CSCI 481: Data Mining
Assignment 5
Due Date: 5/6/2021 11:59pm

Please type in your assignment and submit it online through Canvas by the due data. No hard copies will be accepted.

Introduction:

This assignment is entirely based around the corresponding spreadsheet included in this assignment. According to the directions, up to three attempts are allowed in that spreadsheet, denoted by different columns. In my spreadsheet, column 1 is filled with predictions based off of the KNN example that was directed in class on May 4th. Column 2 is populated with my own personally made KNN prediction, built separately from the class example. To finish off with some variety, column 3 is occupied with a Random Forest prediction.

Column 1: KNN That was done in class:

I decided to include this example mainly as a baseline, giving a platform to compare the rest of my methods to. I manipulated the validation code and some of the training data to be able to test my methods to see how well I could be doing. The best scoe I was able to get was a 0.9885 score seen score, 0.3599 unseen score, resulting in a 0.7199 total using 30% of the emb training data for test and the other 70% for training.

Column 2: Other KNN:

My personal KNN prediction used the emb_test and emb_train data. Using this data, I created a KNN model using the pdist function with seuclidean distance. I did this a little differently, first dividing the train data into two parts, then running pdist2 on both halves and the whole, making three separate distribution sets. I took the two sets made off of the halves and used them to get a better idea as to what data had closer or further distances. This stratagy did not end up working, creating two datasets that

were vastly different and unusable. I decided to instead do a low threshold replacement, replacing all values above a distance of 13.

Column 3: Random Forest:

In creating the prediction with a Random Forest, I decided that I wanted to use the actual nucleotide sequences as the predictor variable. This was the prediction I spent the most time on. The nucleotide sequences are not in a format that can be directly fed into the MATLAB TreeBagger function. The provided nuc_train and nuc_test data is in cell format. This format is incompatible with TreeBagger, as TreeBagger requires the X element to be a matrix. To transform the sequences into a format that was acceptable to the function, I decided to use an algorithm that I implemented in Assignment 1 Question 5. In that question, tokens were to be derived from nucleotide sequences and a multinomial probability vector was to be created. This method used tokens in singlet, doublet, and triplet format. I did not expect this to provide enough precision, so I added quartet, quintet, and sextet tokens, increasing the unique token count from 84 to 5460.

This, however, introduced an unforeseen problem. The script I created and expanded off of the algorithm from assignment 1 took substantial time to run. Processing 16,128 separate sequences, some of which were over 700 characters long, was not very quick; an average of 3.8 million different tokens would have to be checked per sequence. This is around 61.6 billion loops; after running the script for an hour, only 67 of the 16,128 sequences were parsed through. At this rate, the script would not have finished in over 10 days, only taking the training data into account. The test data added another 5989 sequences, most likely adding 3 or 4 days on top of the 10. I stopped the script and decided to remove sextet tokens from the token list, reducing the number to 1364. After running this reduced script for around 45 minutes, a check reveled that only around 200 of the sequences had been checked. This would still have taken around 2 1/2 days to run the training data. It was Tuesday evening and was not confident that if I let this run through that I would finish on time. I decided to reduce the script again, removing quintet and only checking for up to quartet. Going from 1364 to 340 created a script that was much more manageable.

After running the script overnight, the test data was able to be token the test data. After running the training data for almost 11 hours, my

computer decided that it needed to update and gave me 15 minutes to wrap up everything before it was going to restart. I paused the script, taking account that 9477 of the 16128 sequences had been parsed across 10.836 hours. A quick calculation ($\frac{9477}{16128} = \frac{10.836}{?}$, $\frac{(10.836*16128)}{9477} = 18.44$) revealed that the script was to take a total of around 18 $1/2$ hours total. Since the script still had $(18.44 - 10.836 = 7.6)$ around 7 $1/2$ hours remaining, I decided that I was going to have to cut the script short and try to continue the rest after the restart. I stopped the script at 9897. ($\frac{9897}{16128} = 0.6137$) At around 61.4% of the data parsed, I was very disappointed. I at this point, it was Wednesday evening. I only had around a day left and did not think it wise to tempt fate and try to run the data through again, knowing that the TreeBagger function took a few hours to run itself. I did not like the thought of only using $< \frac{2}{3}$ of the data and I believe that my results suffered, creating a weaker model. Nevertheless, I persisted, using the data I did have to create the prediction.

I created a validation algorithm based off of the one presented in class. When dividing dividing the data, I split it 25% test, 75% train, making sure to intermix the data and giving every entry of 75 of the 775 unique species left to test to create unseen sector in the data. Using test data, I created two separate TreeBagger models, one for ytrain species data and one for gtrain genus data. Each was made with 100 trees. any more than 100 had a negligible effect on model success. When creating the corresponding predictions, I had the functions also print out a score table. This table specified the likelihood of each observation being a part of each possible classifier. Using this table, I checked for species predictions with a score less than 0.2. If so, I checked to see if the corresponding genus prediction was above 0.3. If it was, I swapped the genus prediction in for the species prediction. If it was lower, I swapped in the corresponding gtrain data for that entry.

In testing, This prediction model did as well as 0.981 on seen data and 0.260 on unseen data, resulting in a total score of 0.206. The unseen score left much to be desired; I had difficulty in effectively dividing my test data since a portion of it had to be unseen. I also acknowledge that my model may preform much worse as well; I did not process an astounding 6231 sequences. This will have a serious negative impact on the success of this prediction. Since leaving out a large section of training entries created long sections of zeros in the prediction, all zero entries in the prediction were replaced with values from my other KNN prediction.

Conclusion:

Overall, I believe that my predictions will do well. I believe that my Random Forest had potential to do much better, but time constraints required it to be cut short. I still believe it will do the best, as it is primarily made of predictions generated by TreeBagger, only near the end of the prediction was making an ensemble necessary. Nevertheless, I am content that despite the shortcomings, my predictions may still be relatively accurate.

Code Appendix:

Column 1: KNN:

```
nt=size(emb_test,1);

for i=1:nt
    if rem(i,1000)==0
        disp(i)
    end
    [D(i),I(i)]=pdist2(emb_train,emb_test(i,:),'euclidean','smallest',1);
end

 ypred=ytrain(I);
 th=20;
 for i=1:nt
     if D(i)>th
         ypred(i)=unique(gtrain(ytrain==ypred(i)));
     end
 end

csvwrite('D:\Users\jason\Documents\IUPUI_S6_2021_Spring\CSCI 48100\HW5\sample_subm

%the numbers below are for my own testing. you have to update them based on
%your own validation set.
 ypred_seen=ypred(1:4033);
ypred_unseen=ypred(4034:5989);
ytest_seen=ytest(1:4033);
ytest_unseen=gtest(4034:5989);

classes=unique(ytest_unseen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(ytest_unseen==classes(i));
        acc_per_class(i) = sum(ytest_unseen(idx) == ypred_unseen(idx)) / length(id
    end
```

```
    gzsl_unseen_acc = mean(acc_per_class);


classes=unique(ytest_seen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(ytest_seen==classes(i));
        acc_per_class(i) = sum(ytest_seen(idx) == ypred_seen(idx)) / length(idx);
    end
    gzsl_seen_acc = mean(acc_per_class);
  H = 2 * gzsl_unseen_acc * gzsl_seen_acc / (gzsl_unseen_acc + gzsl_seen_acc);

  disp(['GZSL unseen: averaged per-class accuracy=' num2str(gzsl_unseen_acc) ]);
disp(['GZSL seen: averaged per-class accuracy=' num2str(gzsl_seen_acc) ]);
disp(['GZSL: H=' num2str(H)]);
```

Column 2: KNN:

```
%creating prediction method validation set.

%shuffles the training data so that proper unseen data is
%transfered correctly
%{
randCol = colIndex(randperm(size(colIndex, 1)), :);

rand_emb_train(randCol,:) = emb_train(colIndex,:);
rand_ytrain(randCol,:) = ytrain(colIndex,:);
rand_gtrain(randCol,:) = gtrain(colIndex,:);
%}

%Split training data 30% and 70%
%{
emb_train_0_30 = rand_emb_train(1:4838,:);
emb_train_31_100 = rand_emb_train(4839:16128,:);
```

```
ytrain_0_30 = rand_ytrain(1:4838,:);
ytrain_31_100 = rand_ytrain(4839:16128,:);
gtrain_0_30 = rand_gtrain(1:4838,:);
gtrain_31_100 = rand_gtrain(4839:16128,:);
%}

%for loop selects and transfers unique species in the 70% to the 30%
%to create unclassified(unseen) species
%
%{
uSp = size(unique(ytrain),1);
n = size(ytrain_31_100,1);
m = size(ytrain_0_30,1)+1;
for i=1:uSp
    r = randi([i,i+74]);
     if mod(r, 50) == 0
         disp(i);
         for j=1:n
             if j < n
                 if ytrain_31_100(j,1) == i
                    m = size(ytrain_0_30,1)+1;
                    emb_train_0_30(m,:) = emb_train_31_100(j,:);
                    emb_train_31_100(j,:) = [];
                    ytrain_0_30(m,:) = ytrain_31_100(j,:);
                    ytrain_31_100(j,:) = [];
                    gtrain_0_30(m,:) = gtrain_31_100(j,:);
                    gtrain_31_100(j,:) = [];
                    n = size(ytrain_31_100,1);
                    j = j-1;
                 end
             end
         end
         j= 0;
     end
end
 %}

%creates KNN distribution based on the training data
```

```
%{
sz=size(emb_train_0_30,1);

for i=1:sz
    if rem(i,250)==0
        disp(i)
    end
    [D(i),I(i)]=pdist2(emb_train_31_100,emb_train_0_30(i,:),'euclidean','smallest'
end
%}

%creates prediction and changes specific species into genus
%based on KNN threshold
%{
 ypred_0_30=ytrain_31_100(I);
 th=25;
 for i=1:sz
     if D(i)>th
         disp(i)
         ypred_0_30(i)=unique(gtrain_31_100(ytrain_31_100==ypred_0_30(i)));
     end
 end
%}
%{
[gR, ~] = find(ismember(ytrain_0_30, setdiff(unique(ytrain_0_30),unique(ytrain_31_
ypred_0_30_seen=ypred_0_30(:);
ypred_0_30_seen(gR)=[];
ypred_0_30_unseen=ypred_0_30(gR);
ytrain_0_30_seen=ytrain_0_30(:);
ytrain_0_30_seen(gR)=[];
ytrain_0_30_unseen=gtrain_0_30(gR);
%}

%{
classes=unique(ytrain_0_30_unseen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
```

```
        idx = find(ytrain_0_30_unseen==classes(i));
        acc_per_class(i) = sum(ytrain_0_30_unseen(idx) == ypred_0_30_unseen(idx))
    end
    gzsl_unseen_acc = mean(acc_per_class);

classes=unique(ytrain_0_30_seen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(ytrain_0_30_seen==classes(i));
        acc_per_class(i) = sum(ytrain_0_30_seen(idx) == ypred_0_30_seen(idx)) / le
    end
    gzsl_seen_acc = mean(acc_per_class);
  H = 2 * gzsl_unseen_acc * gzsl_seen_acc / (gzsl_unseen_acc + gzsl_seen_acc);
%}

%{
  disp(['GZSL unseen: averaged per-class accuracy=' num2str(gzsl_unseen_acc) ]);
disp(['GZSL seen: averaged per-class accuracy=' num2str(gzsl_seen_acc) ]);
disp(['GZSL: H=' num2str(H)]);
%}

hsz=size(emb_test,1)/2;
sz=size(emb_test,1);
for i=1:hsz
    if rem(i,250)==0
        disp(i)
    end
    [Q(i),G(i)]=pdist2(emb_train(1:8064,:),emb_test(i,:),'seuclidean','smallest',1
    [R(i),J(i)]=pdist2(emb_train(8065:16128,:),emb_test(i,:),'seuclidean','smalles
end
for i=1:sz
    if rem(i,290)==0
        disp(i)
    end
    [V(i),X(i)]=pdist2(emb_train,emb_test(i,:),'euclidean','smallest',1);

end
```

```
 ypredF=ytrain(X);
 th=13;
 for i=1:sz
     if V(i)>th
         ypredF(i)=unique(gtrain(ytrain==ypredF(i)));
     end
 end

csvwrite('D:\Users\jason\Documents\IUPUI_S6_2021_Spring\CSCI 48100\HW5\submissionK
%}
%creates prediction and changes specific species into genus
%based on KNN threshold
```

Column 3: Random Forest:

```
%special characters 'N' and '-' denote missing values
%{
testTokens = exists(1:340, 1:5989);
trainTokens = existsTR(1:340, 1:9896);
trainTokensC = trainTokens;
testTokensC = testTokens;
trainTokens = trainTokensC';
testTokens = testTokensC';
ytrainReduced = ytrain(1:9896);
gtrainReduced = gtrain(1:9896);

totalR = zeros(9096,1);
r = 0;
n = 68;
z = n;
m = 0;
q = 0;
for i = 1:9896
    totalR(i,1) = randi([1,8]);
    if totalR(i,1) > 1 && n > 0
        n = n - 1;
```

```
    else
        r = r + 1;
        if n == 0
            m = m + 1;
            totalR(i,1) = 1;
            if m == 16
                n = z;
                m = 0;
                q = q + 1;
            end
        end
    end
end
ytrainR25 = zeros(r,1);
gtrainR25 = zeros(r,1);
trainT25 = zeros(r,340);
ytrainR75 = zeros(9896 - r,1);
gtrainR75 = zeros(9896 - r,1);
trainT75 = zeros(9896 - r,340);

j = 1;
k = 1;
for i = 1:9896
    if totalR(i,1) == 1
        ytrainR25(j,:) = ytrainReduced(i,:);
        gtrainR25(j,:) = gtrainReduced(i,:);
        trainT25(j,:) = trainTokens(i,:);
        j = j + 1;
    else
        ytrainR75(k,:) = ytrainReduced(i,:);
        gtrainR75(k,:) = gtrainReduced(i,:);
        trainT75(k,:) = trainTokens(i,:);
        k = k + 1;
    end
end
tokenWeight = zeros(1, 340);
tokenWeight(1, 1:4) = .2;
tokenWeight(1, 5:20) = .3;
```

```
tokenWeight(1, 21:84) = .9;
tokenWeight(1, 85:340) = 1;



tr75sBagger = TreeBagger(100, trainT75, ytrainR75);
tr75gBagger = TreeBagger(100, trainT75, gtrainR75);

[cell75s25,score75s25] = predict(tr75sBagger, trainT25);
[cell75g25,score75g25] = predict(tr75gBagger, trainT25);

tr75sPred25 = str2double(cell75s25);
tr75gPred25 = str2double(cell75g25);
sz = size(tr75sPred25,1);
tr75ScoredPred25 = zeros(sz,1);

for i = 1:sz
    if max(score75s25(i,:)) < 0.2
        if max(score75g25(i,:)) > max(score75s25(i,:))
            tr75ScoredPred25(i,:) = tr75gPred25(i,:);
        else
            unique(gtrainR75(ytrainR75==tr75sPred25(i)));
        end
    else
        tr75ScoredPred25(i,:) = tr75sPred25(i,:);
    end
end

%//////////////////////////////////////////

[tsR, ~] = find(ismember(ytrainR25, setdiff(unique(ytrainR25),unique(ytrainR75))))
tr75sPred25Seen=tr75sPred25(:);
tr75sPred25Seen(tsR)=[];
tr75sPred25Unseen=tr75sPred25(tsR);
ytrainR25Seen=ytrainR25(:);
ytrainR25Seen(tsR)=[];
ytrainR25Unseen=ytrainR25(tsR);
```

```
classes=unique(ytrainR25Unseen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(ytrainR25Unseen==classes(i));
        acc_per_class(i) = sum(ytrainR25Unseen(idx) == tr75sPred25Unseen(idx)) / l
    end
    gzsl_unseen_acc = mean(acc_per_class);

classes=unique(ytrainR25Seen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(ytrainR25Seen==classes(i));
        acc_per_class(i) = sum(ytrainR25Seen(idx) == tr75sPred25Seen(idx)) / lengt
    end
    gzsl_seen_acc = mean(acc_per_class);
  H = 2 * gzsl_unseen_acc * gzsl_seen_acc / (gzsl_unseen_acc + gzsl_seen_acc);

  disp(['Species unseen: averaged per-class accuracy=' num2str(gzsl_unseen_acc) ])
disp(['Species seen: averaged per-class accuracy=' num2str(gzsl_seen_acc) ]);
disp(['Species: H=' num2str(H)]);

%//////////////////////////////////////

tr75gPred25Seen=tr75gPred25(:);
tr75gPred25Seen(tsR)=[];
tr75gPred25Unseen=tr75gPred25(tsR);
gtrainR25Seen=gtrainR25(:);
gtrainR25Seen(tsR)=[];
gtrainR25Unseen=gtrainR25(tsR);

classes=unique(gtrainR25Unseen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(gtrainR25Unseen==classes(i));
        acc_per_class(i) = sum(gtrainR25Unseen(idx) == tr75gPred25Unseen(idx)) / l
```

```
    end
    gzsl_unseen_acc = mean(acc_per_class);

classes=unique(gtrainR25Seen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(gtrainR25Seen==classes(i));
        acc_per_class(i) = sum(gtrainR25Seen(idx) == tr75gPred25Seen(idx)) / lengt
    end
    gzsl_seen_acc = mean(acc_per_class);
  H = 2 * gzsl_unseen_acc * gzsl_seen_acc / (gzsl_unseen_acc + gzsl_seen_acc);

  disp(['Genus unseen: averaged per-class accuracy=' num2str(gzsl_unseen_acc) ]);
disp(['Genus seen: averaged per-class accuracy=' num2str(gzsl_seen_acc) ]);
disp(['Genus: H=' num2str(H)]);

%//////////////////////////////////////

tr75ScoredPred25Seen=tr75ScoredPred25(:);
tr75ScoredPred25Seen(tsR)=[];
tr75ScoredPred25Unseen=tr75ScoredPred25(tsR);
scoredSeen=ytrainR25(:);
scoredSeen(tsR)=[];
scoredUnseen=gtrainR25(tsR);

classes=unique(scoredUnseen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
    for i=1:nclass
        idx = find(scoredUnseen==classes(i));
        acc_per_class(i) = sum(scoredUnseen(idx) == tr75ScoredPred25Unseen(idx)) /
    end
    gzsl_unseen_acc = mean(acc_per_class);

classes=unique(scoredSeen);
    nclass = length(classes);
    acc_per_class = zeros(nclass, 1);
```

```
    for i=1:nclass
        idx = find(scoredSeen==classes(i));
        acc_per_class(i) = sum(scoredSeen(idx) == tr75ScoredPred25Seen(idx)) / len
    end
    gzsl_seen_acc = mean(acc_per_class);
  H = 2 * gzsl_unseen_acc * gzsl_seen_acc / (gzsl_unseen_acc + gzsl_seen_acc);

  disp(['Scored unseen: averaged per-class accuracy=' num2str(gzsl_unseen_acc) ]);
disp(['Scored seen: averaged per-class accuracy=' num2str(gzsl_seen_acc) ]);
disp(['Scored: H=' num2str(H)]);

%//////////////////////////////////////
%}

trainBaggerS = TreeBagger(100, trainTokens, ytrainReduced);
trainBaggerG = TreeBagger(100, trainTokens, ytrainReduced);

[cellS,scoreS] = predict(trainBaggerS, testTokens);
[cellG,scoreG] = predict(trainBaggerG, testTokens);

sPrediction = str2double(cellS);
gPrediction = str2double(cellG);
sz = size(sPrediction,1);
finalPrediction = zeros(sz,1);

for i = 1:sz
    if max(scoreS(i,:)) < 0.2
        if max(scoreG(i,:)) > 0.2
            finalPrediction(i,:) = gPrediction(i,:);
        else
            unique(gtrainReduced(ytrainReduced==sPrediction(i)));
        end
    else
        finalPrediction(i,:) = sPrediction(i,:);
    end
end

csvwrite('D:\Users\jason\Documents\IUPUI_S6_2021_Spring\CSCI 48100\HW5\rForestSubm
```

```
singlet = ["A", "T", "C", "G"];
doublet = ["AT", "AA", "TA", "TT", "AG", "GG", "GA", "AC", "CC", "CA", "TC", "TG",
triplet = ["AAT", "AAA", "ATA", "ATT", "AAG", "AGG", "AGA", "AAC", "ACC", "ACA", "
"ACG", "AGC", "TAT", "TAA", "TTA", "TTT", "TAG", "TGG", "TGA", "TAC", "TCC", "TCA'
"TCT", "TGT", "TCG", "TGC", "GAT", "GAA", "GTA", "GTT", "GAG", "GGG", "GGA", "GAC'
"GTC", "GTG", "GCT", "GGT", "GCG", "GGC", "CAT", "CAA", "CTA", "CTT", "CAG", "CGG'
"CCC", "CCA", "CTC", "CTG", "CCT", "CGT", "CCG", "CGC"];
quartet = ["AAAA", "AAAT", "AAAC", "AAAG", "AATA", "AATT", "AATC", "AATG", "AACA",
"AAGT", "AAGC", "AAGG", "ATAA", "ATAT", "ATAC", "ATAG", "ATTA", "ATTT", "ATTC", "A
"ATCC", "ATCG", "ATGA", "ATGT", "ATGC", "ATGG", "ACAA", "ACAT", "ACAC", "ACAG", "A
"ACTG", "ACCA", "ACCT", "ACCC", "ACCG", "ACGA", "ACGT", "ACGC", "ACGG", "AGAA", "A
"AGTA", "AGTT", "AGTC", "AGTG", "AGCA", "AGCT", "AGCC", "AGCG", "AGGA", "AGGT", "A
"TAAT", "TAAC", "TAAG", "TATA", "TATT", "TATC", "TATG", "TACA", "TACT", "TACC", "T
"TAGC", "TAGG", "TTAA", "TTAT", "TTAC", "TTAG", "TTTA", "TTTT", "TTTC", "TTTG", "T
"TTCG", "TTGA", "TTGT", "TTGC", "TTGG", "TCAA", "TCAT", "TCAC", "TCAG", "TCTA", "T
"TCCA", "TCCT", "TCCC", "TCCG", "TCGA", "TCGT", "TCGC", "TCGG", "TGAA", "TGAT", "T
"TGTT", "TGTC", "TGTG", "TGCA", "TGCT", "TGCC", "TGCG", "TGGA", "TGGT", "TGGC", "T
"CAAC", "CAAG", "CATA", "CATT", "CATC", "CATG", "CACA", "CACT", "CACC", "CACG", "C
"CAGG", "CTAA", "CTAT", "CTAC", "CTAG", "CTTA", "CTTT", "CTTC", "CTTG", "CTCA", "C
"CTGA", "CTGT", "CTGC", "CTGG", "CCAA", "CCAT", "CCAC", "CCAG", "CCTA", "CCTT", "C
"CCCT", "CCCC", "CCCG", "CCGA", "CCGT", "CCGC", "CCGG", "CGAA", "CGAT", "CGAC", "C
"CGTC", "CGTG", "CGCA", "CGCT", "CGCC", "CGCG", "CGGA", "CGGT", "CGGC", "CGGG", "C
"GAAG", "GATA", "GATT", "GATC", "GATG", "GACA", "GACT", "GACC", "GACG", "GAGA", "C
"GTAA", "GTAT", "GTAC", "GTAG", "GTTA", "GTTT", "GTTC", "GTTG", "GTCA", "GTCT", "C
"GTGT", "GTGC", "GTGG", "GCAA", "GCAT", "GCAC", "GCAG", "GCTA", "GCTT", "GCTC", "C
"GCCC", "GCCG", "GCGA", "GCGT", "GCGC", "GCGG", "GGAA", "GGAT", "GGAC", "GGAG", "C
"GGTG", "GGCA", "GGCT", "GGCC", "GGCG", "GGGA", "GGGT", "GGGC", "GGGG"];
%{
        quintet = ["AAAAA", "AAAAT", "AAAAC", "AAAAG", "AAATA", "AAATT", "AAATC",
"AAAGA", "AAAGT", "AAAGC", "AAAGG", "AATAA", "AATAT", "AATAC", "AATAG", "AATTA", "
"AATCA", "AATCT", "AATCC", "AATCG", "AATGA", "AATGT", "AATGC", "AATGG", "AACAA", "
"AACTA", "AACTT", "AACTC", "AACTG", "AACCA", "AACCT", "AACCC", "AACCG", "AACGA", "
"AAGAA", "AAGAT", "AAGAC", "AAGAG", "AAGTA", "AAGTT", "AAGTC", "AAGTG", "AAGCA", "
"AAGGA", "AAGGT", "AAGGC", "AAGGG", "ATAAA", "ATAAT", "ATAAC", "ATAAG", "ATATA", "
"ATACA", "ATACT", "ATACC", "ATACG", "ATAGA", "ATAGT", "ATAGC", "ATAGG", "ATTAA", "
"ATTTA", "ATTTT", "ATTTC", "ATTTG", "ATTCA", "ATTCT", "ATTCC", "ATTCG", "ATTGA", "
"ATCAA", "ATCAT", "ATCAC", "ATCAG", "ATCTA", "ATCTT", "ATCTC", "ATCTG", "ATCCA", "
"ATCGA", "ATCGT", "ATCGC", "ATCGG", "ATGAA", "ATGAT", "ATGAC", "ATGAG", "ATGTA", "
```

"ATGCA", "ATGCT", "ATGCC", "ATGCG", "ATGGA", "ATGGT", "ATGGC", "ATGGG", "ACAAA", "
"ACATA", "ACATT", "ACATC", "ACATG", "ACACA", "ACACT", "ACACC", "ACACG", "ACAGA", "
"ACTAA", "ACTAT", "ACTAC", "ACTAG", "ACTTA", "ACTTT", "ACTTC", "ACTTG", "ACTCA", "
"ACTGA", "ACTGT", "ACTGC", "ACTGG", "ACCAA", "ACCAT", "ACCAC", "ACCAG", "ACCTA", "
"ACCCA", "ACCCT", "ACCCC", "ACCCG", "ACCGA", "ACCGT", "ACCGC", "ACCGG", "ACGAA", "
"ACGTA", "ACGTT", "ACGTC", "ACGTG", "ACGCA", "ACGCT", "ACGCC", "ACGCG", "ACGGA", "
"AGAAA", "AGAAT", "AGAAC", "AGAAG", "AGATA", "AGATT", "AGATC", "AGATG", "AGACA", "
"AGAGA", "AGAGT", "AGAGC", "AGAGG", "AGTAA", "AGTAT", "AGTAC", "AGTAG", "AGTTA", "
"AGTCA", "AGTCT", "AGTCC", "AGTCG", "AGTGA", "AGTGT", "AGTGC", "AGTGG", "AGCAA", "
"AGCTA", "AGCTT", "AGCTC", "AGCTG", "AGCCA", "AGCCT", "AGCCC", "AGCCG", "AGCGA", "
"AGGAA", "AGGAT", "AGGAC", "AGGAG", "AGGTA", "AGGTT", "AGGTC", "AGGTG", "AGGCA", "
"AGGGA", "AGGGT", "AGGGC", "AGGGG", "TAAAA", "TAAAT", "TAAAC", "TAAAG", "TAATA", "
"TAACA", "TAACT", "TAACC", "TAACG", "TAAGA", "TAAGT", "TAAGC", "TAAGG", "TATAA", "
"TATTA", "TATTT", "TATTC", "TATTG", "TATCA", "TATCT", "TATCC", "TATCG", "TATGA", "
"TACAA", "TACAT", "TACAC", "TACAG", "TACTA", "TACTT", "TACTC", "TACTG", "TACCA", "
"TACGA", "TACGT", "TACGC", "TACGG", "TAGAA", "TAGAT", "TAGAC", "TAGAG", "TAGTA", "
"TAGCA", "TAGCT", "TAGCC", "TAGCG", "TAGGA", "TAGGT", "TAGGC", "TAGGG", "TTAAA", "
"TTATA", "TTATT", "TTATC", "TTATG", "TTACA", "TTACT", "TTACC", "TTACG", "TTAGA", "
"TTTAA", "TTTAT", "TTTAC", "TTTAG", "TTTTA", "TTTTT", "TTTTC", "TTTTG", "TTTCA", "
"TTTGA", "TTTGT", "TTTGC", "TTTGG", "TTCAA", "TTCAT", "TTCAC", "TTCAG", "TTCTA", "
"TTCCA", "TTCCT", "TTCCC", "TTCCG", "TTCGA", "TTCGT", "TTCGC", "TTCGG", "TTGAA", "
"TTGTA", "TTGTT", "TTGTC", "TTGTG", "TTGCA", "TTGCT", "TTGCC", "TTGCG", "TTGGA", "
"TCAAA", "TCAAT", "TCAAC", "TCAAG", "TCATA", "TCATT", "TCATC", "TCATG", "TCACA", "
"TCAGA", "TCAGT", "TCAGC", "TCAGG", "TCTAA", "TCTAT", "TCTAC", "TCTAG", "TCTTA", "
"TCTCA", "TCTCT", "TCTCC", "TCTCG", "TCTGA", "TCTGT", "TCTGC", "TCTGG", "TCCAA", "
"TCCTA", "TCCTT", "TCCTC", "TCCTG", "TCCCA", "TCCCT", "TCCCC", "TCCCG", "TCCGA", "
"TCGAA", "TCGAT", "TCGAC", "TCGAG", "TCGTA", "TCGTT", "TCGTC", "TCGTG", "TCGCA", "
"TCGGA", "TCGGT", "TCGGC", "TCGGG", "TGAAA", "TGAAT", "TGAAC", "TGAAG", "TGATA", "
"TGACA", "TGACT", "TGACC", "TGACG", "TGAGA", "TGAGT", "TGAGC", "TGAGG", "TGTAA", "
"TGTTA", "TGTTT", "TGTTC", "TGTTG", "TGTCA", "TGTCT", "TGTCC", "TGTCG", "TGTGA", "
"TGCAA", "TGCAT", "TGCAC", "TGCAG", "TGCTA", "TGCTT", "TGCTC", "TGCTG", "TGCCA", "
"TGCGA", "TGCGT", "TGCGC", "TGCGG", "TGGAA", "TGGAT", "TGGAC", "TGGAG", "TGGTA", "
"TGGCA", "TGGCT", "TGGCC", "TGGCG", "TGGGA", "TGGGT", "TGGGC", "TGGGG", "CAAAA", "
"CAATA", "CAATT", "CAATC", "CAATG", "CAACA", "CAACT", "CAACC", "CAACG", "CAAGA", "
"CATAA", "CATAT", "CATAC", "CATAG", "CATTA", "CATTT", "CATTC", "CATTG", "CATCA", "
"CATGA", "CATGT", "CATGC", "CATGG", "CACAA", "CACAT", "CACAC", "CACAG", "CACTA", "
"CACCA", "CACCT", "CACCC", "CACCG", "CACGA", "CACGT", "CACGC", "CACGG", "CAGAA", "
"CAGTA", "CAGTT", "CAGTC", "CAGTG", "CAGCA", "CAGCT", "CAGCC", "CAGCG", "CAGGA", "

```
"CTAAA", "CTAAT", "CTAAC", "CTAAG", "CTATA", "CTATT", "CTATC", "CTATG", "CTACA", "
"CTAGA", "CTAGT", "CTAGC", "CTAGG", "CTTAA", "CTTAT", "CTTAC", "CTTAG", "CTTTA", "
"CTTCA", "CTTCT", "CTTCC", "CTTCG", "CTTGA", "CTTGT", "CTTGC", "CTTGG", "CTCAA", "
"CTCTA", "CTCTT", "CTCTC", "CTCTG", "CTCCA", "CTCCT", "CTCCC", "CTCCG", "CTCGA", "
"CTGAA", "CTGAT", "CTGAC", "CTGAG", "CTGTA", "CTGTT", "CTGTC", "CTGTG", "CTGCA", "
"CTGGA", "CTGGT", "CTGGC", "CTGGG", "CCAAA", "CCAAT", "CCAAC", "CCAAG", "CCATA", "
"CCACA", "CCACT", "CCACC", "CCACG", "CCAGA", "CCAGT", "CCAGC", "CCAGG", "CCTAA", "
"CCTTA", "CCTTT", "CCTTC", "CCTTG", "CCTCA", "CCTCT", "CCTCC", "CCTCG", "CCTGA", "
"CCCAA", "CCCAT", "CCCAC", "CCCAG", "CCCTA", "CCCTT", "CCCTC", "CCCTG", "CCCCA", "
"CCCGA", "CCCGT", "CCCGC", "CCCGG", "CCGAA", "CCGAT", "CCGAC", "CCGAG", "CCGTA", "
"CCGCA", "CCGCT", "CCGCC", "CCGCG", "CCGGA", "CCGGT", "CCGGC", "CCGGG", "CGAAA", "
"CGATA", "CGATT", "CGATC", "CGATG", "CGACA", "CGACT", "CGACC", "CGACG", "CGAGA", "
"CGTAA", "CGTAT", "CGTAC", "CGTAG", "CGTTA", "CGTTT", "CGTTC", "CGTTG", "CGTCA", "
"CGTGA", "CGTGT", "CGTGC", "CGTGG", "CGCAA", "CGCAT", "CGCAC", "CGCAG", "CGCTA", "
"CGCCA", "CGCCT", "CGCCC", "CGCCG", "CGCGA", "CGCGT", "CGCGC", "CGCGG", "CGGAA", "
"CGGTA", "CGGTT", "CGGTC", "CGGTG", "CGGCA", "CGGCT", "CGGCC", "CGGCG", "CGGGA", "
"GAAAA", "GAAAT", "GAAAC", "GAAAG", "GAATA", "GAATT", "GAATC", "GAATG", "GAACA", "
"GAAGA", "GAAGT", "GAAGC", "GAAGG", "GATAA", "GATAT", "GATAC", "GATAG", "GATTA", "
"GATCA", "GATCT", "GATCC", "GATCG", "GATGA", "GATGT", "GATGC", "GATGG", "GACAA", "
"GACTA", "GACTT", "GACTC", "GACTG", "GACCA", "GACCT", "GACCC", "GACCG", "GACGA", "
"GAGAA", "GAGAT", "GAGAC", "GAGAG", "GAGTA", "GAGTT", "GAGTC", "GAGTG", "GAGCA", "
"GAGGA", "GAGGT", "GAGGC", "GAGGG", "GTAAA", "GTAAT", "GTAAC", "GTAAG", "GTATA", "
"GTACA", "GTACT", "GTACC", "GTACG", "GTAGA", "GTAGT", "GTAGC", "GTAGG", "GTTAA", "
"GTTTA", "GTTTT", "GTTTC", "GTTTG", "GTTCA", "GTTCT", "GTTCC", "GTTCG", "GTTGA", "
"GTCAA", "GTCAT", "GTCAC", "GTCAG", "GTCTA", "GTCTT", "GTCTC", "GTCTG", "GTCCA", "
"GTCGA", "GTCGT", "GTCGC", "GTCGG", "GTGAA", "GTGAT", "GTGAC", "GTGAG", "GTGTA", "
"GTGCA", "GTGCT", "GTGCC", "GTGCG", "GTGGA", "GTGGT", "GTGGC", "GTGGG", "GCAAA", "
"GCATA", "GCATT", "GCATC", "GCATG", "GCACA", "GCACT", "GCACC", "GCACG", "GCAGA", "
"GCTAA", "GCTAT", "GCTAC", "GCTAG", "GCTTA", "GCTTT", "GCTTC", "GCTTG", "GCTCA", "
"GCTGA", "GCTGT", "GCTGC", "GCTGG", "GCCAA", "GCCAT", "GCCAC", "GCCAG", "GCCTA", "
"GCCCA", "GCCCT", "GCCCC", "GCCCG", "GCCGA", "GCCGT", "GCCGC", "GCCGG", "GCGAA", "
"GCGTA", "GCGTT", "GCGTC", "GCGTG", "GCGCA", "GCGCT", "GCGCC", "GCGCG", "GCGGA", "
"GGAAA", "GGAAT", "GGAAC", "GGAAG", "GGATA", "GGATT", "GGATC", "GGATG", "GGACA", "
"GGAGA", "GGAGT", "GGAGC", "GGAGG", "GGTAA", "GGTAT", "GGTAC", "GGTAG", "GGTTA", "
"GGTCA", "GGTCT", "GGTCC", "GGTCG", "GGTGA", "GGTGT", "GGTGC", "GGTGG", "GGCAA", "
"GGCTA", "GGCTT", "GGCTC", "GGCTG", "GGCCA", "GGCCT", "GGCCC", "GGCCG", "GGCGA", "
"GGGAA", "GGGAT", "GGGAC", "GGGAG", "GGGTA", "GGGTT", "GGGTC", "GGGTG", "GGGCA", "
"GGGGA", "GGGGT", "GGGGC", "GGGGG"];
```

```
        %}
%{
        sextet =  ["AAAAAA", "AAAAAT", "AAAAAC", "AAAAAG", "AAAATA", "AAAATT", "AA
"AAAACG", "AAAAGA", "AAAAGT", "AAAAGC", "AAAAGG", "AAATAA", "AAATAT", "AAATAC", "A
"AAATTC", "AAATTG", "AAATCA", "AAATCT", "AAATCC", "AAATCG", "AAATGA", "AAATGT", "A
"AAACAT", "AAACAC", "AAACAG", "AAACTA", "AAACTT", "AAACTC", "AAACTG", "AAACCA", "A
"AAACGA", "AAACGT", "AAACGC", "AAACGG", "AAAGAA", "AAAGAT", "AAAGAC", "AAAGAG", "A
"AAAGTG", "AAAGCA", "AAAGCT", "AAAGCC", "AAAGCG", "AAAGGA", "AAAGGT", "AAAGGC", "A
"AATAAC", "AATAAG", "AATATA", "AATATT", "AATATC", "AATATG", "AATACA", "AATACT", "A
"AATAGT", "AATAGC", "AATAGG", "AATTAA", "AATTAT", "AATTAC", "AATTAG", "AATTTA", "A
"AATTCA", "AATTCT", "AATTCC", "AATTCG", "AATTGA", "AATTGT", "AATTGC", "AATTGG", "A
"AATCAG", "AATCTA", "AATCTT", "AATCTC", "AATCTG", "AATCCA", "AATCCT", "AATCCC", "A
"AATCGC", "AATCGG", "AATGAA", "AATGAT", "AATGAC", "AATGAG", "AATGTA", "AATGTT", "A
"AATGCT", "AATGCC", "AATGCG", "AATGGA", "AATGGT", "AATGGC", "AATGGG", "AACAAA", "A
"AACATA", "AACATT", "AACATC", "AACATG", "AACACA", "AACACT", "AACACC", "AACACG", "A
"AACAGG", "AACTAA", "AACTAT", "AACTAC", "AACTAG", "AACTTA", "AACTTT", "AACTTC", "A
"AACTCC", "AACTCG", "AACTGA", "AACTGT", "AACTGC", "AACTGG", "AACCAA", "AACCAT", "A
"AACCTT", "AACCTC", "AACCTG", "AACCCA", "AACCCT", "AACCCC", "AACCCG", "AACCGA", "A
"AACGAA", "AACGAT", "AACGAC", "AACGAG", "AACGTA", "AACGTT", "AACGTC", "AACGTG", "A
"AACGCG", "AACGGA", "AACGGT", "AACGGC", "AACGGG", "AAGAAA", "AAGAAT", "AAGAAC", "A
"AAGATC", "AAGATG", "AAGACA", "AAGACT", "AAGACC", "AAGACG", "AAGAGA", "AAGAGT", "A
"AAGTAT", "AAGTAC", "AAGTAG", "AAGTTA", "AAGTTT", "AAGTTC", "AAGTTG", "AAGTCA", "A
"AAGTGA", "AAGTGT", "AAGTGC", "AAGTGG", "AAGCAA", "AAGCAT", "AAGCAC", "AAGCAG", "A
"AAGCTG", "AAGCCA", "AAGCCT", "AAGCCC", "AAGCCG", "AAGCGA", "AAGCGT", "AAGCGC", "A
"AAGGAC", "AAGGAG", "AAGGTA", "AAGGTT", "AAGGTC", "AAGGTG", "AAGGCA", "AAGGCT", "A
"AAGGGT", "AAGGGC", "AAGGGG", "ATAAAA", "ATAAAT", "ATAAAC", "ATAAAG", "ATAATA", "A
"ATAACA", "ATAACT", "ATAACC", "ATAACG", "ATAAGA", "ATAAGT", "ATAAGC", "ATAAGG", "A
"ATATAG", "ATATTA", "ATATTT", "ATATTC", "ATATTG", "ATATCA", "ATATCT", "ATATCC", "A
"ATATGC", "ATATGG", "ATACAA", "ATACAT", "ATACAC", "ATACAG", "ATACTA", "ATACTT", "A
"ATACCT", "ATACCC", "ATACCG", "ATACGA", "ATACGT", "ATACGC", "ATACGG", "ATAGAA", "A
"ATAGTA", "ATAGTT", "ATAGTC", "ATAGTG", "ATAGCA", "ATAGCT", "ATAGCC", "ATAGCG", "A
"ATAGGG", "ATTAAA", "ATTAAT", "ATTAAC", "ATTAAG", "ATTATA", "ATTATT", "ATTATC", "A
"ATTACC", "ATTACG", "ATTAGA", "ATTAGT", "ATTAGC", "ATTAGG", "ATTTAA", "ATTTAT", "A
"ATTTTT", "ATTTTC", "ATTTTG", "ATTTCA", "ATTTCT", "ATTTCC", "ATTTCG", "ATTTGA", "A
"ATTCAA", "ATTCAT", "ATTCAC", "ATTCAG", "ATTCTA", "ATTCTT", "ATTCTC", "ATTCTG", "A
"ATTCCG", "ATTCGA", "ATTCGT", "ATTCGC", "ATTCGG", "ATTGAA", "ATTGAT", "ATTGAC", "A
"ATTGTC", "ATTGTG", "ATTGCA", "ATTGCT", "ATTGCC", "ATTGCG", "ATTGGA", "ATTGGT", "A
"ATCAAT", "ATCAAC", "ATCAAG", "ATCATA", "ATCATT", "ATCATC", "ATCATG", "ATCACA", "A
```

"ATCAGA", "ATCAGT", "ATCAGC", "ATCAGG", "ATCTAA", "ATCTAT", "ATCTAC", "ATCTAG", "A
"ATCTTG", "ATCTCA", "ATCTCT", "ATCTCC", "ATCTCG", "ATCTGA", "ATCTGT", "ATCTGC", "A
"ATCCAC", "ATCCAG", "ATCCTA", "ATCCTT", "ATCCTC", "ATCCTG", "ATCCCA", "ATCCCT", "A
"ATCCGT", "ATCCGC", "ATCCGG", "ATCGAA", "ATCGAT", "ATCGAC", "ATCGAG", "ATCGTA", "A
"ATCGCA", "ATCGCT", "ATCGCC", "ATCGCG", "ATCGGA", "ATCGGT", "ATCGGC", "ATCGGG", "A
"ATGAAG", "ATGATA", "ATGATT", "ATGATC", "ATGATG", "ATGACA", "ATGACT", "ATGACC", "A
"ATGAGC", "ATGAGG", "ATGTAA", "ATGTAT", "ATGTAC", "ATGTAG", "ATGTTA", "ATGTTT", "A
"ATGTCT", "ATGTCC", "ATGTCG", "ATGTGA", "ATGTGT", "ATGTGC", "ATGTGG", "ATGCAA", "A
"ATGCTA", "ATGCTT", "ATGCTC", "ATGCTG", "ATGCCA", "ATGCCT", "ATGCCC", "ATGCCG", "A
"ATGCGG", "ATGGAA", "ATGGAT", "ATGGAC", "ATGGAG", "ATGGTA", "ATGGTT", "ATGGTC", "A
"ATGGCC", "ATGGCG", "ATGGGA", "ATGGGT", "ATGGGC", "ATGGGG", "ACAAAA", "ACAAAT", "A
"ACAATT", "ACAATC", "ACAATG", "ACAACA", "ACAACT", "ACAACC", "ACAACG", "ACAAGA", "A
"ACATAA", "ACATAT", "ACATAC", "ACATAG", "ACATTA", "ACATTT", "ACATTC", "ACATTG", "A
"ACATCG", "ACATGA", "ACATGT", "ACATGC", "ACATGG", "ACACAA", "ACACAT", "ACACAC", "A
"ACACTC", "ACACTG", "ACACCA", "ACACCT", "ACACCC", "ACACCG", "ACACGA", "ACACGT", "A
"ACAGAT", "ACAGAC", "ACAGAG", "ACAGTA", "ACAGTT", "ACAGTC", "ACAGTG", "ACAGCA", "A
"ACAGGA", "ACAGGT", "ACAGGC", "ACAGGG", "ACTAAA", "ACTAAT", "ACTAAC", "ACTAAG", "A
"ACTATG", "ACTACA", "ACTACT", "ACTACC", "ACTACG", "ACTAGA", "ACTAGT", "ACTAGC", "A
"ACTTAC", "ACTTAG", "ACTTTA", "ACTTTT", "ACTTTC", "ACTTTG", "ACTTCA", "ACTTCT", "A
"ACTTGT", "ACTTGC", "ACTTGG", "ACTCAA", "ACTCAT", "ACTCAC", "ACTCAG", "ACTCTA", "A
"ACTCCA", "ACTCCT", "ACTCCC", "ACTCCG", "ACTCGA", "ACTCGT", "ACTCGC", "ACTCGG", "A
"ACTGAG", "ACTGTA", "ACTGTT", "ACTGTC", "ACTGTG", "ACTGCA", "ACTGCT", "ACTGCC", "A
"ACTGGC", "ACTGGG", "ACCAAA", "ACCAAT", "ACCAAC", "ACCAAG", "ACCATA", "ACCATT", "A
"ACCACT", "ACCACC", "ACCACG", "ACCAGA", "ACCAGT", "ACCAGC", "ACCAGG", "ACCTAA", "A
"ACCTTA", "ACCTTT", "ACCTTC", "ACCTTG", "ACCTCA", "ACCTCT", "ACCTCC", "ACCTCG", "A
"ACCTGG", "ACCCAA", "ACCCAT", "ACCCAC", "ACCCAG", "ACCCTA", "ACCCTT", "ACCCTC", "A
"ACCCCC", "ACCCCG", "ACCCGA", "ACCCGT", "ACCCGC", "ACCCGG", "ACCGAA", "ACCGAT", "A
"ACCGTT", "ACCGTC", "ACCGTG", "ACCGCA", "ACCGCT", "ACCGCC", "ACCGCG", "ACCGGA", "A
"ACGAAA", "ACGAAT", "ACGAAC", "ACGAAG", "ACGATA", "ACGATT", "ACGATC", "ACGATG", "A
"ACGACG", "ACGAGA", "ACGAGT", "ACGAGC", "ACGAGG", "ACGTAA", "ACGTAT", "ACGTAC", "A
"ACGTTC", "ACGTTG", "ACGTCA", "ACGTCT", "ACGTCC", "ACGTCG", "ACGTGA", "ACGTGT", "A
"ACGCAT", "ACGCAC", "ACGCAG", "ACGCTA", "ACGCTT", "ACGCTC", "ACGCTG", "ACGCCA", "A
"ACGCGA", "ACGCGT", "ACGCGC", "ACGCGG", "ACGGAA", "ACGGAT", "ACGGAC", "ACGGAG", "A
"ACGGTG", "ACGGCA", "ACGGCT", "ACGGCC", "ACGGCG", "ACGGGA", "ACGGGT", "ACGGGC", "A
"AGAAAC", "AGAAAG", "AGAATA", "AGAATT", "AGAATC", "AGAATG", "AGAACA", "AGAACT", "A
"AGAAGT", "AGAAGC", "AGAAGG", "AGATAA", "AGATAT", "AGATAC", "AGATAG", "AGATTA", "A
"AGATCA", "AGATCT", "AGATCC", "AGATCG", "AGATGA", "AGATGT", "AGATGC", "AGATGG", "A
"AGACAG", "AGACTA", "AGACTT", "AGACTC", "AGACTG", "AGACCA", "AGACCT", "AGACCC", "A

"AGACGC", "AGACGG", "AGAGAA", "AGAGAT", "AGAGAC", "AGAGAG", "AGAGTA", "AGAGTT", "A
"AGAGCT", "AGAGCC", "AGAGCG", "AGAGGA", "AGAGGT", "AGAGGC", "AGAGGG", "AGTAAA", "A
"AGTATA", "AGTATT", "AGTATC", "AGTATG", "AGTACA", "AGTACT", "AGTACC", "AGTACG", "A
"AGTAGG", "AGTTAA", "AGTTAT", "AGTTAC", "AGTTAG", "AGTTTA", "AGTTTT", "AGTTTC", "A
"AGTTCC", "AGTTCG", "AGTTGA", "AGTTGT", "AGTTGC", "AGTTGG", "AGTCAA", "AGTCAT", "A
"AGTCTT", "AGTCTC", "AGTCTG", "AGTCCA", "AGTCCT", "AGTCCC", "AGTCCG", "AGTCGA", "A
"AGTGAA", "AGTGAT", "AGTGAC", "AGTGAG", "AGTGTA", "AGTGTT", "AGTGTC", "AGTGTG", "A
"AGTGCG", "AGTGGA", "AGTGGT", "AGTGGC", "AGTGGG", "AGCAAA", "AGCAAT", "AGCAAC", "A
"AGCATC", "AGCATG", "AGCACA", "AGCACT", "AGCACC", "AGCACG", "AGCAGA", "AGCAGT", "A
"AGCTAT", "AGCTAC", "AGCTAG", "AGCTTA", "AGCTTT", "AGCTTC", "AGCTTG", "AGCTCA", "A
"AGCTGA", "AGCTGT", "AGCTGC", "AGCTGG", "AGCCAA", "AGCCAT", "AGCCAC", "AGCCAG", "A
"AGCCTG", "AGCCCA", "AGCCCT", "AGCCCC", "AGCCCG", "AGCCGA", "AGCCGT", "AGCCGC", "A
"AGCGAC", "AGCGAG", "AGCGTA", "AGCGTT", "AGCGTC", "AGCGTG", "AGCGCA", "AGCGCT", "A
"AGCGGT", "AGCGGC", "AGCGGG", "AGGAAA", "AGGAAT", "AGGAAC", "AGGAAG", "AGGATA", "A
"AGGACA", "AGGACT", "AGGACC", "AGGACG", "AGGAGA", "AGGAGT", "AGGAGC", "AGGAGG", "A
"AGGTAG", "AGGTTA", "AGGTTT", "AGGTTC", "AGGTTG", "AGGTCA", "AGGTCT", "AGGTCC", "A
"AGGTGC", "AGGTGG", "AGGCAA", "AGGCAT", "AGGCAC", "AGGCAG", "AGGCTA", "AGGCTT", "A
"AGGCCT", "AGGCCC", "AGGCCG", "AGGCGA", "AGGCGT", "AGGCGC", "AGGCGG", "AGGGAA", "A
"AGGGTA", "AGGGTT", "AGGGTC", "AGGGTG", "AGGGCA", "AGGGCT", "AGGGCC", "AGGGCG", "A
"AGGGGG", "TAAAAA", "TAAAAT", "TAAAAC", "TAAAAG", "TAAATA", "TAAATT", "TAAATC", "T
"TAAACC", "TAAACG", "TAAAGA", "TAAAGT", "TAAAGC", "TAAAGG", "TAATAA", "TAATAT", "T
"TAATTT", "TAATTC", "TAATTG", "TAATCA", "TAATCT", "TAATCC", "TAATCG", "TAATGA", "T
"TAACAA", "TAACAT", "TAACAC", "TAACAG", "TAACTA", "TAACTT", "TAACTC", "TAACTG", "T
"TAACCG", "TAACGA", "TAACGT", "TAACGC", "TAACGG", "TAAGAA", "TAAGAT", "TAAGAC", "T
"TAAGTC", "TAAGTG", "TAAGCA", "TAAGCT", "TAAGCC", "TAAGCG", "TAAGGA", "TAAGGT", "T
"TATAAT", "TATAAC", "TATAAG", "TATATA", "TATATT", "TATATC", "TATATG", "TATACA", "T
"TATAGA", "TATAGT", "TATAGC", "TATAGG", "TATTAA", "TATTAT", "TATTAC", "TATTAG", "T
"TATTTG", "TATTCA", "TATTCT", "TATTCC", "TATTCG", "TATTGA", "TATTGT", "TATTGC", "T
"TATCAC", "TATCAG", "TATCTA", "TATCTT", "TATCTC", "TATCTG", "TATCCA", "TATCCT", "T
"TATCGT", "TATCGC", "TATCGG", "TATGAA", "TATGAT", "TATGAC", "TATGAG", "TATGTA", "T
"TATGCA", "TATGCT", "TATGCC", "TATGCG", "TATGGA", "TATGGT", "TATGGC", "TATGGG", "T
"TACAAG", "TACATA", "TACATT", "TACATC", "TACATG", "TACACA", "TACACT", "TACACC", "T
"TACAGC", "TACAGG", "TACTAA", "TACTAT", "TACTAC", "TACTAG", "TACTTA", "TACTTT", "T
"TACTCT", "TACTCC", "TACTCG", "TACTGA", "TACTGT", "TACTGC", "TACTGG", "TACCAA", "T
"TACCTA", "TACCTT", "TACCTC", "TACCTG", "TACCCA", "TACCCT", "TACCCC", "TACCCG", "T
"TACCGG", "TACGAA", "TACGAT", "TACGAC", "TACGAG", "TACGTA", "TACGTT", "TACGTC", "T
"TACGCC", "TACGCG", "TACGGA", "TACGGT", "TACGGC", "TACGGG", "TAGAAA", "TAGAAT", "T
"TAGATT", "TAGATC", "TAGATG", "TAGACA", "TAGACT", "TAGACC", "TAGACG", "TAGAGA", "T

"TAGTAA", "TAGTAT", "TAGTAC", "TAGTAG", "TAGTTA", "TAGTTT", "TAGTTC", "TAGTTG", "T
"TAGTCG", "TAGTGA", "TAGTGT", "TAGTGC", "TAGTGG", "TAGCAA", "TAGCAT", "TAGCAC", "T
"TAGCTC", "TAGCTG", "TAGCCA", "TAGCCT", "TAGCCC", "TAGCCG", "TAGCGA", "TAGCGT", "T
"TAGGAT", "TAGGAC", "TAGGAG", "TAGGTA", "TAGGTT", "TAGGTC", "TAGGTG", "TAGGCA", "T
"TAGGGA", "TAGGGT", "TAGGGC", "TAGGGG", "TTAAAA", "TTAAAT", "TTAAAC", "TTAAAG", "T
"TTAATG", "TTAACA", "TTAACT", "TTAACC", "TTAACG", "TTAAGA", "TTAAGT", "TTAAGC", "T
"TTATAC", "TTATAG", "TTATTA", "TTATTT", "TTATTC", "TTATTG", "TTATCA", "TTATCT", "T
"TTATGT", "TTATGC", "TTATGG", "TTACAA", "TTACAT", "TTACAC", "TTACAG", "TTACTA", "T
"TTACCA", "TTACCT", "TTACCC", "TTACCG", "TTACGA", "TTACGT", "TTACGC", "TTACGG", "T
"TTAGAG", "TTAGTA", "TTAGTT", "TTAGTC", "TTAGTG", "TTAGCA", "TTAGCT", "TTAGCC", "T
"TTAGGC", "TTAGGG", "TTTAAA", "TTTAAT", "TTTAAC", "TTTAAG", "TTTATA", "TTTATT", "T
"TTTACT", "TTTACC", "TTTACG", "TTTAGA", "TTTAGT", "TTTAGC", "TTTAGG", "TTTTAA", "T
"TTTTTA", "TTTTTT", "TTTTTC", "TTTTTG", "TTTTCA", "TTTTCT", "TTTTCC", "TTTTCG", "T
"TTTTGG", "TTTCAA", "TTTCAT", "TTTCAC", "TTTCAG", "TTTCTA", "TTTCTT", "TTTCTC", "T
"TTTCCC", "TTTCCG", "TTTCGA", "TTTCGT", "TTTCGC", "TTTCGG", "TTTGAA", "TTTGAT", "T
"TTTGTT", "TTTGTC", "TTTGTG", "TTTGCA", "TTTGCT", "TTTGCC", "TTTGCG", "TTTGGA", "T
"TTCAAA", "TTCAAT", "TTCAAC", "TTCAAG", "TTCATA", "TTCATT", "TTCATC", "TTCATG", "T
"TTCACG", "TTCAGA", "TTCAGT", "TTCAGC", "TTCAGG", "TTCTAA", "TTCTAT", "TTCTAC", "T
"TTCTTC", "TTCTTG", "TTCTCA", "TTCTCT", "TTCTCC", "TTCTCG", "TTCTGA", "TTCTGT", "T
"TTCCAT", "TTCCAC", "TTCCAG", "TTCCTA", "TTCCTT", "TTCCTC", "TTCCTG", "TTCCCA", "T
"TTCCGA", "TTCCGT", "TTCCGC", "TTCCGG", "TTCGAA", "TTCGAT", "TTCGAC", "TTCGAG", "T
"TTCGTG", "TTCGCA", "TTCGCT", "TTCGCC", "TTCGCG", "TTCGGA", "TTCGGT", "TTCGGC", "T
"TTGAAC", "TTGAAG", "TTGATA", "TTGATT", "TTGATC", "TTGATG", "TTGACA", "TTGACT", "T
"TTGAGT", "TTGAGC", "TTGAGG", "TTGTAA", "TTGTAT", "TTGTAC", "TTGTAG", "TTGTTA", "T
"TTGTCA", "TTGTCT", "TTGTCC", "TTGTCG", "TTGTGA", "TTGTGT", "TTGTGC", "TTGTGG", "T
"TTGCAG", "TTGCTA", "TTGCTT", "TTGCTC", "TTGCTG", "TTGCCA", "TTGCCT", "TTGCCC", "T
"TTGCGC", "TTGCGG", "TTGGAA", "TTGGAT", "TTGGAC", "TTGGAG", "TTGGTA", "TTGGTT", "T
"TTGGCT", "TTGGCC", "TTGGCG", "TTGGGA", "TTGGGT", "TTGGGC", "TTGGGG", "TCAAAA", "T
"TCAATA", "TCAATT", "TCAATC", "TCAATG", "TCAACA", "TCAACT", "TCAACC", "TCAACG", "T
"TCAAGG", "TCATAA", "TCATAT", "TCATAC", "TCATAG", "TCATTA", "TCATTT", "TCATTC", "T
"TCATCC", "TCATCG", "TCATGA", "TCATGT", "TCATGC", "TCATGG", "TCACAA", "TCACAT", "T
"TCACTT", "TCACTC", "TCACTG", "TCACCA", "TCACCT", "TCACCC", "TCACCG", "TCACGA", "T
"TCAGAA", "TCAGAT", "TCAGAC", "TCAGAG", "TCAGTA", "TCAGTT", "TCAGTC", "TCAGTG", "T
"TCAGCG", "TCAGGA", "TCAGGT", "TCAGGC", "TCAGGG", "TCTAAA", "TCTAAT", "TCTAAC", "T
"TCTATC", "TCTATG", "TCTACA", "TCTACT", "TCTACC", "TCTACG", "TCTAGA", "TCTAGT", "T
"TCTTAT", "TCTTAC", "TCTTAG", "TCTTTA", "TCTTTT", "TCTTTC", "TCTTTG", "TCTTCA", "T
"TCTTGA", "TCTTGT", "TCTTGC", "TCTTGG", "TCTCAA", "TCTCAT", "TCTCAC", "TCTCAG", "T
"TCTCTG", "TCTCCA", "TCTCCT", "TCTCCC", "TCTCCG", "TCTCGA", "TCTCGT", "TCTCGC", "T

"TCTGAC", "TCTGAG", "TCTGTA", "TCTGTT", "TCTGTC", "TCTGTG", "TCTGCA", "TCTGCT", "T
"TCTGGT", "TCTGGC", "TCTGGG", "TCCAAA", "TCCAAT", "TCCAAC", "TCCAAG", "TCCATA", "T
"TCCACA", "TCCACT", "TCCACC", "TCCACG", "TCCAGA", "TCCAGT", "TCCAGC", "TCCAGG", "T
"TCCTAG", "TCCTTA", "TCCTTT", "TCCTTC", "TCCTTG", "TCCTCA", "TCCTCT", "TCCTCC", "T
"TCCTGC", "TCCTGG", "TCCCAA", "TCCCAT", "TCCCAC", "TCCCAG", "TCCCTA", "TCCCTT", "T
"TCCCCT", "TCCCCC", "TCCCCG", "TCCCGA", "TCCCGT", "TCCCGC", "TCCCGG", "TCCGAA", "T
"TCCGTA", "TCCGTT", "TCCGTC", "TCCGTG", "TCCGCA", "TCCGCT", "TCCGCC", "TCCGCG", "T
"TCCGGG", "TCGAAA", "TCGAAT", "TCGAAC", "TCGAAG", "TCGATA", "TCGATT", "TCGATC", "T
"TCGACC", "TCGACG", "TCGAGA", "TCGAGT", "TCGAGC", "TCGAGG", "TCGTAA", "TCGTAT", "T
"TCGTTT", "TCGTTC", "TCGTTG", "TCGTCA", "TCGTCT", "TCGTCC", "TCGTCG", "TCGTGA", "T
"TCGCAA", "TCGCAT", "TCGCAC", "TCGCAG", "TCGCTA", "TCGCTT", "TCGCTC", "TCGCTG", "T
"TCGCCG", "TCGCGA", "TCGCGT", "TCGCGC", "TCGCGG", "TCGGAA", "TCGGAT", "TCGGAC", "T
"TCGGTC", "TCGGTG", "TCGGCA", "TCGGCT", "TCGGCC", "TCGGCG", "TCGGGA", "TCGGGT", "T
"TGAAAT", "TGAAAC", "TGAAAG", "TGAATA", "TGAATT", "TGAATC", "TGAATG", "TGAACA", "T
"TGAAGA", "TGAAGT", "TGAAGC", "TGAAGG", "TGATAA", "TGATAT", "TGATAC", "TGATAG", "T
"TGATTG", "TGATCA", "TGATCT", "TGATCC", "TGATCG", "TGATGA", "TGATGT", "TGATGC", "T
"TGACAC", "TGACAG", "TGACTA", "TGACTT", "TGACTC", "TGACTG", "TGACCA", "TGACCT", "T
"TGACGT", "TGACGC", "TGACGG", "TGAGAA", "TGAGAT", "TGAGAC", "TGAGAG", "TGAGTA", "T
"TGAGCA", "TGAGCT", "TGAGCC", "TGAGCG", "TGAGGA", "TGAGGT", "TGAGGC", "TGAGGG", "T
"TGTAAG", "TGTATA", "TGTATT", "TGTATC", "TGTATG", "TGTACA", "TGTACT", "TGTACC", "T
"TGTAGC", "TGTAGG", "TGTTAA", "TGTTAT", "TGTTAC", "TGTTAG", "TGTTTA", "TGTTTT", "T
"TGTTCT", "TGTTCC", "TGTTCG", "TGTTGA", "TGTTGT", "TGTTGC", "TGTTGG", "TGTCAA", "T
"TGTCTA", "TGTCTT", "TGTCTC", "TGTCTG", "TGTCCA", "TGTCCT", "TGTCCC", "TGTCCG", "T
"TGTCGG", "TGTGAA", "TGTGAT", "TGTGAC", "TGTGAG", "TGTGTA", "TGTGTT", "TGTGTC", "T
"TGTGCC", "TGTGCG", "TGTGGA", "TGTGGT", "TGTGGC", "TGTGGG", "TGCAAA", "TGCAAT", "T
"TGCATT", "TGCATC", "TGCATG", "TGCACA", "TGCACT", "TGCACC", "TGCACG", "TGCAGA", "T
"TGCTAA", "TGCTAT", "TGCTAC", "TGCTAG", "TGCTTA", "TGCTTT", "TGCTTC", "TGCTTG", "T
"TGCTCG", "TGCTGA", "TGCTGT", "TGCTGC", "TGCTGG", "TGCCAA", "TGCCAT", "TGCCAC", "T
"TGCCTC", "TGCCTG", "TGCCCA", "TGCCCT", "TGCCCC", "TGCCCG", "TGCCGA", "TGCCGT", "T
"TGCGAT", "TGCGAC", "TGCGAG", "TGCGTA", "TGCGTT", "TGCGTC", "TGCGTG", "TGCGCA", "T
"TGCGGA", "TGCGGT", "TGCGGC", "TGCGGG", "TGGAAA", "TGGAAT", "TGGAAC", "TGGAAG", "T
"TGGATG", "TGGACA", "TGGACT", "TGGACC", "TGGACG", "TGGAGA", "TGGAGT", "TGGAGC", "T
"TGGTAC", "TGGTAG", "TGGTTA", "TGGTTT", "TGGTTC", "TGGTTG", "TGGTCA", "TGGTCT", "T
"TGGTGT", "TGGTGC", "TGGTGG", "TGGCAA", "TGGCAT", "TGGCAC", "TGGCAG", "TGGCTA", "T
"TGGCCA", "TGGCCT", "TGGCCC", "TGGCCG", "TGGCGA", "TGGCGT", "TGGCGC", "TGGCGG", "T
"TGGGAG", "TGGGTA", "TGGGTT", "TGGGTC", "TGGGTG", "TGGGCA", "TGGGCT", "TGGGCC", "T
"TGGGGC", "TGGGGG", "CAAAAA", "CAAAAT", "CAAAAC", "CAAAAG", "CAAATA", "CAAATT", "C
"CAAACT", "CAAACC", "CAAACG", "CAAAGA", "CAAAGT", "CAAAGC", "CAAAGG", "CAATAA", "C

"CAATTA", "CAATTT", "CAATTC", "CAATTG", "CAATCA", "CAATCT", "CAATCC", "CAATCG", "C
"CAATGG", "CAACAA", "CAACAT", "CAACAC", "CAACAG", "CAACTA", "CAACTT", "CAACTC", "C
"CAACCC", "CAACCG", "CAACGA", "CAACGT", "CAACGC", "CAACGG", "CAAGAA", "CAAGAT", "C
"CAAGTT", "CAAGTC", "CAAGTG", "CAAGCA", "CAAGCT", "CAAGCC", "CAAGCG", "CAAGGA", "C
"CATAAA", "CATAAT", "CATAAC", "CATAAG", "CATATA", "CATATT", "CATATC", "CATATG", "C
"CATACG", "CATAGA", "CATAGT", "CATAGC", "CATAGG", "CATTAA", "CATTAT", "CATTAC", "C
"CATTTC", "CATTTG", "CATTCA", "CATTCT", "CATTCC", "CATTCG", "CATTGA", "CATTGT", "C
"CATCAT", "CATCAC", "CATCAG", "CATCTA", "CATCTT", "CATCTC", "CATCTG", "CATCCA", "C
"CATCGA", "CATCGT", "CATCGC", "CATCGG", "CATGAA", "CATGAT", "CATGAC", "CATGAG", "C
"CATGTG", "CATGCA", "CATGCT", "CATGCC", "CATGCG", "CATGGA", "CATGGT", "CATGGC", "C
"CACAAC", "CACAAG", "CACATA", "CACATT", "CACATC", "CACATG", "CACACA", "CACACT", "C
"CACAGT", "CACAGC", "CACAGG", "CACTAA", "CACTAT", "CACTAC", "CACTAG", "CACTTA", "C
"CACTCA", "CACTCT", "CACTCC", "CACTCG", "CACTGA", "CACTGT", "CACTGC", "CACTGG", "C
"CACCAG", "CACCTA", "CACCTT", "CACCTC", "CACCTG", "CACCCA", "CACCCT", "CACCCC", "C
"CACCGC", "CACCGG", "CACGAA", "CACGAT", "CACGAC", "CACGAG", "CACGTA", "CACGTT", "C
"CACGCT", "CACGCC", "CACGCG", "CACGGA", "CACGGT", "CACGGC", "CACGGG", "CAGAAA", "C
"CAGATA", "CAGATT", "CAGATC", "CAGATG", "CAGACA", "CAGACT", "CAGACC", "CAGACG", "C
"CAGAGG", "CAGTAA", "CAGTAT", "CAGTAC", "CAGTAG", "CAGTTA", "CAGTTT", "CAGTTC", "C
"CAGTCC", "CAGTCG", "CAGTGA", "CAGTGT", "CAGTGC", "CAGTGG", "CAGCAA", "CAGCAT", "C
"CAGCTT", "CAGCTC", "CAGCTG", "CAGCCA", "CAGCCT", "CAGCCC", "CAGCCG", "CAGCGA", "C
"CAGGAA", "CAGGAT", "CAGGAC", "CAGGAG", "CAGGTA", "CAGGTT", "CAGGTC", "CAGGTG", "C
"CAGGCG", "CAGGGA", "CAGGGT", "CAGGGC", "CAGGGG", "CTAAAA", "CTAAAT", "CTAAAC", "C
"CTAATC", "CTAATG", "CTAACA", "CTAACT", "CTAACC", "CTAACG", "CTAAGA", "CTAAGT", "C
"CTATAT", "CTATAC", "CTATAG", "CTATTA", "CTATTT", "CTATTC", "CTATTG", "CTATCA", "C
"CTATGA", "CTATGT", "CTATGC", "CTATGG", "CTACAA", "CTACAT", "CTACAC", "CTACAG", "C
"CTACTG", "CTACCA", "CTACCT", "CTACCC", "CTACCG", "CTACGA", "CTACGT", "CTACGC", "C
"CTAGAC", "CTAGAG", "CTAGTA", "CTAGTT", "CTAGTC", "CTAGTG", "CTAGCA", "CTAGCT", "C
"CTAGGT", "CTAGGC", "CTAGGG", "CTTAAA", "CTTAAT", "CTTAAC", "CTTAAG", "CTTATA", "C
"CTTACA", "CTTACT", "CTTACC", "CTTACG", "CTTAGA", "CTTAGT", "CTTAGC", "CTTAGG", "C
"CTTTAG", "CTTTTA", "CTTTTT", "CTTTTC", "CTTTTG", "CTTTCA", "CTTTCT", "CTTTCC", "C
"CTTTGC", "CTTTGG", "CTTCAA", "CTTCAT", "CTTCAC", "CTTCAG", "CTTCTA", "CTTCTT", "C
"CTTCCT", "CTTCCC", "CTTCCG", "CTTCGA", "CTTCGT", "CTTCGC", "CTTCGG", "CTTGAA", "C
"CTTGTA", "CTTGTT", "CTTGTC", "CTTGTG", "CTTGCA", "CTTGCT", "CTTGCC", "CTTGCG", "C
"CTTGGG", "CTCAAA", "CTCAAT", "CTCAAC", "CTCAAG", "CTCATA", "CTCATT", "CTCATC", "C
"CTCACC", "CTCACG", "CTCAGA", "CTCAGT", "CTCAGC", "CTCAGG", "CTCTAA", "CTCTAT", "C
"CTCTTT", "CTCTTC", "CTCTTG", "CTCTCA", "CTCTCT", "CTCTCC", "CTCTCG", "CTCTGA", "C
"CTCCAA", "CTCCAT", "CTCCAC", "CTCCAG", "CTCCTA", "CTCCTT", "CTCCTC", "CTCCTG", "C
"CTCCCG", "CTCCGA", "CTCCGT", "CTCCGC", "CTCCGG", "CTCGAA", "CTCGAT", "CTCGAC", "C

"CTCGTC", "CTCGTG", "CTCGCA", "CTCGCT", "CTCGCC", "CTCGCG", "CTCGGA", "CTCGGT", "C
"CTGAAT", "CTGAAC", "CTGAAG", "CTGATA", "CTGATT", "CTGATC", "CTGATG", "CTGACA", "C
"CTGAGA", "CTGAGT", "CTGAGC", "CTGAGG", "CTGTAA", "CTGTAT", "CTGTAC", "CTGTAG", "C
"CTGTTG", "CTGTCA", "CTGTCT", "CTGTCC", "CTGTCG", "CTGTGA", "CTGTGT", "CTGTGC", "C
"CTGCAC", "CTGCAG", "CTGCTA", "CTGCTT", "CTGCTC", "CTGCTG", "CTGCCA", "CTGCCT", "C
"CTGCGT", "CTGCGC", "CTGCGG", "CTGGAA", "CTGGAT", "CTGGAC", "CTGGAG", "CTGGTA", "C
"CTGGCA", "CTGGCT", "CTGGCC", "CTGGCG", "CTGGGA", "CTGGGT", "CTGGGC", "CTGGGG", "C
"CCAAAG", "CCAATA", "CCAATT", "CCAATC", "CCAATG", "CCAACA", "CCAACT", "CCAACC", "C
"CCAAGC", "CCAAGG", "CCATAA", "CCATAT", "CCATAC", "CCATAG", "CCATTA", "CCATTT", "C
"CCATCT", "CCATCC", "CCATCG", "CCATGA", "CCATGT", "CCATGC", "CCATGG", "CCACAA", "C
"CCACTA", "CCACTT", "CCACTC", "CCACTG", "CCACCA", "CCACCT", "CCACCC", "CCACCG", "C
"CCACGG", "CCAGAA", "CCAGAT", "CCAGAC", "CCAGAG", "CCAGTA", "CCAGTT", "CCAGTC", "C
"CCAGCC", "CCAGCG", "CCAGGA", "CCAGGT", "CCAGGC", "CCAGGG", "CCTAAA", "CCTAAT", "C
"CCTATT", "CCTATC", "CCTATG", "CCTACA", "CCTACT", "CCTACC", "CCTACG", "CCTAGA", "C
"CCTTAA", "CCTTAT", "CCTTAC", "CCTTAG", "CCTTTA", "CCTTTT", "CCTTTC", "CCTTTG", "C
"CCTTCG", "CCTTGA", "CCTTGT", "CCTTGC", "CCTTGG", "CCTCAA", "CCTCAT", "CCTCAC", "C
"CCTCTC", "CCTCTG", "CCTCCA", "CCTCCT", "CCTCCC", "CCTCCG", "CCTCGA", "CCTCGT", "C
"CCTGAT", "CCTGAC", "CCTGAG", "CCTGTA", "CCTGTT", "CCTGTC", "CCTGTG", "CCTGCA", "C
"CCTGGA", "CCTGGT", "CCTGGC", "CCTGGG", "CCCAAA", "CCCAAT", "CCCAAC", "CCCAAG", "C
"CCCATG", "CCCACA", "CCCACT", "CCCACC", "CCCACG", "CCCAGA", "CCCAGT", "CCCAGC", "C
"CCCTAC", "CCCTAG", "CCCTTA", "CCCTTT", "CCCTTC", "CCCTTG", "CCCTCA", "CCCTCT", "C
"CCCTGT", "CCCTGC", "CCCTGG", "CCCCAA", "CCCCAT", "CCCCAC", "CCCCAG", "CCCCTA", "C
"CCCCCA", "CCCCCT", "CCCCCC", "CCCCCG", "CCCCGA", "CCCCGT", "CCCCGC", "CCCCGG", "C
"CCCGAG", "CCCGTA", "CCCGTT", "CCCGTC", "CCCGTG", "CCCGCA", "CCCGCT", "CCCGCC", "C
"CCCGGC", "CCCGGG", "CCGAAA", "CCGAAT", "CCGAAC", "CCGAAG", "CCGATA", "CCGATT", "C
"CCGACT", "CCGACC", "CCGACG", "CCGAGA", "CCGAGT", "CCGAGC", "CCGAGG", "CCGTAA", "C
"CCGTTA", "CCGTTT", "CCGTTC", "CCGTTG", "CCGTCA", "CCGTCT", "CCGTCC", "CCGTCG", "C
"CCGTGG", "CCGCAA", "CCGCAT", "CCGCAC", "CCGCAG", "CCGCTA", "CCGCTT", "CCGCTC", "C
"CCGCCC", "CCGCCG", "CCGCGA", "CCGCGT", "CCGCGC", "CCGCGG", "CCGGAA", "CCGGAT", "C
"CCGGTT", "CCGGTC", "CCGGTG", "CCGGCA", "CCGGCT", "CCGGCC", "CCGGCG", "CCGGGA", "C
"CGAAAA", "CGAAAT", "CGAAAC", "CGAAAG", "CGAATA", "CGAATT", "CGAATC", "CGAATG", "C
"CGAACG", "CGAAGA", "CGAAGT", "CGAAGC", "CGAAGG", "CGATAA", "CGATAT", "CGATAC", "C
"CGATTC", "CGATTG", "CGATCA", "CGATCT", "CGATCC", "CGATCG", "CGATGA", "CGATGT", "C
"CGACAT", "CGACAC", "CGACAG", "CGACTA", "CGACTT", "CGACTC", "CGACTG", "CGACCA", "C
"CGACGA", "CGACGT", "CGACGC", "CGACGG", "CGAGAA", "CGAGAT", "CGAGAC", "CGAGAG", "C
"CGAGTG", "CGAGCA", "CGAGCT", "CGAGCC", "CGAGCG", "CGAGGA", "CGAGGT", "CGAGGC", "C
"CGTAAC", "CGTAAG", "CGTATA", "CGTATT", "CGTATC", "CGTATG", "CGTACA", "CGTACT", "C
"CGTAGT", "CGTAGC", "CGTAGG", "CGTTAA", "CGTTAT", "CGTTAC", "CGTTAG", "CGTTTA", "C

"CGTTCA", "CGTTCT", "CGTTCC", "CGTTCG", "CGTTGA", "CGTTGT", "CGTTGC", "CGTTGG", "C
"CGTCAG", "CGTCTA", "CGTCTT", "CGTCTC", "CGTCTG", "CGTCCA", "CGTCCT", "CGTCCC", "C
"CGTCGC", "CGTCGG", "CGTGAA", "CGTGAT", "CGTGAC", "CGTGAG", "CGTGTA", "CGTGTT", "C
"CGTGCT", "CGTGCC", "CGTGCG", "CGTGGA", "CGTGGT", "CGTGGC", "CGTGGG", "CGCAAA", "C
"CGCATA", "CGCATT", "CGCATC", "CGCATG", "CGCACA", "CGCACT", "CGCACC", "CGCACG", "C
"CGCAGG", "CGCTAA", "CGCTAT", "CGCTAC", "CGCTAG", "CGCTTA", "CGCTTT", "CGCTTC", "C
"CGCTCC", "CGCTCG", "CGCTGA", "CGCTGT", "CGCTGC", "CGCTGG", "CGCCAA", "CGCCAT", "C
"CGCCTT", "CGCCTC", "CGCCTG", "CGCCCA", "CGCCCT", "CGCCCC", "CGCCCG", "CGCCGA", "C
"CGCGAA", "CGCGAT", "CGCGAC", "CGCGAG", "CGCGTA", "CGCGTT", "CGCGTC", "CGCGTG", "C
"CGCGCG", "CGCGGA", "CGCGGT", "CGCGGC", "CGCGGG", "CGGAAA", "CGGAAT", "CGGAAC", "C
"CGGATC", "CGGATG", "CGGACA", "CGGACT", "CGGACC", "CGGACG", "CGGAGA", "CGGAGT", "C
"CGGTAT", "CGGTAC", "CGGTAG", "CGGTTA", "CGGTTT", "CGGTTC", "CGGTTG", "CGGTCA", "C
"CGGTGA", "CGGTGT", "CGGTGC", "CGGTGG", "CGGCAA", "CGGCAT", "CGGCAC", "CGGCAG", "C
"CGGCTG", "CGGCCA", "CGGCCT", "CGGCCC", "CGGCCG", "CGGCGA", "CGGCGT", "CGGCGC", "C
"CGGGAC", "CGGGAG", "CGGGTA", "CGGGTT", "CGGGTC", "CGGGTG", "CGGGCA", "CGGGCT", "C
"CGGGGT", "CGGGGC", "CGGGGG", "GAAAAA", "GAAAAT", "GAAAAC", "GAAAAG", "GAAATA", "C
"GAAACA", "GAAACT", "GAAACC", "GAAACG", "GAAAGA", "GAAAGT", "GAAAGC", "GAAAGG", "C
"GAATAG", "GAATTA", "GAATTT", "GAATTC", "GAATTG", "GAATCA", "GAATCT", "GAATCC", "C
"GAATGC", "GAATGG", "GAACAA", "GAACAT", "GAACAC", "GAACAG", "GAACTA", "GAACTT", "C
"GAACCT", "GAACCC", "GAACCG", "GAACGA", "GAACGT", "GAACGC", "GAACGG", "GAAGAA", "C
"GAAGTA", "GAAGTT", "GAAGTC", "GAAGTG", "GAAGCA", "GAAGCT", "GAAGCC", "GAAGCG", "C
"GAAGGG", "GATAAA", "GATAAT", "GATAAC", "GATAAG", "GATATA", "GATATT", "GATATC", "C
"GATACC", "GATACG", "GATAGA", "GATAGT", "GATAGC", "GATAGG", "GATTAA", "GATTAT", "C
"GATTTT", "GATTTC", "GATTTG", "GATTCA", "GATTCT", "GATTCC", "GATTCG", "GATTGA", "C
"GATCAA", "GATCAT", "GATCAC", "GATCAG", "GATCTA", "GATCTT", "GATCTC", "GATCTG", "C
"GATCCG", "GATCGA", "GATCGT", "GATCGC", "GATCGG", "GATGAA", "GATGAT", "GATGAC", "C
"GATGTC", "GATGTG", "GATGCA", "GATGCT", "GATGCC", "GATGCG", "GATGGA", "GATGGT", "C
"GACAAT", "GACAAC", "GACAAG", "GACATA", "GACATT", "GACATC", "GACATG", "GACACA", "C
"GACAGA", "GACAGT", "GACAGC", "GACAGG", "GACTAA", "GACTAT", "GACTAC", "GACTAG", "C
"GACTTG", "GACTCA", "GACTCT", "GACTCC", "GACTCG", "GACTGA", "GACTGT", "GACTGC", "C
"GACCAC", "GACCAG", "GACCTA", "GACCTT", "GACCTC", "GACCTG", "GACCCA", "GACCCT", "C
"GACCGT", "GACCGC", "GACCGG", "GACGAA", "GACGAT", "GACGAC", "GACGAG", "GACGTA", "C
"GACGCA", "GACGCT", "GACGCC", "GACGCG", "GACGGA", "GACGGT", "GACGGC", "GACGGG", "C
"GAGAAG", "GAGATA", "GAGATT", "GAGATC", "GAGATG", "GAGACA", "GAGACT", "GAGACC", "C
"GAGAGC", "GAGAGG", "GAGTAA", "GAGTAT", "GAGTAC", "GAGTAG", "GAGTTA", "GAGTTT", "C
"GAGTCT", "GAGTCC", "GAGTCG", "GAGTGA", "GAGTGT", "GAGTGC", "GAGTGG", "GAGCAA", "C
"GAGCTA", "GAGCTT", "GAGCTC", "GAGCTG", "GAGCCA", "GAGCCT", "GAGCCC", "GAGCCG", "C
"GAGCGG", "GAGGAA", "GAGGAT", "GAGGAC", "GAGGAG", "GAGGTA", "GAGGTT", "GAGGTC", "C

"GAGGCC", "GAGGCG", "GAGGGA", "GAGGGT", "GAGGGC", "GAGGGG", "GTAAAA", "GTAAAT", "G
"GTAATT", "GTAATC", "GTAATG", "GTAACA", "GTAACT", "GTAACC", "GTAACG", "GTAAGA", "G
"GTATAA", "GTATAT", "GTATAC", "GTATAG", "GTATTA", "GTATTT", "GTATTC", "GTATTG", "G
"GTATCG", "GTATGA", "GTATGT", "GTATGC", "GTATGG", "GTACAA", "GTACAT", "GTACAC", "G
"GTACTC", "GTACTG", "GTACCA", "GTACCT", "GTACCC", "GTACCG", "GTACGA", "GTACGT", "G
"GTAGAT", "GTAGAC", "GTAGAG", "GTAGTA", "GTAGTT", "GTAGTC", "GTAGTG", "GTAGCA", "G
"GTAGGA", "GTAGGT", "GTAGGC", "GTAGGG", "GTTAAA", "GTTAAT", "GTTAAC", "GTTAAG", "G
"GTTATG", "GTTACA", "GTTACT", "GTTACC", "GTTACG", "GTTAGA", "GTTAGT", "GTTAGC", "G
"GTTTAC", "GTTTAG", "GTTTTA", "GTTTTT", "GTTTTC", "GTTTTG", "GTTTCA", "GTTTCT", "G
"GTTTGT", "GTTTGC", "GTTTGG", "GTTCAA", "GTTCAT", "GTTCAC", "GTTCAG", "GTTCTA", "G
"GTTCCA", "GTTCCT", "GTTCCC", "GTTCCG", "GTTCGA", "GTTCGT", "GTTCGC", "GTTCGG", "G
"GTTGAG", "GTTGTA", "GTTGTT", "GTTGTC", "GTTGTG", "GTTGCA", "GTTGCT", "GTTGCC", "G
"GTTGGC", "GTTGGG", "GTCAAA", "GTCAAT", "GTCAAC", "GTCAAG", "GTCATA", "GTCATT", "G
"GTCACT", "GTCACC", "GTCACG", "GTCAGA", "GTCAGT", "GTCAGC", "GTCAGG", "GTCTAA", "G
"GTCTTA", "GTCTTT", "GTCTTC", "GTCTTG", "GTCTCA", "GTCTCT", "GTCTCC", "GTCTCG", "G
"GTCTGG", "GTCCAA", "GTCCAT", "GTCCAC", "GTCCAG", "GTCCTA", "GTCCTT", "GTCCTC", "G
"GTCCCC", "GTCCCG", "GTCCGA", "GTCCGT", "GTCCGC", "GTCCGG", "GTCGAA", "GTCGAT", "G
"GTCGTT", "GTCGTC", "GTCGTG", "GTCGCA", "GTCGCT", "GTCGCC", "GTCGCG", "GTCGGA", "G
"GTGAAA", "GTGAAT", "GTGAAC", "GTGAAG", "GTGATA", "GTGATT", "GTGATC", "GTGATG", "G
"GTGACG", "GTGAGA", "GTGAGT", "GTGAGC", "GTGAGG", "GTGTAA", "GTGTAT", "GTGTAC", "G
"GTGTTC", "GTGTTG", "GTGTCA", "GTGTCT", "GTGTCC", "GTGTCG", "GTGTGA", "GTGTGT", "G
"GTGCAT", "GTGCAC", "GTGCAG", "GTGCTA", "GTGCTT", "GTGCTC", "GTGCTG", "GTGCCA", "G
"GTGCGA", "GTGCGT", "GTGCGC", "GTGCGG", "GTGGAA", "GTGGAT", "GTGGAC", "GTGGAG", "G
"GTGGTG", "GTGGCA", "GTGGCT", "GTGGCC", "GTGGCG", "GTGGGA", "GTGGGT", "GTGGGC", "G
"GCAAAC", "GCAAAG", "GCAATA", "GCAATT", "GCAATC", "GCAATG", "GCAACA", "GCAACT", "G
"GCAAGT", "GCAAGC", "GCAAGG", "GCATAA", "GCATAT", "GCATAC", "GCATAG", "GCATTA", "G
"GCATCA", "GCATCT", "GCATCC", "GCATCG", "GCATGA", "GCATGT", "GCATGC", "GCATGG", "G
"GCACAG", "GCACTA", "GCACTT", "GCACTC", "GCACTG", "GCACCA", "GCACCT", "GCACCC", "G
"GCACGC", "GCACGG", "GCAGAA", "GCAGAT", "GCAGAC", "GCAGAG", "GCAGTA", "GCAGTT", "G
"GCAGCT", "GCAGCC", "GCAGCG", "GCAGGA", "GCAGGT", "GCAGGC", "GCAGGG", "GCTAAA", "G
"GCTATA", "GCTATT", "GCTATC", "GCTATG", "GCTACA", "GCTACT", "GCTACC", "GCTACG", "G
"GCTAGG", "GCTTAA", "GCTTAT", "GCTTAC", "GCTTAG", "GCTTTA", "GCTTTT", "GCTTTC", "G
"GCTTCC", "GCTTCG", "GCTTGA", "GCTTGT", "GCTTGC", "GCTTGG", "GCTCAA", "GCTCAT", "G
"GCTCTT", "GCTCTC", "GCTCTG", "GCTCCA", "GCTCCT", "GCTCCC", "GCTCCG", "GCTCGA", "G
"GCTGAA", "GCTGAT", "GCTGAC", "GCTGAG", "GCTGTA", "GCTGTT", "GCTGTC", "GCTGTG", "G
"GCTGCG", "GCTGGA", "GCTGGT", "GCTGGC", "GCTGGG", "GCCAAA", "GCCAAT", "GCCAAC", "G
"GCCATC", "GCCATG", "GCCACA", "GCCACT", "GCCACC", "GCCACG", "GCCAGA", "GCCAGT", "G
"GCCTAT", "GCCTAC", "GCCTAG", "GCCTTA", "GCCTTT", "GCCTTC", "GCCTTG", "GCCTCA", "G

```
"GCCTGA", "GCCTGT", "GCCTGC", "GCCTGG", "GCCCAA", "GCCCAT", "GCCCAC", "GCCCAG", "G
"GCCCTG", "GCCCCA", "GCCCCT", "GCCCCC", "GCCCCG", "GCCCGA", "GCCCGT", "GCCCGC", "G
"GCCGAC", "GCCGAG", "GCCGTA", "GCCGTT", "GCCGTC", "GCCGTG", "GCCGCA", "GCCGCT", "G
"GCCGGT", "GCCGGC", "GCCGGG", "GCGAAA", "GCGAAT", "GCGAAC", "GCGAAG", "GCGATA", "G
"GCGACA", "GCGACT", "GCGACC", "GCGACG", "GCGAGA", "GCGAGT", "GCGAGC", "GCGAGG", "G
"GCGTAG", "GCGTTA", "GCGTTT", "GCGTTC", "GCGTTG", "GCGTCA", "GCGTCT", "GCGTCC", "G
"GCGTGC", "GCGTGG", "GCGCAA", "GCGCAT", "GCGCAC", "GCGCAG", "GCGCTA", "GCGCTT", "G
"GCGCCT", "GCGCCC", "GCGCCG", "GCGCGA", "GCGCGT", "GCGCGC", "GCGCGG", "GCGGAA", "G
"GCGGTA", "GCGGTT", "GCGGTC", "GCGGTG", "GCGGCA", "GCGGCT", "GCGGCC", "GCGGCG", "G
"GCGGGG", "GGAAAA", "GGAAAT", "GGAAAC", "GGAAAG", "GGAATA", "GGAATT", "GGAATC", "G
"GGAACC", "GGAACG", "GGAAGA", "GGAAGT", "GGAAGC", "GGAAGG", "GGATAA", "GGATAT", "G
"GGATTT", "GGATTC", "GGATTG", "GGATCA", "GGATCT", "GGATCC", "GGATCG", "GGATGA", "G
"GGACAA", "GGACAT", "GGACAC", "GGACAG", "GGACTA", "GGACTT", "GGACTC", "GGACTG", "G
"GGACCG", "GGACGA", "GGACGT", "GGACGC", "GGACGG", "GGAGAA", "GGAGAT", "GGAGAC", "G
"GGAGTC", "GGAGTG", "GGAGCA", "GGAGCT", "GGAGCC", "GGAGCG", "GGAGGA", "GGAGGT", "G
"GGTAAT", "GGTAAC", "GGTAAG", "GGTATA", "GGTATT", "GGTATC", "GGTATG", "GGTACA", "G
"GGTAGA", "GGTAGT", "GGTAGC", "GGTAGG", "GGTTAA", "GGTTAT", "GGTTAC", "GGTTAG", "G
"GGTTTG", "GGTTCA", "GGTTCT", "GGTTCC", "GGTTCG", "GGTTGA", "GGTTGT", "GGTTGC", "G
"GGTCAC", "GGTCAG", "GGTCTA", "GGTCTT", "GGTCTC", "GGTCTG", "GGTCCA", "GGTCCT", "G
"GGTCGT", "GGTCGC", "GGTCGG", "GGTGAA", "GGTGAT", "GGTGAC", "GGTGAG", "GGTGTA", "G
"GGTGCA", "GGTGCT", "GGTGCC", "GGTGCG", "GGTGGA", "GGTGGT", "GGTGGC", "GGTGGG", "G
"GGCAAG", "GGCATA", "GGCATT", "GGCATC", "GGCATG", "GGCACA", "GGCACT", "GGCACC", "G
"GGCAGC", "GGCAGG", "GGCTAA", "GGCTAT", "GGCTAC", "GGCTAG", "GGCTTA", "GGCTTT", "G
"GGCTCT", "GGCTCC", "GGCTCG", "GGCTGA", "GGCTGT", "GGCTGC", "GGCTGG", "GGCCAA", "G
"GGCCTA", "GGCCTT", "GGCCTC", "GGCCTG", "GGCCCA", "GGCCCT", "GGCCCC", "GGCCCG", "G
"GGCCGG", "GGCGAA", "GGCGAT", "GGCGAC", "GGCGAG", "GGCGTA", "GGCGTT", "GGCGTC", "G
"GGCGCC", "GGCGCG", "GGCGGA", "GGCGGT", "GGCGGC", "GGCGGG", "GGGAAA", "GGGAAT", "G
"GGGATT", "GGGATC", "GGGATG", "GGGACA", "GGGACT", "GGGACC", "GGGACG", "GGGAGA", "G
"GGGTAA", "GGGTAT", "GGGTAC", "GGGTAG", "GGGTTA", "GGGTTT", "GGGTTC", "GGGTTG", "G
"GGGTCG", "GGGTGA", "GGGTGT", "GGGTGC", "GGGTGG", "GGGCAA", "GGGCAT", "GGGCAC", "G
"GGGCTC", "GGGCTG", "GGGCCA", "GGGCCT", "GGGCCC", "GGGCCG", "GGGCGA", "GGGCGT", "G
"GGGGAT", "GGGGAC", "GGGGAG", "GGGGTA", "GGGGTT", "GGGGTC", "GGGGTG", "GGGGCA", "G
"GGGGGA", "GGGGGT", "GGGGGC", "GGGGGG"];
%}
tokens = [singlet, doublet, triplet, quartet];
existsTR = zeros(340, 16128);

speciesIndex = 0;
```

```
speciesCount = zeros(13, 1);
dashCount = 0;
for m = 1:16128
    sequence = nuc_train(m);
    l = strlength(sequence);
    for i = 1:l
        for j = 1:340
            series = extractBetween(sequence, i, i);
            if strcmp("-", series)
                dashCount = dashCount + 1;
            else
                if strlength(tokens(j)) == 1
                    if strcmp(tokens(j), series)
                        existsTR(j, m) = existsTR(j, m) + 1;
                    end
                end
                if l-i > 1 && (strlength(tokens(j)) == 2)
                    series = extractBetween(sequence, i, i+1);
                    if strcmp(tokens(j), series)
                        existsTR(j, m) = existsTR(j, m) + 1;
                    end
                end
                if (l-i > 2) && (strlength(tokens(j)) == 3)
                    series = extractBetween(sequence, i, i+2);
                    if strcmp(tokens(j), series)
                        existsTR(j, m) = existsTR(j, m) + 1;
                    end
                end
                if (l-i > 3) && (strlength(tokens(j)) == 4)
                    series = extractBetween(sequence, i, i+3);
                    if strcmp(tokens(j), series)
                        existsTR(j, m) = existsTR(j, m) + 1;
                    end
                end
            end
        end
    end
end
```

```
%{
multProbVect = zeros(84, 13);
for n = 1:13
   for o = 1:84
       multProbVect(o, n) = existsTR(o, n) / speciesCount(n);
   end
end
%}
```