1. （15 分）我們有下面的 BNF 文法。

S ::= NP AS

NP ::= NOUN PHRASES

NOUN ::= “a man” | “a lady” | “a dog” | “a cat”

PHRASES ::= | PROP NOUN PHRASES

PROP ::= “with” | “for” | “by”

AS ::= VERB NP | AS CONC AS

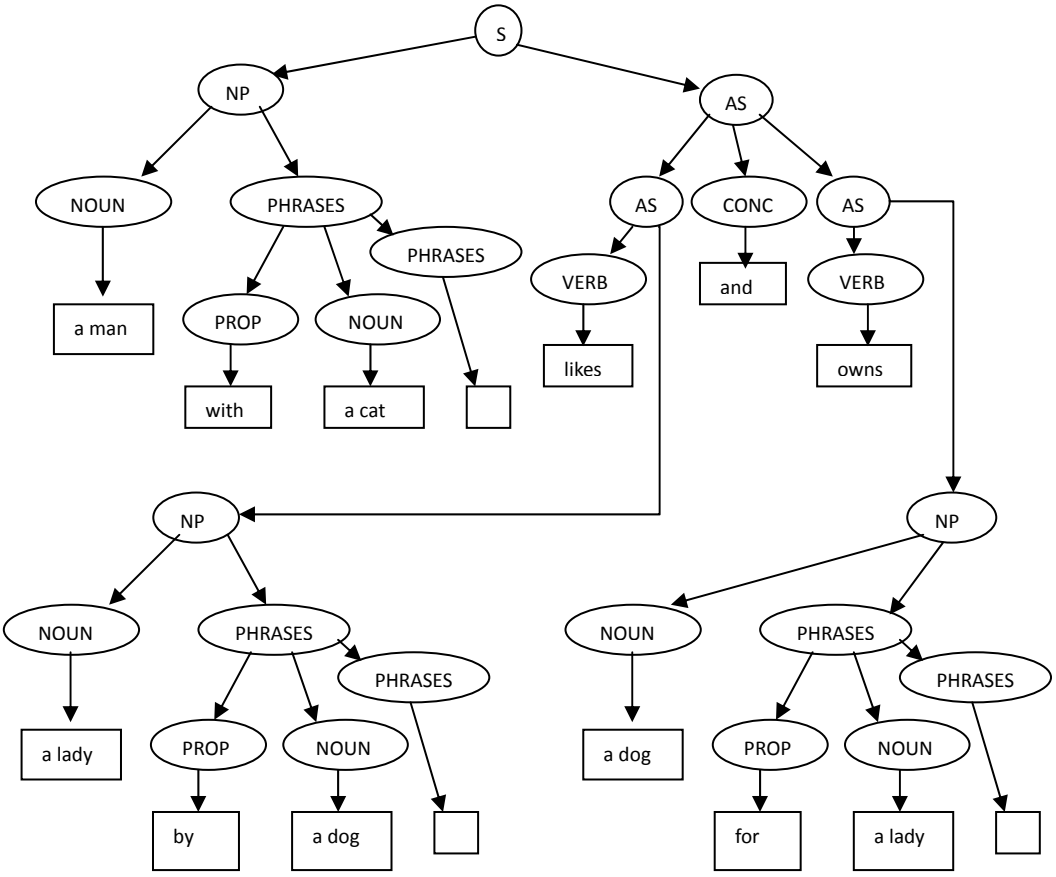
VERB ::= “likes” | “owns” | “catches”

CONC ::= “and”

請依據上述文法，畫出下面句子的 derivation tree。

“a man with a cat likes a lady by a dog and owns a dog for a lady”

解答：



2. （15 分）

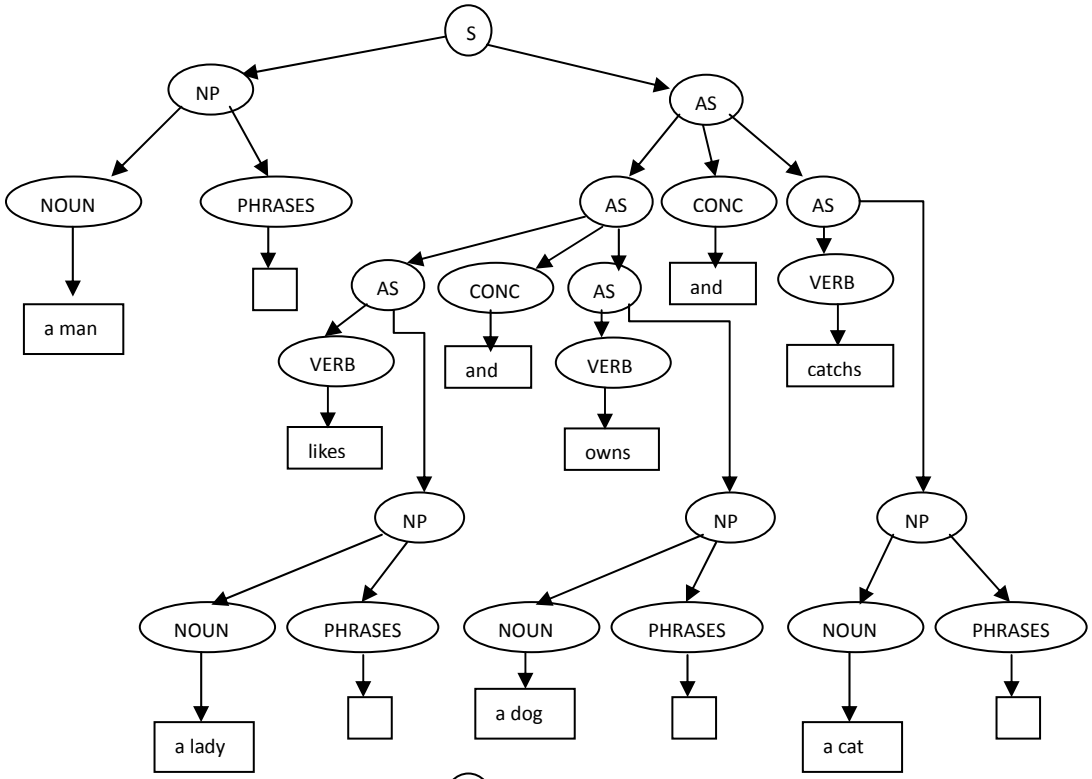
2.a 請問在第一題的 BNF 文法，是不是 ambiguous?

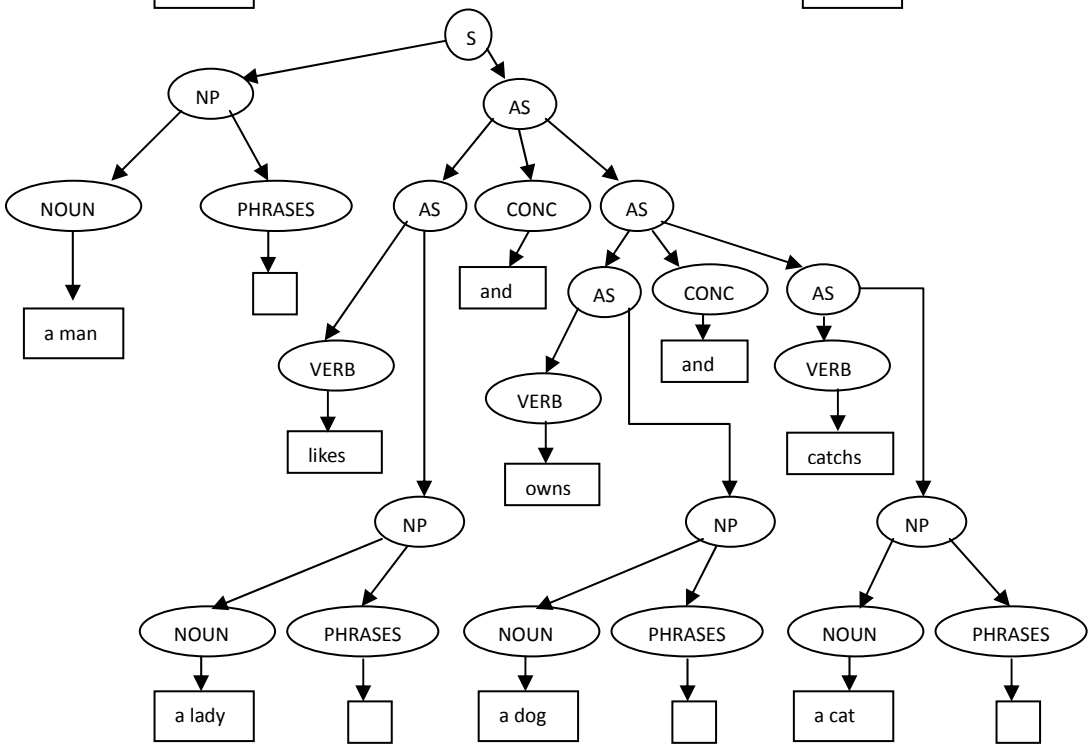
2.b 請證明你的答案。

解答：

2.a Ambiguous.

2.b 因為 a man likes a lady and owns a dog an catches a dog 有下面兩個 derivation trees。





3. （20 分）我們有下列 C/C++語言程序。

```
int f(int n) {
  if (n == 1 || n == 2)
     return(1);
  else
     return(f(n-1)+f(n-2));
}
```

3.a 請描述這個程序在算什麼？

3.b 請把這個程序的計算目的的遞迴方程式寫出來，並且把對任意引數 n 的解，寫出來。

3.c 請問這個程序的計算時間複雜度是多少？

3.d 請問根據你的描述，這個程序有什麼錯誤？

解答：

3.a 這個函數在計算 Fibonacci 函數。

3.b f(n)=f(n-1)+f(n-2), n>2; f(1)=f(2)=1.

f(n) = (((1+sqrt(5))/2)^n - ((1-sqrt(5))/2)^n) /sqrt(5)。

這裡 sqrt(5)是 5 的平方根。

3.c 這個程序的複雜度是 O（2^n)。

3.d 當 n 小於等於零時，這個程序不會結束。

4. （25 分）假設我們可以在程式中宣告 semaphore 變數，而且我們有下列個系統程序。

sem_wait(semaphore s)

sem_signal(semaphore s)

4.a 請問什麼是 semaphore 變數？

4.b 請問這兩個程序的功能為何？

4.c 請用 spinlock 的技術，寫出上述兩個程序的內容。

解答：

4.a

semaphore 變數是用來做 processes（或 threads）之間的 synchronization 變數。semaphore 本身是一個整數變數。我們可以對他做 wait 或 signal 兩種運算。

4.b

一開始，semaphore 變數若是大於零，代表了可用的資源數。若是小於零，代表了等待使用的資源的 process 數目。

每次 wait 後，就會把變數的值減一，若是小於零，代表作 wait 指令的 process 要等待。若是大於等於零，就可以繼續進行運算。

每次 signal 後，就會把 semaphore 變數值加一。加一後，若變數值大於等於零，就可以讓一個正在等待中的 process 繼續進行。

4.c

```
semaphore s = 1;
sem_wait(semaphore s) {
  while (s <= 0) ;
  s--;
}
signal_lock(semaphore s) {
  s++;
}
```

5. （25 分）假設我們有下列程式片段，有六個指令，有 a,b,c,d,e,k 等六個變數。

```
1: b = a;         // LSU    5 us
2: b = b + 3;     // ALU    10 us
3: k = d;         // LSU    5 us
4: c = d + 5;     // ALU    10 us
5: e = c;         // LSU    5 us
6: a = b / k;     // ALU    20 us
```

假設這個程式，現在經過了 compiler 的分析，要在一個 single CPU 上執行。在指令的左邊，我們寫上了指令的行號。指令的右邊，我們寫上了執行指令所需使用的硬體元件與時間（以微秒 us 為單位）。ALU 代表所用的硬體元件是 CPU 中唯一的算術邏輯單元。LSU 代表所用的硬體元件是 CPU 中唯一的存取單元。我們假設這個 CPU，可以容許 LSU 與 ALU 同時執行不同的指令。

5.a 請問這個程式片段，在 in-order execution 時，需要花多少時間？

5.b 請畫出上述程式片段的 data-dependency graph。

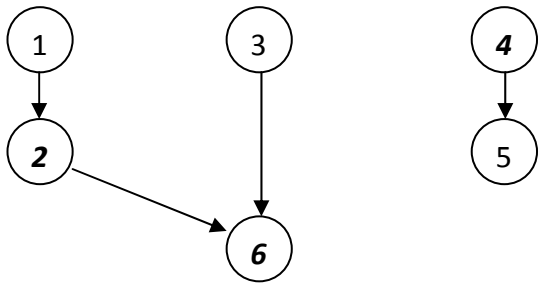
5.c 依照上述 data-dependency graph，這個程式片段，在我們的 single CPU 假設下，做 out-of-order execution 時，請問最短可以用多少時間完成？

5.d 在 5.c 的答案下，所需的 out-of-order execution 是什麼？

解答：

5.a 70us

5.b



5.c 40us

5.d

LSU/ALU

1/4 // 5us

3/4 // 5us

5/2 // 5us

*/2 // 5us

*/6 // 20us