The solutions given here are in Matlab. Of course, you are free to use any programming language of your choice.

**Preprocessing**

We first pre-process the data. First copy 'spambase.data' into the same folder that contains your Matlab code. Then load the data using 'csvread()'.

```matlab
% Load data from file
data = csvread('spambase.data');

% Verify that data has 4601 rows and 58 columns
data_size = size(data);

% Filter away columns 49-57
data = data(:, [1:48 58]);

% We don't use word frequencies, so if data(i,j)>0, we call it 1, otherwise
% we call it 0
data = (data > 0);
```

**Problem 1**

Here, we separate the data into training set and test set. This is a standard procedure in machine learning. We first learn the patterns from the training set. We then use the knowledge learnt to make predictions on the test set. Finally we report the accuracy of our predictions.

```matlab
% Problem 1
% Separate last 200 emails of each type as test
% Remaining data is training data
spam_indices = find(data(:, end) == 1);
no_spam_indices = find(data(:, end) == 0);

spam = data(spam_indices, :);
no_spam = data(no_spam_indices, :);

spam_train = spam(1:end-200, 1:end-1);
spam_test = spam(end-199:end, 1:end-1);
```
Problem 2

Here, we estimate $P(W_i|S)$ and $P(W_i|S^c)$ from the training data for spam and legitimate emails respectively.

% Problem 2
% For this problem, we use training data to learn the probabilities
spam_word_count=sum(spam_train);
n_spam=size(spam_train,1);
p_wordi_given_spam=(spam_word_count+k)/(n_spam)+1/4201;

no_spam_word_count=sum(no_spam_train);
n_no_spam=size(no_spam_train,1);
p_wordi_given_no_spam=(no_spam_word_count+k)/(n_no_spam)+1/4201;

Note that the addition of 1/4201 ensures that all probabilities are non-zero.

Problem 3

In our problem, the email is represented by $(W_1, W_2, \ldots, W_{48})$ where each $W_i$ is either 1 or 0 indicating the presence or absence of Word $i$.

In this problem, we compute the probabilities $P(W_1, W_2, \ldots, W_{48}|S)$ and $P(W_1, W_2, \ldots, W_{48}|S^c)$ for each email $(W_1, W_2, \ldots, W_{48})$ among the 400 test cases.

We use the independence assumption:

$$P(W_1, W_2, \ldots, W_{48}|S) = P(W_1|S)P(W_2|S) \ldots P(W_{48}|S)$$
$$P(W_1, W_2, \ldots, W_{48}|S^c) = P(W_1|S^c)P(W_2|S^c) \ldots P(W_{48}|S^c)$$

This conditional independence assumption is a good approximation. It works reasonably well in predicting spam emails.

To avoid very small numbers, it is a good practice compute the log likelihood instead.

$$\log P(W_1, W_2, \ldots, W_{48}|S) = \log P(W_1|S) + \log P(W_2|S) + \cdots + \log P(W_{48}|S)$$
$$\log P(W_1, W_2, \ldots, W_{48}|S^c) = \log P(W_1|S^c) + \log P(W_2|S^c) + \cdots + \log P(W_{48}|S^c)$$
log_likelihood_of_no_spam=zeros(1,size(emails,1));

% Compute log likelihood for each test email
for i=1:size(emails,1)
    email=emails(i,:);
    present_words_indices=find(email==1);
    absent_words_indices=find(email==0);

    log_p_email_given_spam=... 
    sum(log(p_wordi_given_spam(present_words_indices)))... 
    +sum(log(1-p_wordi_given_spam(absent_words_indices)));
    log_likelihood_of_spam(i)=log_p_email_given_spam;

    log_p_email_given_no_spam=... 
    sum(log(p_wordi_given_no_spam(present_words_indices)))... 
    +sum(log(1-p_wordi_given_no_spam(absent_words_indices)));
    log_likelihood_of_no_spam(i)=log_p_email_given_no_spam;
end

Note that a common mistake is to ignore $P(W_i|S)$ when the $i$-th word does not appear in the calculation of $P(W_1, W_2, \ldots, W_{48}|S)$.

Problem 4

Bayes' rule gives

$$P(S|W_1, W_2, \ldots, W_{48}) = \frac{P(W_1, W_2, \ldots, W_{48}|S)P(S)}{P(W_1, W_2, \ldots, W_{48}|S)P(S) + P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)}$$

$$P(S^c|W_1, W_2, \ldots, W_{48}) = \frac{P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)}{P(W_1, W_2, \ldots, W_{48}|S)P(S) + P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)}$$

Note that $P(S|W_1, W_2, \ldots, W_{48}) + P(S^c|W_1, W_2, \ldots, W_{48}) = 1$. We will declare the email $(W_1, W_2, \ldots, W_{48})$ as spam if $P(S|W_1, W_2, \ldots, W_{48}) \geq 0.5$, and legitimate otherwise.

$$P(S|W_1, W_2, \ldots, W_{48}) > 0.5$$

$$\iff \frac{P(W_1, W_2, \ldots, W_{48}|S)P(S)}{P(W_1, W_2, \ldots, W_{48}|S)P(S) + P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)} \geq \frac{1}{2}$$

$$\iff 2P(W_1, W_2, \ldots, W_{48}|S)P(S) \geq P(W_1, W_2, \ldots, W_{48}|S)P(S) + P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)$$

$$\iff P(W_1, W_2, \ldots, W_{48}|S)P(S) \geq P(W_1, W_2, \ldots, W_{48}|S^c)P(S^c)$$

$$\iff P(W_1, W_2, \ldots, W_{48}|S) \geq P(W_1, W_2, \ldots, W_{48}|S^c)$$

The last inequality is because we assumed $P(S) = P(S^c) = \frac{1}{2}$. Thus we simplify the code for this section by comparing the log-likelihoods.

prediction=log_likelihood_of_spam>log_likelihood_of_no_spam;
A common mistake is to equate

\[ P(W_1, W_2, \ldots, W_{48}) = P(W_1)P(W_2)\ldots P(W_{48}) \]

which is not necessarily true, even with the assumption of conditional independence.

**Problem 5**

Here we have already computed the predictions. We just have to count the number of false positives and false negatives.

```matlab
% Problem 5
false_negatives = sum(prediction(1:200) == 0);
false.positives = sum(prediction(201:end) == 1);
```

**Problem 6**

In order to avoid cases where \( P(W_i|S) \) or \( P(W_i|S^c) \) is 0 or extremely small, we add a quantity \( k \) in both the numerator and denominator of our estimates.

```matlab
% Problem 6
% We iterate over different values of k
k_values = [0, 1, 2, 5, 10, 50, 100, 500];
for iter = 1:length(k_values)
    k = k_values(iter);
    % Problem 2
    % For this problem, we use training data to learn the probabilities
    spam_word_count = sum(spam_train);
    n_spam = size(spam_train, 1);
    p_wordi_given_spam = (spam_word_count + k) / (n_spam + k) + 1/4201;

    no_spam_word_count = sum(no_spam_train);
    n_no_spam = size(no_spam_train, 1);
    p_wordi_given_no_spam = (no_spam_word_count + k) / (n_no_spam + k) + 1/4201;
```

**Complete Code**

We now put all the code together.
% Load data from file
data = csvread('spambase.data');

% Verify that data has 4601 rows and 58 columns
data_size = size(data);

% Filter away columns 49-57
data = data(:,[1:48 58]);

% We don't use word frequencies, so if data(i,j)>0, we call it 1, otherwise we call it 0
data = (data>0);

% Problem 1
% Separate last 200 emails of each type as test
% Remaining data is training data
spam_indices = find(data(:,end)==1);
o_spam_indices = find(data(:,end)==0);

spam = data(spam_indices,:);
o_spam = data(o_spam_indices,:);

spam_train = spam(1:end-200,1:end-1);
spam_test = spam(end-199:end,1:end-1);

no_spam_train = no_spam(1:end-200,1:end-1);
o_spam_test = no_spam(end-199:end,1:end-1);

% Problem 6
% We iterate over different values of k
k_values = [0,1,2,5,10,50,100,500];
fpk = zeros(1,length(k_values));
fnk = zeros(1,length(k_values));

for iter = 1:length(k_values)
    k = k_values(iter);

    % Problem 2
    % For this problem, we use training data to learn the probabilities
    spam_word_count = sum(spam_train);
n_spam = size(spam_train,1);
p_wordi_given_spam = (spam_word_count + k) / (n_spam + k) + 1/4201;

    no_spam_word_count = sum(no_spam_train);
n_no_spam = size(no_spam_train,1);

p_wordi_given_no_spam=(no_spam_word_count+k)/(n_no_spam+k) + 1/4201;

% Problem 3
% Mix spam emails and legitimate emails in test set
emails=[spam_test; no_spam_test];

% Results will be computed for each email and stored in these variables
log_likelihood_of_spam=zeros(1,size(emails,1));
log_likelihood_of_no_spam=zeros(1,size(emails,1));

% Compute log likelihood for each test email
for i=1:size(emails,1)
    email=emails(i,:);
    present_words_indices=find(email==1);
    absent_words_indices=find(email==0);
    log_p_email_given_spam=...
        sum(log(p_wordi_given_spam(present_words_indices)))...
        +sum(log(1-p_wordi_given_spam(absent_words_indices)));
    log_likelihood_of_spam(i)=log_p_email_given_spam;
    log_p_email_given_no_spam=...
        sum(log(p_wordi_given_no_spam(present_words_indices)))...
        +sum(log(1-p_wordi_given_no_spam(absent_words_indices)));
    log_likelihood_of_no_spam(i)=log_p_email_given_no_spam;
end

% Problem 4
% Predictions
% We will compare log p(email | spam) against p(email | no spam)
% It can be shown that this is equivalent to applying Bayes’ rule
% Prediction=1 if we predict spam, 0 otherwise.
prediction=log_likelihood_of_spam>log_likelihood_of_no_spam;

% Problem 5
false_negatives=sum(prediction(1:200)==0);
false_positives=sum(prediction(201:end)==1);
fpk(iter)=false_positives;
fnk(iter)=false_negatives;
end
% Display k, false_positives, false_negatives
[k_values;fpk;fnk]

Here is the output:

ans =
<table>
<thead>
<tr>
<th>k</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>44</td>
<td>44</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>45</td>
<td>38</td>
<td>7</td>
</tr>
<tr>
<td>FN</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>57</td>
<td>60</td>
<td>93</td>
</tr>
</tbody>
</table>

The first row is the value of \( k \). The second row is the number of false positives and the third row is the number of false negatives.