1.

Programmed with R, the .R file is attached with the submission;

```r
GradientDescentAlgorithm <- function(k, n_iter) {
  # Step size: k
  # Number of iteration: n_iter

  training_data <- matrix(c(3,1,0,4,2,2,1,3), nrow = 4, ncol = 2)
  m = nrow(training_data)

  θ <- matrix(c(0,1), nrow = 2, ncol = 1)
  J <- matrix(0, nrow = n_iter, ncol = 1)
  dJ_dθ = c(0,0)

  for (i in 1:n_iter) {
    for (j in 1:m) {
      J[i,1] = J[i,1] + 1/(2*m) * (θ[1,1] + θ[2,1] * training_data[j,1] - training_data[j,2])^2
      dJ_dθ = dJ_dθ + 1/m * (θ[1,1] + θ[2,1] * training_data[j,1] - training_data[j,2]) * c(1, training_data[j,1])
    }
    θ <- θ - k * dJ_dθ
  }
  show(J)
  show(θ)
}
```

Through experiment, only with 5 iterations, step size = 0.1775 has been found to have the lowest error of 0.189. For this step size, the error doesn't go down after 5 iterations.

Error through each iteration and the final values of are shown below:

> GradientDescentAlgorithm(0.1775,5)
error:
[1] 0.5000000
[2] 0.3310145
[3] 0.4595113
[4] 0.2308235
[5] 0.1893229

θ:
θ_0: 1.0223701
θ_1: 0.7004018

2.
False Positive: 10%
— Those who do not have disease are identified with disease;

False Negative: 20%
— Those who have disease are not identified with disease.

3.
a.
Pros: If new data comes in and S is still consistent, then we would have a precise knowledge of the concept we need to learn;
Cons: S is not likely to be consistent with all the new data that comes in.

b.
Pros: G is likely to be consistent with the new data coming in;
Cons: We would have a too general knowledge of the concept that we need to learn.

4.
Consistent hypothesis: a hypothesis h is consistent with a set of training examples D of target concept c, if h(x) = c(x) for each training example <x, c(x)> in D.

Version space: with respect to hypothesis space H and training examples D, version space is the subset of hypotheses from H consistent with D

5. the least constrained

6.
a. $2^4 = 16$

b. $1 + 4*2 + 6*2^2 + 4*2^3 + 1*2^4 = 81$

c. $3^4 * 2 = 162$

d. $6*2 = 12$

e. $2*2 = 4$

7.

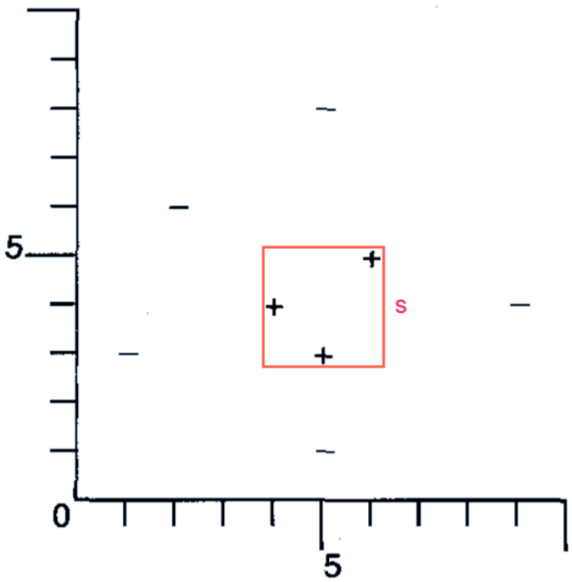
(⟨1, 1, 0, 1, 1⟩, 1)	h1 = (1, 1, 0, 1, 1)
(⟨0, 1, 0, 1, 1⟩, 0)	h2 = (1, 1, 0, 1, 1)
(⟨1, 1, 1, 1, 0⟩, 1)	h3 = (1, 1, ?, 1, ?)
(⟨0, 0, 0, 1, 1⟩, 0)	h4 = (1, 1, ?, 1, ?)
(⟨1, 1, 1, 1, 1⟩, 1)	h5 = (1, 1, ?, 1, ?)

8.
+ : $((GPA < 3.5) \wedge (Exp >= 3)) \vee ((GPA >= 3.5) \wedge (Exp >= 1))$

- : $((GPA < 3.5) \wedge (Exp < 3)) \vee ((GPA >= 3.5) \wedge (Exp < 1))$

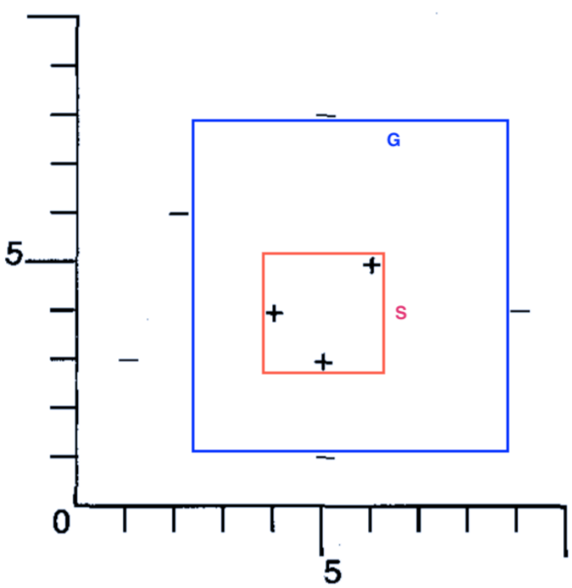
9.

a.



S boundary: {<4 <= x <= 6, 3 <= y <= 5>}

b.



G boundary: {<3 <= x <= 8, 2 <= y <= 7>}

c.
A query which is in between of the S and G boundary will be guaranteed to reduce the version space, regrades of how it will be labeled. The query can be (7, 5).

A query which is inside of S boundary or outside of G boundary won't be able to reduce the size of the version space.

d.
It would need four positive training examples, and four negative training examples. Each positive example should be at each corner of the boundary, each negative example should be against each corner of the boundary.

10.
a.

		S: {<(∅, ∅, ∅, ∅), (∅, ∅, ∅, ∅)>} G: {<(?, ?, ?, ?), (?, ?, ?, ?)>}
〈(ug, se, l, hs), (gr, cs, h, hs)〉	+	S: {<(ug, se, l, hs), (gr, cs, h, hs)>} G: {<(?, ?, ?, ?), (?, ?, ?, ?)>}
〈(ug, se, h, fr), (gr, cs, h, hs)〉	+	S: {<(ug, se, ?, ?), (gr, cs, h, hs)>} G: {<(?, ?, ?, ?), (?, ?, ?, ?)>}
〈(gr, se, l, so), (gr, cs, h, se)〉	-	S: {<(ug, se, ?, ?), (gr, cs, h, hs)>} G: {<(ug, ?, ?, ?), (?, ?, ?, ?)>, <(?, ?, ?, ?), (?, ?, ?, hs)>}
〈(ug, se, l, ju), (gr, se, h, ju)〉	+	S: {<(ug, se, ?, ?), (gr, ?, h, ?)>} G: {<(ug, ?, ?, ?), (?, ?, ?, ?)>}

b.

S:	{<(ug, se, ?, ?), (gr, ?, h, ?)>}
Between S and G:	<(ug, ?, ?, ?), (gr, ?, h, ?)> <(ug, se, ?, ?), (gr, ?, ?, ?)>
G:	{<(ug, ?, ?, ?), (?, ?, ?, ?)>}

(1) In total 4 hypotheses are returned;

(2) Consistent with 〈(ug, cs, h, do), (gr, ma, l, se)〉 +:
Only 1, the most general hypothesis {<(ug, ?, ?, ?), (?, ?, ?, ?)>}