1.

a. Θ^{t+1} is defined to be the vector parameter where the algorithm makes its t^{th} error. There for, because t starts at 1, $\Theta^1 = 0$

b. the update of the t^{th} error $\Theta^{t+1} = \Theta^t + y_k x_k$ by the definition of the algorithm. We can add another parameter vector Θ^* to each side and that will keep the equality the same. We are thereby able to write the equation as $\Theta^{t+1} \cdot \Theta^* = (\Theta^t + y_k x_k) \cdot \Theta^*$

c. From (7) we have $\Theta^{t+1} \cdot \Theta^* = \Theta^t \cdot \Theta^* + y_k x_k \cdot \Theta^*$. We are told that and some scalar $\gamma > 0$ such that for all i = 1, ..., n, $y_k x_k \cdot \Theta^* \geq \gamma$ By substituting γ for $y_k x_k \cdot \Theta^*$ we can see that: $\Theta^t \cdot \Theta^* + y_k x_k \cdot \Theta^* \geq \Theta^t \cdot \Theta^* + \gamma$ (8)

d. We can solve this proof by induction:
Assume that for t, $\Theta^t \cdot \Theta^* = (t-1)\gamma$
Then $\Theta^{t+1} \cdot \Theta^* = \Theta^t \cdot \Theta^* + \gamma = (t-1)\gamma + \gamma = t\gamma$

e. Because $|\,|\,||\Theta^{t+1}|| \times ||\Theta^*|| \geq \Theta^{t+1} \cdot \Theta^*$ *and* from the explanation from D we saw ,
$\Theta^{t+1} \cdot \Theta^* = \Theta^t \cdot \Theta^* + \gamma = (t-1)\gamma + \gamma = t\gamma$
So we have $||\Theta^{t+1}|| \geq t\gamma$

f. $y_t^2 = 1$ *by assuptions of the theorom* $y_t^2||x_t||^2 = ||x_t||^2 \leq R^2$
Also, $2y_k x_k \cdot \Theta^t \leq 0$ because the parameter vector
Θ^t *gave an error on the* t^{th} *example*
So $||\Theta^{t+1}||^2 = ||\Theta^t||^2 + y_t^2||x_t||^2 + 2y_k x_k \cdot \Theta^t \leq ||\Theta^t||^2 + R^2$

g. Combining equations (10) and (14) gives us the following equation:
$t^2\gamma^2 \leq ||\Theta^{t+1}||^2 \leq tR^2$
Which can be simplified to be $t \leq R^2/\gamma^2$

2.

1. In 15(a), the slack variable ξ(i) can be substituted with the loss hinge function s.t
   1) yi(θ · xi + θ0) ≥ 1 – ξi 2) ξi
   ≥ 0, for i = 1, ..., n

   by definition of the lhinge function we see that

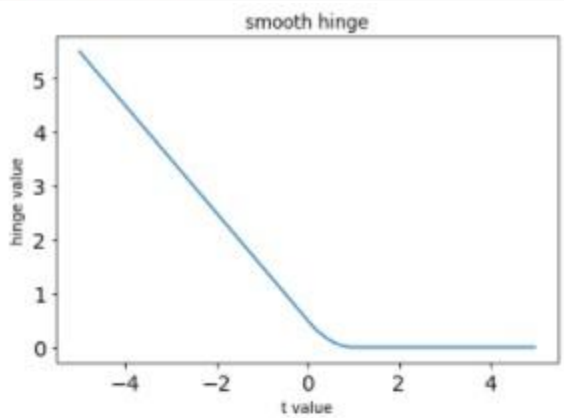
   1) yi(θ · xi + θ0) ≥ 1 – lhinge(yi(θ · xi + θ0))
   2) lhinge(yi(θ · xi + θ0)) ≥ 0, for i = 1, ..., n

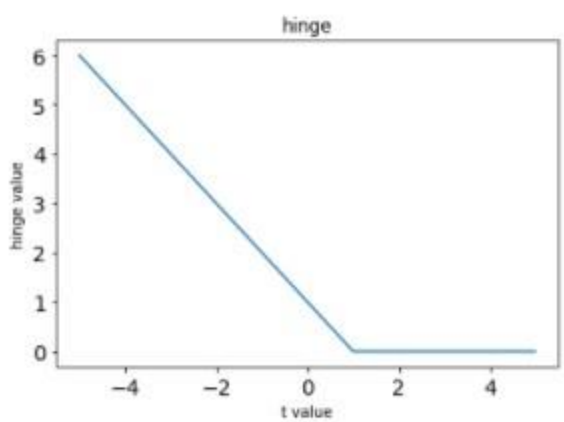
2.

```
### STUDENT: Start of code ### def
hinge_loss_smooth(t):
```

```
   if (t<=0):
      return ((1/2)-t)
   elif(t>0 and t<1):
      return (1/2)*(pow((1-t),2))
else:        return (0)

def hinge_loss(t):

   if (t<1):
return (1-t)
else :
return (0)
       x
= []
sh = [0]*500 h = [0]*500 for i
in range(500):     t = (i-250)/50
x.append(t)
sh[i]=(hinge_loss_smooth(t))
   h[i]=(hinge_loss(t))



plt.plot(x,sh) plt.title('smooth
hinge') plt.ylabel('hinge
value') plt.xlabel('t value')
plt.show()


plt.plot(x,h) plt.title('hinge')
plt.ylabel('hinge value')
plt.xlabel('t value')
plt.show()


### End of code ###
```

```
### STUDENT: Start of code ###
def hinge_loss_smooth(t):

    if (t<=0):
        return ((1/2)-t)
    elif(t>0 and t<1):
        return (1/2)*(pow((1-t),2))
    else:
        return (0)

def hinge_loss(t):

    if (t<1):
        return (1-t)
    else :
        return (0)

x = []
sh = [0]*500
h = [0]*500
for i in range(500):
    t = (i-250)/50
    x.append(t)
    sh[i]=(hinge_loss_smooth(t))
    h[i]=(hinge_loss(t))


plt.plot(x,sh)
plt.title('smooth hinge')
plt.ylabel('hinge value')
plt.xlabel('t value')
plt.show()


plt.plot(x,h)
plt.title('hinge')
plt.ylabel('hinge value')
plt.xlabel('t value')
plt.show()


### End of code ###
```





3. $L_{\text{smooth-hinge}}(t)dt = \{ \begin{array}{ll} -1 & if\ t \leq 0 \\ \text{1-t} & \text{if } 0 < t < 1 \\ 0 & \text{if } t \geq 1 \end{array} \}$

4. $t = (y_i(x_i \cdot \theta + \theta_0))$

$$L_{\text{smooth-hinge}}(y_i(x_i \cdot \theta + \theta_0))d\theta = \{ \begin{array}{ll} -y_i(x_i) & if\ t \leq 0 \\ 1\text{- }(y_i(x_i \cdot \theta + \theta_0)) & \text{if } 0 < t < 1 \\ 0 & \text{if } t \geq 1 \end{array} \}$$

$$L_{\text{smooth-hinge}}(y_i(x_i \cdot \theta + \theta_0))d\theta_0 = \{ \begin{array}{ll} -y_i & if\ t \leq 0 \\ 1\text{- }(y_i(x_i \cdot \theta + \theta_0)) & \text{if } 0 < t < 1 \\ 0 & \text{if } t \geq 1 \end{array} \}$$

5.

```
## STUDENT: Start of code ###
    score_matrix = (feature_matrix.dot(theta) + theta0)  * labels

    feature_matrix1=np.array([np.zeros(4500)]*2500)
feature_matrix2=np.array([np.zeros(4500)]*2500)

    for i in range(len(score_matrix)):
if score_matrix[i]<=0:
        feature_matrix1[i] = feature_matrix[i]
elif score_matrix[i]<1:
        feature_matrix2[i] = feature_matrix[i]


    grad_theta =  2*theta-C*((feature_matrix1).T).dot(labels)-
C*((feature_matrix2).T).dot(labels*(1-(feature_matrix2.dot(theta)+theta0)*labels))
grad_theta0 = np.sum(-C*((feature_matrix1).T).dot(labels)-
C*((feature_matrix2).T).dot(labels*(1-(feature_matrix2.dot(theta)+theta0)*labels)))


    return grad_theta, grad_theta0
    # End of code ###
```

```
def weight_derivative(theta, theta0, C, feature_matrix, labels):
    # Input:
    # theta: weight vector theta, a numpy vector of dimension d
    # theta0: intercept theta0, a numpy vector of dimension 1
    # C: constant C
    # feature_matrix: numpy array of size n by d, where n is the number of data points, and d is the feature dimension
    # labels: true labels y, a numpy vector of dimension d, each with value -1 or +1
    # Output:
    # Derivative of the cost function with respect to the weight theta, grad_theta
    # Derivative of the cost function with respect to the weight theta0, grad_theta0


    ## STUDENT: Start of code ###
    score_matrix = (feature_matrix.dot(theta) + theta0)  * labels

    feature_matrix1=np.array([np.zeros(4500)]*2500)
    feature_matrix2=np.array([np.zeros(4500)]*2500)

    for i in range(len(score_matrix)):
        if score_matrix[i]<=0:
            feature_matrix1[i] = feature_matrix[i]
        elif score_matrix[i]<1:
            feature_matrix2[i] = feature_matrix[i]


    grad_theta =  2*theta-C*((feature_matrix1).T).dot(labels)-C*((feature_matrix2).T).dot(labels*(1-(feature_matrix2.dot(theta)+t
    grad_theta0 = np.sum(-C*((feature_matrix1).T).dot(labels)-C*((feature_matrix2).T).dot(labels*(1-(feature_matrix2.dot(theta)+t


    return grad_theta, grad_theta0
    # End of code ###
```

```
# STUDENT: PRINT THE OUTPUT AND COPY IT TO THE SOLUTION FILE
theta = np.ones(data_mat.shape[1]) # a weight of all 1s

theta0 = np.ones(1) # a number 1
C = 0.05
grad_theta, grad_theta0 = weight_derivative(theta, theta0, C,train_data,train_labels)

print (grad_theta[:10])
print (grad_theta0)
```

```
[2.   2.05 2.05 2.1  2.05 2.05 3.8  2.15 2.   2.  ]
592.5500000000001
```

6.

```
# Initialize the weights, step size and tolerance
# Start of code
initial_theta = np.ones(data_mat.shape[1]) ## STUDENT: initialize theta
initial_theta0 = 1   ## STUDENT: initialize theta0
C =  0.05## STUDENT: choose the C
step_size =  0.25## STUDENT: choose the step_size
tolerance =  0.5## STUDENT: choose the tolerance

# end of code

theta, theta0 = adam_optimizer(train_data,train_labels, initial_theta, init
print (theta)
print (theta0)
```

```
Iteration:  92 objective:  40.48889739510529 tr err:  0.108 gradient_magnitude:  2.917119368550896
Iteration:  93 objective:  40.60111390676613 tr err:  0.1064 gradient_magnitude:  1.7208277488403978
Iteration:  94 objective:  40.584498871127614 tr err:  0.1072 gradient_magnitude:  4.623163530006727
Iteration:  95 objective:  40.429846851473734 tr err:  0.104 gradient_magnitude:  4.794827802291178
Iteration:  96 objective:  40.392105530804486 tr err:  0.1064 gradient_magnitude:  2.29350759361624
Iteration:  97 objective:  40.4797115719878 tr err:  0.1112 gradient_magnitude:  1.4457211050388274
Iteration:  98 objective:  40.482363959918565 tr err:  0.1124 gradient_magnitude:  3.871047426382583
Iteration:  99 objective:  40.40173705006277 tr err:  0.1068 gradient_magnitude:  3.862558691278447
Iteration:  100 objective:  40.402259392763135 tr err:  0.1052 gradient_magnitude:  1.642998029952203
Iteration:  101 objective:  40.460004770136564 tr err:  0.1036 gradient_magnitude:  1.4719824265025059
Iteration:  102 objective:  40.43778652473517 tr err:  0.1028 gradient_magnitude:  3.241571235803624
Iteration:  103 objective:  40.37078679536347 tr err:  0.1068 gradient_magnitude:  2.952188597251902
Iteration:  104 objective:  40.372147817825066 tr err:  0.1096 gradient_magnitude:  0.984490808285676
Iteration:  105 objective:  40.40497741235107 tr err:  0.1104 gradient_magnitude:  1.5031123759722722
Iteration:  106 objective:  40.38433499824998 tr err:  0.11 gradient_magnitude:  2.712678533926603
Iteration:  107 objective:  40.3516492282706 tr err:  0.1072 gradient_magnitude:  2.159703990259043
Iteration:  108 objective:  40.369307267958455 tr err:  0.1048 gradient_magnitude:  0.42838936555828144
[ 0.00600027 -0.01356996 -0.02692656 ... -0.08502523 -0.00293407
 -0.01479011]
[-0.1015464]
```

```python
print ("Training error: ", float(errs_train)/len(train_labels))
print ("Test error: ", float(errs_test)/len(test_labels))
```

```
Training error:  0.1048
Test error:  0.188
```

7.

```python
# STUDENT: your code here
## number of sentaneces found that meet criteria
good_found =0
bad_found =0

good_sentance = []
bad_sentance = []
for i in range(len(preds_train)):
    if((preds_train[i] > 0.0) and (train_labels[i] < 0.0)):
        if (bad_found<4):
            bad_found+=1
            bad_sentance.append(sentences[i])
    elif((preds_train[i] < 0.0) and (train_labels[i] > 0.0)):
        if (bad_found<4):
            bad_found+=1
            bad_sentance.append(sentences[i])
    else:
        if (good_found<4):
            good_found+=1
            good_sentance.append(sentences[i])
    if good_found>=4 and bad_found>=4:
        break

print('Correctly identified  sentances')
for sentance in good_sentance:
    print(sentance)
print(' ')
print('Incorrectly identified sentances')
for sentance in bad_sentance:
    print(sentance)
```

```
Correctly identified  sentances
So there is no way for me to plug it in here in the US unless I go by a converter.
Good case, Excellent value.
Great for the jawbone.
Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!

Incorrectly identified sentances
If you are Razr owner...you must have this!
You need at least 3 mins to get to your phone book from the time you first turn on the phone.Battery life is short.
A week later after I activated it, it suddenly died.
Even in my BMW 3 series which is fairly quiet, I have trouble hearing what the other person is saying.
```