Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
CSF13 – Fundamentos de Programação 1
Prof. Bogdan Tomoyuki Nassu
Profa. Leyza Baldo Dorini

1.
Binário:
a) 1111 1010
b) 0011 1111
c) 0100 0000
d) 0111 1111
e) 1000 0000
f) 1111 1111
g) 0001 0000 0000
h) 0001 1110 00011

Hexadecimal:
a) 0xFA
b) 0x3F
c) 0x40
d) 0x7F
e) 0x80
f) 0xFF
g) 0x0100
h) 0x1E3

2.
Binário:
a) 1111 1000
b) 1010 1011
c) 0001 0110
d) 0111 0101
e) 1111 1111

3.

a)
(10000101 & 01111010) | 00001101
00000000 | 00001101
00001101
Resultado: 0x0D

b)
!foo é igual 1 se foo for igual a 0, ou 0 do contrário. Como foo tem valor 3 (lembre-se que o tipo char é um tipo inteiro), !foo tem valor 0.

c)
Não existe número que seja ao mesmo tempo menor ou igual a 10 e maior que 10. Portanto, o resultado desta expressão é 0, para qualquer valor de x.

d) (200 | 6) ^ (30 & 0x08)
(11001000 | 00000110) ^ (00011110 & 00001000)
11001110 ^ 00001000
11000110
Resultado: 0xC6, ou 198

4.

```
/* O código para este programa é bem simples. Tudo o que precisamos lembrar é
   que o último bit de um número par é 0, e de um número ímpar é 1. */

#include <stdio.h>

int main ()
{
    int n;

    scanf ("%d", &n);

    if (n & 0x01)
        printf ("impar\n");
    else
        printf ("par\n");

    return (0);
}
```

5.

a) (as interrogações são respondidas na letra b)
? ? ?
67 67 88
11000010
-24 67 1000

b) As variáveis não estão sendo inicializadas antes do primeiro printf. Esquecer de inicializar variáveis é um dos erros mais comuns entre principiantes.

c) O %d imprime o número na base 10. O valor 88 é o correspondente na base 10 ao número hexadecimal 0x58.

d) TODAS as variáveis são SEMPRE armazenadas usando representação binária.

e) O que é impresso é a sequência binária correspondente ao número 67... mas ao contrário! Existe um algoritmo para converter números decimais em binários que consiste exatamente em realizar sucessivas divisões por 2, analisando o resto dessas divisões. Este algoritmo não foi descrito porque é muito fácil ceder à tentação de usá-lo mecanicamente, sem compreender a razão do seu funcionamento. Procure refletir sobre a razão pela qual este algoritmo funciona.

f) O que acontece é um overflow. O tipo char só tem 1 byte, e não é capaz de armazenar o valor 1000.

6.

A chave para compreender esta questão é o fato de que os aarghs trabalham com 5 "símbolos" (posições para os tentáculos). Isso é equivalente a dizer que os aarghs contam em base 5. Assim como na base 10 cada dígito é associado a uma potência de 10, e na base 2 cada bit é associado a uma potência de 2, na base 5 cada "dígito" (ou tentáculo, no caso) é associado a uma potência de 5.

a)

1 ● ● ↑
2 ● ● →
3 ● ● ↓
4 ● ● ←
5 ● ↑ ●
6 ● ↑ ↑
7 ● ↑ →
8 ● ↑ ↓
9 ● ↑ ←
10 ● → ●

b) 42 = 1*25 + 3*5 + 2 = ↑ ↓ →

c) 100 = 4*25 = ← ● ●

d) ↑ ↑ ↑ = 1*25 + 1*5 + 1 = 31

e) → → ↑ = 2*25 + 2*5 + 1 = 61

f) 124 (é 5^3-1).