```
from collections import Counter


emails = [
    ("Win money now", "spam"),
    ("Lowest price guaranteed", "spam"),
    ("Cheap meds available", "spam"),
    ("Hello friend how are you", "ham"),
    ("Let’s have lunch tomorrow", "ham"),
    ("Meeting schedule attached", "ham"),
    ("Win a free lottery ticket", "spam"),
    ("See you at the conference", "ham"),
    ("Project deadline reminder", "ham"),
    ("Cheap loans available", "spam"),
]


def tokenize(text):
    return text.lower().split()


spam_words, ham_words = [], []
for text, label in emails:
    if label == "spam":
        spam_words.extend(tokenize(text))
    else:
        ham_words.extend(tokenize(text))

spam_count, ham_count = Counter(spam_words), Counter(ham_words)


vocab = set(spam_words + ham_words)
V = len(vocab)

def word_prob(word, label):
    if label == "spam":
        return (spam_count[word] + 1) / (len(spam_words) + V)
    else:
        return (ham_count[word] + 1) / (len(ham_words) + V)


test_email = ["cheap", "price", "now"]

p_spam = 0.5
p_ham = 0.5

for w in test_email:
    p_spam *= word_prob(w, "spam")
    p_ham *= word_prob(w, "ham")

print("P(Spam|text) =", p_spam)
print("P(Ham|text)  =", p_ham)
print("Prediction   =", "Spam" if p_spam > p_ham else "Ham")
```

```
P(Spam|text) = 4.7999999999999994e-05
P(Ham|text)  = 3.358477132129207e-06
Prediction   = Spam
```

Q3

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

texts = [t for t, _ in emails]
labels = [l for _, l in emails]


vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(texts)


model = MultinomialNB()
model.fit(X, labels)


y_pred = model.predict(X)
print("Accuracy:", accuracy_score(labels, y_pred))
```

```
test = ["cheap price now"]
X_test = vectorizer.transform(test)
print("Prediction:", model.predict(X_test)[0])
```

```
Accuracy: 1.0
Prediction: spam
```