Question 1:

Solution:

First, compute the BoW model for each sentence.

S1: {'sunshine': 2, 'state': 1, 'enjoy': 1}

S2: {'brown': 2, 'fox': 2, 'jump': 1, 'high': 1, 'run': 1}

S3: {'sunshine': 1, 'state': 1, 'fox': 1, 'run': 1, 'fast': 1}

Code:

```
# Computing the BoW model for each sentence

s1 = "sunshine state enjoy sunshine"

s2 = "brown fox jump high, brown fox run"

s3 = "sunshine state fox run fast"


bow_s1 = {}

bow_s2 = {}

bow_s3 = {}


for word in s1.split():

   if word in bow_s1:

      bow_s1[word] += 1

   else:

      bow_s1[word] = 1


for word in s2.split():

   if word in bow_s2:

      bow_s2[word] += 1

   else:

      bow_s2[word] = 1


for word in s3.split():
```

```
    if word in bow_s3:
        bow_s3[word] += 1
    else:
        bow_s3[word] = 1
```

Next, computing the TF model for each term in each sentence.

S1: {'sunshine': 2/3, 'state': 1/3, 'enjoy': 1/3}

S2: {'brown': 2/5, 'fox': 2/5, 'jump': 1/5, 'high': 1/5, 'run': 1/5}

S3: {'sunshine': 1/5, 'state': 1/5, 'fox': 1/5, 'run': 1/5, 'fast': 1/5}

Code:

```
# Computing the TF model for each term in each sentence

# Sentence 1
s1 = "sunshine state enjoy sunshine"
tf_s1 = {}

for word in s1.split():
    if word in tf_s1:
        tf_s1[word] += 1
    else:
        tf_s1[word] = 1

# Divide the count of each word by the total number of words in the sentence
for word in tf_s1:
    tf_s1[word] = tf_s1[word] / len(s1.split())

print(tf_s1)

# Sentence 2
s2 = "brown fox jump high, brown fox run"
```

```
tf_s2 = {}

for word in s2.split():
    if word in tf_s2:
        tf_s2[word] += 1
    else:
        tf_s2[word] = 1

# Divide the count of each word by the total number of words in the sentence
for word in tf_s2:
    tf_s2[word] = tf_s2[word] / len(s2.split())

print(tf_s2)

# Sentence 3
s3 = "sunshine state fox run fast"
tf_s3 = {}

for word in s3.split():
    if word in tf_s3:
        tf_s3[word] += 1
    else:
        tf_s3[word] = 1

# Divide the count of each word by the total number of words in the sentence
for word in tf_s3:
    tf_s3[word] = tf_s3[word] / len(s3.split())
print(tf_s3)
```

The output will be the TF model for each term in each sentence:

```
{'sunshine': 0.5, 'state': 0.25, 'enjoy': 0.25}
{'brown': 0.4, 'fox': 0.4, 'jump': 0.2, 'high': 0.2, 'run': 0.2}
```

{'sunshine': 0.2, 'state': 0.2, 'fox': 0.2, 'run': 0.2, 'fast': 0.2}

Now, computing the IDF model for each term.

IDF: {'sunshine': log(3/2), 'state': log(3/2), 'enjoy': log(3/1), 'brown': log(3/2), 'fox': log(3/2), 'jump': log(3/1), 'high': log(3/1), 'run': log(3/2), 'fast': log(3/1)}

Code:

```
# Compute the IDF model
idf = {}

for word in set(s1.split() + s2.split() + s3.split()):
    n = sum([1 for sentence in [s1, s2, s3] if word in sentence])
    idf[word] = log(3/n)

print(idf)
```

This will output the following IDF values:

```
{'sunshine': 0.4054651081081644,
 'state': 0.4054651081081644,
 'enjoy': 1.0986122886681098,
 'brown': 0.4054651081081644,
 'fox': 0.4054651081081644,
 'jump': 1.0986122886681098,
 'high': 1.0986122886681098,
 'run': 0.4054651081081644,
 'fast': 1.0986122886681098}
```

Finally, calculating the TF.IDF values for each term in each sentence.

S1: {'sunshine': (2/3) * log(3/2), 'state': (1/3) * log(3/2), 'enjoy': (1/3) * log(3/1)}

S2: {'brown': (2/5) * log(3/2), 'fox': (2/5) * log(3/2), 'jump': (1/5) * log(3/1), 'high': (1/5) * log(3/1), 'run': (1/5) * log(3/2)}

S3: {'sunshine': (1/5) * log(3/2), 'state': (1/5) * log(3/2), 'fox': (1/5) * log(3/2), 'run': (1/5) * log(3/2), 'fast': (1/5) * log(3/1)}

Code:

```
# Calculate the TF.IDF values for each term in each sentence
tfidf_s1 = {}
tfidf_s2 = {}
tfidf_s3 = {}

for word in bow_s1:
    tfidf_s1[word] = (bow_s1[word]/sum(bow_s1.values())) * idf[word]

for word in bow_s2:
    tfidf_s2[word] = (bow_s2[word]/sum(bow_s2.values())) * idf[word]

for word in bow_s3:
    tfidf_s3[word] = (bow_s3[word]/sum(bow_s3.values())) * idf[word]

print(tfidf_s1)
print(tfidf_s2)
print(tfidf_s3)
```

Question 2:

S1:

sunshine 2

state 1

enjoy 1

S3:

sunshine 1

state 1

fox 1

run 1

fast 1

Next, we need to compute the dot product of the two sentences by multiplying the corresponding term frequencies and summing the results. This is given by:

(2 * 1) + (1 * 1) = 3

We also need to compute the magnitudes of each sentence, which is given by the square root of the sum of the squares of the term frequencies.

S1: sqrt(2^2 + 1^2 + 1^2) = sqrt(6) = 2.45

S3: sqrt(1^2 + 1^2 + 1^2 + 1^2 + 1^2) = sqrt(5) = 2.24

cosine similarity = 3 / (2.45 * 2.24) = 0.52

Code:

```
from math import sqrt

# Sentence vectors

s1_vector = [0.5, 0.25, 0.25]

s3_vector = [0.2, 0.2, 0.2, 0.2, 0.2]


# Compute the dot product

dot_product = sum([s1_vector[i] * s3_vector[i] for i in range(len(s1_vector))])


# Compute the magnitudes

magnitude_s1 = sum([s1_vector[i] ** 2 for i in range(len(s1_vector))]) ** 0.5

magnitude_s3 = sum([s3_vector[i] ** 2 for i in range(len(s3_vector))]) ** 0.5


# Compute the cosine similarity

cosine_similarity = dot_product / (magnitude_s1 * magnitude_s3)


print(cosine_similarity)
```

The output of cosine similarity between S1 and S3 is:

0.52