNSFTRGVYYPD
-PAYTNSFTRGVYYPDK
-AYTNSFTRGVYYPDKV
-YTNSFTRGVYYPDKVF
-TNSFTRGVYYPDKVFR
-NSFTRGVYYPDKVFRS
-SFTRGVYYPDKVFRSS
-FTRGVYYPDKVFRSSV
-TRGVYYPDKVFRSSVL
-RGVYYPDKVFRSSVLH
-GVYYPDKVFRSSVLHS
-VYYPDKVFRSSVLHST
-YYPDKVFRSSVLHSTQ
-YPDKVFRSSVLHSTQD
-MGILPSPGMPALLSLVS
-GILPSPGMPALLSLVSL
-ILPSPGMPALLSLVSLL
-LPSPGMPALLSLVSLLS
-PSPGMPALLSLVSLLSV
-SPGMPALLSLVSLLSVL
-PGMPALLSLVSLLSVLL
-GMPALLSLVSLLSVLLM
-MPALLSLVSLLSVLLMG
-PALLSLVSLLSVLLMGC
-ALLSLVSLLSVLLMGCV
-LLSLVSLLSVLLMGCVA
-LSLVSLLSVLLMGCVAE
-SLVSLLSVLLMGCVAET
-LVSLLSVLLMGCVAETG
-VSLLSVLLMGCVAETGT
-SLLSVLLMGCVAETGTQ
-LLSVLLMGCVAETGTQC
-LSVLLMGCVAETGTQCV
-SVLLMGCVAETGTQCVN
-VLLMGCVAETGTQCVNL
-LLMGCVAETGTQCVNLT
-LMGCVAETGTQCVNLTT
-MGCVAETGTQCVNLTTR
-GCVAETGTQCVNLTTRT
-CVAETGTQCVNLTTRTQ
-VAETGTQCVNLTTRTQL
-AETGTQCVNLTTRTQLP
-ETGTQCVNLTTRTQLPP
-TGTQCVNLTTRTQLPPA
-GTQCVNLTTRTQLPPAY
-TQCVNLTTRTQLPPAYT
-QCVNLTTRTQLPPAYTN
-CVNLTTRTQLPPAYTNS
-VNLTTRTQLPPAYTNSF
-NLTTRTQLPPAYTNSFT
-LTTRTQLPPAYTNSFTR
```



```
-TRTQLPPAYTNSFTRGV
```

```
-RTQLPPAYTNSFTRGVY
-TQLPPAYTNSFTRGVYY
-QLPPAYTNSFTRGVYYP
-LPPAYTNSFTRGVYYPD
-PPAYTNSFTRGVYYPDK
-PAYTNSFTRGVYYPDKV
-AYTNSFTRGVYYPDKVF
-YTNSFTRGVYYPDKVFR
-TNSFTRGVYYPDKVFRS
-NSFTRGVYYPDKVFRSS
-SFTRGVYYPDKVFRSSV
-FTRGVYYPDKVFRSSVL
-TRGVYYPDKVFRSSVLH
-RGVYYPDKVFRSSVLHS
-GVYYPDKVFRSSVLHST
-VYYPDKVFRSSVLHSTQ
-YYPDKVFRSSVLHSTQD
-YPDKVFRSSVLHSTQDL
-MGILPSPGMPALLSLVSL
-GILPSPGMPALLSLVSLL
-ILPSPGMPALLSLVSLLS
-LPSPGMPALLSLVSLLSV
-PSPGMPALLSLVSLLSVL
-SPGMPALLSLVSLLSVLL
-PGMPALLSLVSLLSVLLM
-GMPALLSLVSLLSVLLMG
-MPALLSLVSLLSVLLMGC
-PALLSLVSLLSVLLMGCV
-ALLSLVSLLSVLLMGCVA
-LLSLVSLLSVLLMGCVAE
-LSLVSLLSVLLMGCVAET
-SLVSLLSVLLMGCVAETG
-LVSLLSVLLMGCVAETGT
-VSLLSVLLMGCVAETGTQ
-SLLSVLLMGCVAETGTQC
-LLSVLLMGCVAETGTQCV
-LSVLLMGCVAETGTQCVN
-SVLLMGCVAETGTQCVNL
-VLLMGCVAETGTQCVNLT
-LLMGCVAETGTQCVNLTT
-LMGCVAETGTQCVNLTTR
-MGCVAETGTQCVNLTTRT
-GCVAETGTQCVNLTTRTQ
-CVAETGTQCVNLTTRTQL
-VAETGTQCVNLTTRTQLP
-AETGTQCVNLTTRTQLPP
-ETGTQCVNLTTRTQLPPA
-TGTQCVNLTTRTQLPPAY
-GTQCVNLTTRTQLPPAYT
-TQCVNLTTRTQLPPAYTN
-QCVNLTTRTQLPPAYTNS
-CVNLTTRTQLPPAYTNSF
-VNLTTRTQLPPAYTNSFT
-NLTTRTQLPPAYTNSFTR
-LTTRTQLPPAYTNSFTRG
-TTRTQLPPAYTNSFTRGV
-TRTQLPPAYTNSFTRGVY
-RTQLPPAYTNSFTRGVYY
-TQLPPAYTNSFTRGVYYP
-QLPPAYTNSFTRGVYYPD
-LPPAYTNSFTRGVYYPDK
-PPAYTNSFTRGVYYPDKV
-PAYTNSFTRGVYYPDKVF
```



```
-YTNSFTRGVYYPDKVFRS
```

-TNSFTRGVYYPDKVFRSS
-NSFTRGVYYPDKVFRSSV
-SFTRGVYYPDKVFRSSVL
-FTRGVYYPDKVFRSSVLH
-TRGVYYPDKVFRSSVLHS
-RGVYYPDKVFRSSVLHST
-GVYYPDKVFRSSVLHSTQ
-VYYPDKVFRSSVLHSTQD
-YYPDKVFRSSVLHSTQDL
-MGILPSPGMPALLSLVSLL
-GILPSPGMPALLSLVSLLS
-ILPSPGMPALLSLVSLLSV
-LPSPGMPALLSLVSLLSVL
-PSPGMPALLSLVSLLSVLL
-SPGMPALLSLVSLLSVLLM
-PGMPALLSLVSLLSVLLMG
-GMPALLSLVSLLSVLLMGC
-MPALLSLVSLLSVLLMGCV
-PALLSLVSLLSVLLMGCVA
-ALLSLVSLLSVLLMGCVAE
-LLSLVSLLSVLLMGCVAET
-LSLVSLLSVLLMGCVAETG
-SLVSLLSVLLMGCVAETGT
-LVSLLSVLLMGCVAETGTQ
-VSLLSVLLMGCVAETGTQC
-SLLSVLLMGCVAETGTQCV
-LLSVLLMGCVAETGTQCVN
-LSVLLMGCVAETGTQCVNL
-SVLLMGCVAETGTQCVNLT
-VLLMGCVAETGTQCVNLTT
-LLMGCVAETGTQCVNLTTR
-LMGCVAETGTQCVNLTTRT
-MGCVAETGTQCVNLTTRTQ
-GCVAETGTQCVNLTTRTQL
-CVAETGTQCVNLTTRTQLP
-VAETGTQCVNLTTRTQLPP
-AETGTQCVNLTTRTQLPPA
-ETGTQCVNLTTRTQLPPAY
-TGTQCVNLTTRTQLPPAYT
-GTQCVNLTTRTQLPPAYTN
-TQCVNLTTRTQLPPAYTNS
-QCVNLTTRTQLPPAYTNSF
-CVNLTTRTQLPPAYTNSFT
-VNLTTRTQLPPAYTNSFTR
-NLTTRTQLPPAYTNSFTRG
-LTTRTQLPPAYTNSFTRGV
-TTRTQLPPAYTNSFTRGVY
-TRTQLPPAYTNSFTRGVYY
-RTQLPPAYTNSFTRGVYYP
-TQLPPAYTNSFTRGVYYPD
-QLPPAYTNSFTRGVYYPDK
-LPPAYTNSFTRGVYYPDKV
-PPAYTNSFTRGVYYPDKVF
-PAYTNSFTRGVYYPDKVFR
-AYTNSFTRGVYYPDKVFRS
-YTNSFTRGVYYPDKVFRSS
-TNSFTRGVYYPDKVFRSSV
-NSFTRGVYYPDKVFRSSVL
-SFTRGVYYPDKVFRSSVLH
-FTRGVYYPDKVFRSSVLHS
-TRGVYYPDKVFRSSVLHST
-RGVYYPDKVFRSSVLHSTQ
-GVYYPDKVFRSSVLHSTQD

-YYPDKVFRSSVLHSTQDLF

```
-MGILPSPGMPALLSLVSLLS
-GILPSPGMPALLSLVSLLSV
-ILPSPGMPALLSLVSLLSVL
-LPSPGMPALLSLVSLLSVLL
-PSPGMPALLSLVSLLSVLLM
-SPGMPALLSLVSLLSVLLMG
-PGMPALLSLVSLLSVLLMGC
-GMPALLSLVSLLSVLLMGCV
-MPALLSLVSLLSVLLMGCVA
-PALLSLVSLLSVLLMGCVAE
-ALLSLVSLLSVLLMGCVAET
-LLSLVSLLSVLLMGCVAETG
-LSLVSLLSVLLMGCVAETGT
-SLVSLLSVLLMGCVAETGTQ
-LVSLLSVLLMGCVAETGTQC
-VSLLSVLLMGCVAETGTQCV
-SLLSVLLMGCVAETGTQCVN
-LLSVLLMGCVAETGTQCVNL
-LSVLLMGCVAETGTQCVNLT
-SVLLMGCVAETGTQCVNLTT
-VLLMGCVAETGTQCVNLTTR
-LLMGCVAETGTQCVNLTTRT
-LMGCVAETGTQCVNLTTRTQ
-MGCVAETGTQCVNLTTRTQL
-GCVAETGTQCVNLTTRTQLP
-CVAETGTQCVNLTTRTQLPP
-VAETGTQCVNLTTRTQLPPA
-AETGTQCVNLTTRTQLPPAY
-ETGTQCVNLTTRTQLPPAYT
-TGTQCVNLTTRTQLPPAYTN
-GTQCVNLTTRTQLPPAYTNS
-TQCVNLTTRTQLPPAYTNSF
-QCVNLTTRTQLPPAYTNSFT
-CVNLTTRTQLPPAYTNSFTR
-VNLTTRTQLPPAYTNSFTRG
-NLTTRTQLPPAYTNSFTRGV
-LTTRTQLPPAYTNSFTRGVY
-TTRTQLPPAYTNSFTRGVYY
-TRTQLPPAYTNSFTRGVYYP
-RTQLPPAYTNSFTRGVYYPD
-TQLPPAYTNSFTRGVYYPDK
-QLPPAYTNSFTRGVYYPDKV
-LPPAYTNSFTRGVYYPDKVF
-PPAYTNSFTRGVYYPDKVFR
-PAYTNSFTRGVYYPDKVFRS
-AYTNSFTRGVYYPDKVFRSS
-YTNSFTRGVYYPDKVFRSSV
-TNSFTRGVYYPDKVFRSSVL
-NSFTRGVYYPDKVFRSSVLH
-SFTRGVYYPDKVFRSSVLHS
-FTRGVYYPDKVFRSSVLHST
-TRGVYYPDKVFRSSVLHSTQ
-RGVYYPDKVFRSSVLHSTQD
```