1. （20 分）我們有下面的 BNF 文法。

S ::= NP AS

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

AS ::= VERB NOUN | AS CONC AS

VERB ::= “likes” | “owns” | “catches”

CONC ::= “and”

上述文法，容許名詞，但不容許形容名詞的介繫詞片語。譬如我們有下列句子，使用了許多這樣的片語：“with a cat”、 “for a lady”、 “by a dog”、 “of a man”。

“a man with a cat for a lady by a dog owns a dog of a man”

1.a 請修改上述文法，讓它可以 derive 上述範例中，無限制的長度的介繫詞片語。

1.b 請用修改過後的文法，將上述句子的 derivation tree 畫出來。

解答：

1.a S ::= NP AS

NP ::= NOUN PHRASES

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

PHRASES ::= | PROP NOUN PHRASES

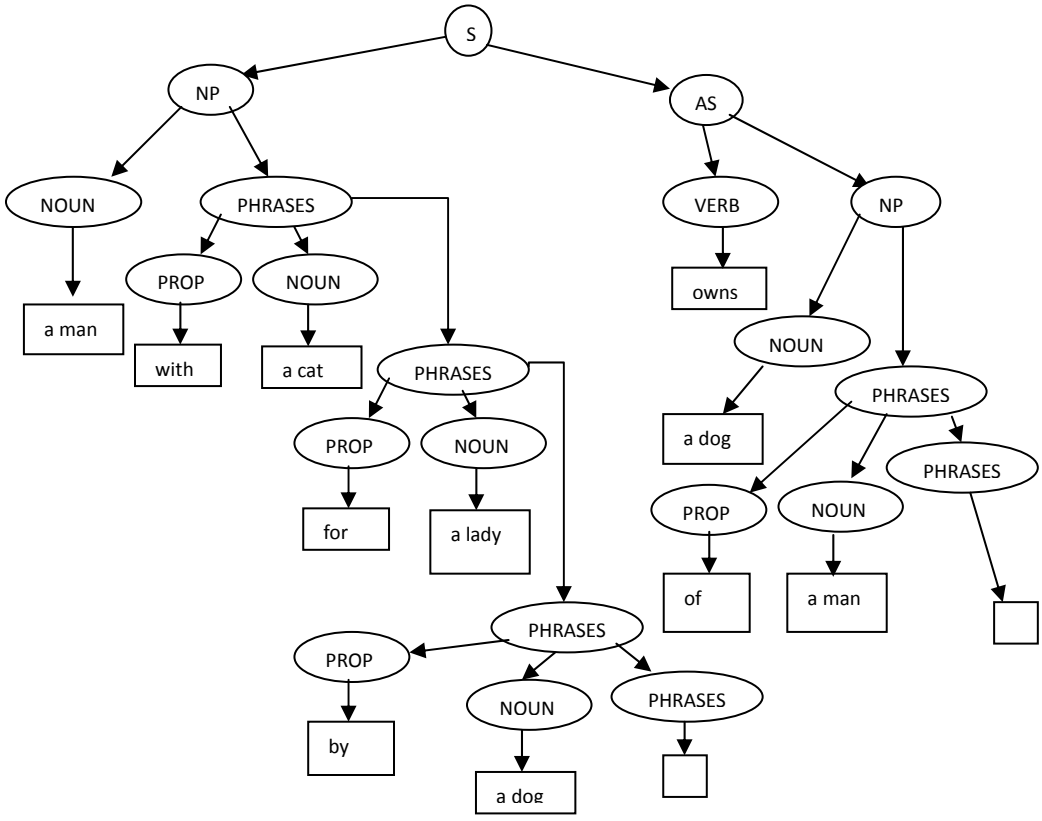
PROP ::= “with” | “for” | “by” | “of”

AS ::= VERB NP | AS CONC AS

VERB ::= “likes” | “owns” | “catches”

CONC ::= “and”

1.b



2. （20 分）Fibonacci sequence 的定義如下。

f(1) = 1

f(2) = 1

f(n)=f(n-1)+f(n-2), n>2

請寫出一個 C/C++的程序，接受一個輸入引數 n，並且計算 f(n)的值。這個程序的時間複雜度為 linear time。你寫出來的程序，必須能夠處理不適當的 n 值。若是需要任何資料結構，你的程序必須自行宣告，而不能假設在這程序執行之前已經存在。

解答：

```
f(n) {
  int   *m, i;

  if (n <= 0) return (0);
  else if (n <= 2) return (1);
  m = (int *) malloc((n+1)*sizeof(int));
  m(1)=m(2)=1;
  for (i = 3; i <= n; i++)
    m(i)=m(i-1)+m(i-2);
  return (m(n));
}
```

3. （20 分）假設我們有下列程式片段，有六個指令，有 a、b、c、d、e、f、g、h、i、x、y, 等 11 個變數。

```
1: a = x * x;     // 20 us
2: b = a + 8;     // 10 us
3: c = a * y;     // 20 us
4: d = a - 4;     // 10 us
5: e = a / y;     // 20 us
6: f = b / c;     // 20 us
7: g = d- e;      // 10 us
8: h = f * g;     // 20 us
9: i = h * y;     // 20 us
```

假設這個程式，現在經過了 compiler 的分析，要在一個三個 cores 的 shared-memory 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時間（以微秒 us 為單位）。

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

3.b 請畫出上述程式片段的 data-dependency graph。

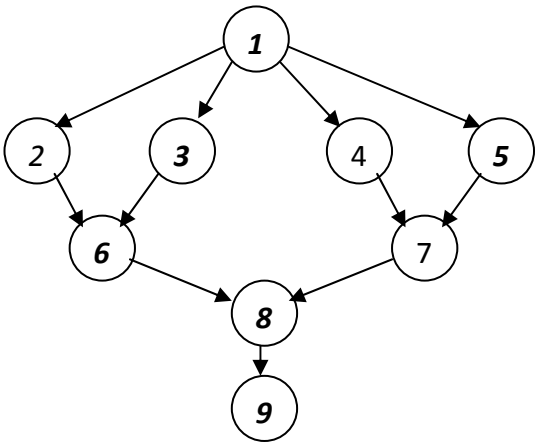
3.c 依照上述 data-dependency graph，這個程式片段，在我們的 3-core shared-memory CPU 假設下，做 out-of-order execution 時，請問最短可以用多少時間完成？

3.d 所需的 out-of-order execution 是什麼？

解答：

3.a 150us

3.b



3.c 100us

3.d

core 1 / core 2 / core 3
1 / * / * // 20 us
2 / 3 / 5 // 10 us
4 / 3 / 5 // 10 us
6 / 7 / * // 10 us
6 / * / * // 10 us
8 / * / * // 20 us
9 / * / * // 20 us

4. （40 分）假設我們可以在程式中宣告 semaphore 變數，而且我們有下列系統程序。

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 請問什麼是 synchronization 研究中的 reader-writer 問題。

用 semaphore 來解決 reader-writer 的問題，我們需要設計下列五個單元。

4.d Global 變數的宣告。這些變數，用來控制 Reader 與 Writer 的行為，以免違法了 Reader-Writer 問題的解決。

4.e Writer 的 Entry section 程式碼。這部分的程式碼，是 Writer 每次要寫入時，必須先執行的程式碼。

4.f Writer 的 Exit section 程式碼。這部分的程式碼，是 Writer 每次寫完後，必須執行的程式碼。

4.g Reader 的 Entry section 程式碼。這部分的程式碼，是 Reader 每次要讀出時，必須先執行的程式碼。

4.h Reader 的 Exit section 程式碼。這部分的程式碼，是 Reader 每次讀完後，必須執行的程式碼。

請把上述五個單元的內容填寫完善。。

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

這個問題，有許多 process(或 thread)要去讀、寫一些變數。我們可以容許許多 reader 同時去讀，但是不能容許超過一個以上的 processes 去寫。

4.d

```
semaphore     mutex = 1, wrt = 1;
int           readcount = 0;
```

4.e

```
sem_wait(wrt);
```

4.f

```
sem_signal(wrt);
```

4.g

```
sem_wait(mutex);
readcount = readcount + 1;
if (readcount == 1)
  sem_wait(wrt);
sem_signal(mutex);
```

4.h

```
sem_wait(mutex);
readcount = readcount - 1;
if (readcount == 0)
  signal(wrt);
signal(mutex);
```