1.1.1

1 En la definición que se ha dado de alfabeto se pide que A sea no vacío. Si hay un símbolo, en A* hay palabras en las que este símbolo se repite n veces para cualquier n>0. Es decir, hay infinitas palabras. Si la definición permite que A sea vacío la afirmación no es cierta, porque en este caso hay un único lenguaje posible (el que solo contiene la palabra vacía ε).

2 Una posible descripción: “una fbf es cualquier cadena finita de símbolos de A, en donde aparezca cada símbolo de A al menos una vez”. Sin embargo, debe entenderse que la respuesta no es única y puede ser tan trivial como "una fbf es una de las cadenas abc, abcab, bcac, bbabbc".

3
```
L → a⟨Símbolo⟩*b
⟨Símbolo⟩ → a|b|c.
```

4 Sea D = {0,1,2,3,4,5,6,7,8,9}. Se define
```
⟨dd⟩ → 0|1|2|3|4|5|6|7|8|9            // dígitos decimales
⟨num_dec⟩ → ⟨dd⟩ | ⟨num_dec⟩⟨dd⟩       // numeros decimales
```

Para representar números en base k>1 se define un alfabeto E = {0,1,…,k-1} y una BNF así:
```
⟨dk⟩ → 0|1|…|k-1                       // dígitos base k (k-1 es un símbolo)
⟨num_k⟩ → ⟨dk⟩ | ⟨num_k⟩⟨dk⟩           // numeros en base k
```

1.1.2

1 Las cadenas que el lenguaje puede formar son todas aquellas que inician con el símbolo a y terminan con el símbolo c y, en medio de éstos, palabras que contengan cualquier cantidad de símbolos b, c, d o e. Por tanto, los números que se pueden representar con palabras del lenguaje son de la forma $1*2^{x}*3^{y+1}*4^{w}*5^{z}$, con $x,y,w,z \geq 0$.

2 La siguiente gramática define los números ternarios:
```
⟨dt⟩ → 0 | 1 | 2
⟨num_ter⟩ → ⟨dt⟩ | ⟨num_ter⟩⟨dt⟩
```

La siguiente semántica interpreta los elementos del lenguaje ⟨num_ter⟩ como valores de números ternarios:
```
H(0) = 0
H(1) = 1
H(2) = 2
H(⟨num_ter⟩⟨dt⟩) = 3*H(⟨num_ter⟩) + H(⟨dt⟩)
```

3 El alfabeto se define como A = {S,0,+,*,(,)}. La BNF puede ser:
```
⟨MSNat⟩ → ⟨SNat⟩ | (⟨SNat⟩)*(⟨SNat⟩)
```

La semántica pretendida interpreta las fbfs de la forma (⟨SNat⟩*⟨SNat⟩) con el producto de las interpretaciones de los ⟨SNat⟩ que aparezcan. Por ejemplo (SS0+SSS0)*(SSS0) debe representar el producto (2+3)*(3).

1.1.3

1 Además de lo ya definido para SNat se pueden incluir en el cálculo deductivo unas reglas para calcular multiplicaciones:
```
[*1]   (x)*(0) ⊢ 0
```

```
[*2]   (x)*(Sy) ⊦ x+(x)*(y)
[*3]   (x)*(y+z) ⊦ (x)*(y)+(x)*(z)
[*4]   (y+z)*(x) ⊦ (y)*(x)+(z)*(x)
```

2 El axioma `MI` tiene una `I`, de modo que este teorema no tiene un número de `Is` divisible por `3`. Si se llega a una palabra con esta propiedad, puede observarse que, si alguna regla de derivación es aplicable, el número de `Is` en la palabra derivada tampoco es divisible por `3`. Como en `MU` el número de `Is` es `0`, y `0` es divisible por `3`, `MU` no es un teorema.