1)

a) Um inteiro tem 4 bytes. A função dump mostra os valores byte a byte. 10000 em binário fica 0000 0000 0000 0000 0010 0111 0001 0000 ; seguindo do byte menos significativo para o mais significativo, os valores em hexadecimal são 10 27 00 00

b) Como o long tem 8 bytes, a função dump mostra os mesmos valores que mostrou para o int mais 00 00 00 00.

c) Como o short tem 2 bytes, a função dump mostra apenas os dois bytes menos significativos, 10 e 27.

d) O char tem 1 byte, então a função dump mostra o valor em hexadecimal desse único byte. No caso do ‘a’ ou 97, o valor é 61.

c) ‘A’: valor em hexadecimal = 41 -> inteiro correspondente na tabela ASCII = 69 (ou seja, o valor de 41 na base 10)

‘ ‘: valor em hexadecimal = 20 -> inteiro correspondente na tabela ASCII = 32

‘,’: valor em hexadecimal = 2c -> inteiro correspondente na tabela ASCII = 44

‘\n’: valor em hexadecimal = 0a -> inteiro correspondente na tabela ASCII = 10

‘$’: valor em hexadecimal = 24 -> inteiro correspondente na tabela ASCII = 36

2)

a) o valor de \*s – ‘0’ será o caractere que \*s aponta, mas como um inteiro ao invés de char, porque os valores inteiros que correspondem aos caracteres ‘0’..’9’ na tabela ASCII são 48..57; então, ao fazer ‘8’ – ‘0’, por exemplo temos que: ‘8’ – ‘0’ = 56 – 48 = 8, e isso vale para todos os caracteres ‘0’..’9’.

Portanto:

1 iteração: (\*s – ‘0’) = ‘1’ – ‘0’ = 1 ; a = 0 + 1

2 iteração: (\*s – ‘0’) = ‘2’ – ‘0’ = 1 ; a = 10 + 2 = 12

3 iteração: (\*s – ‘0’) = ‘3’ – ‘0’ = 3 ; a = 120 \* 3 = 123

4 iteração: (\*s – ‘0’) = ‘4’ – ‘0’ = 4; a = 1230 + 4 = 1234

c) A maior base que pode ser usada é 36.