PUC-Rio – Software Básico – INF1018
Prova 1 – 26/04/12 – Turma 3WA

1. (2,5 pontos) Considere o programa C a seguir:

```
#include <stdio.h>
void dump (void *p, int n) {
  unsigned char *p1 = (unsigned char *) p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}
struct sno {
  short sval;
  int ival;
  struct sno *prox;
  char cval;
};
int main (void) {
  struct sno no[3];
  no[0].sval = 0x1a & -5;
  no[0].ival = -3 | 0x11;
  no[0].prox = &(no[2]);
  no[0].cval = -68;
  dump ((void *)no, sizeof(struct sno));
  return 0;
}
```

   Considerando que `no` será alocado na posição de memória 0xbffff7b8, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (ATENÇÃO: valores sem contas e explicações NÃO valem ponto!!!). Suponha que a máquina de execução é *little-endian* e que as convenções de alinhamento são as do Linux no IA-32. Se houver posições de *padding*, indique seu conteúdo com `PP`.

2. Traduza as funções `foo` e `boo` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Traduza o mais diretamente possível o código de C para assembly. Use registradores para abrigar as variáveis locais que aparecem nessas funções. Comente seu código.

   (Não se preocupe se você não souber o que as funções fazem, apenas traduza-as literalmente!)

   (a) (2,5 pontos)

```
int foo (int a[], int n) {
  if (n==0)
    return 0;
  else
    return a[0] + foo(&(a[1]), n-1);
}
```

(b) (2,5 pontos)

```
struct X {
  short v;
  struct X *prox;
};

void boo (struct X *x) {
  int s = 0;
  while (x!=NULL) {
    s = s + x->v;
    if (((x->v)==0) && s>10)
      x->v = s;
    x = x->prox;
  }
}
```

3. (2,5 pontos) No formato UTF-8, usado para representar caracteres UNICODE, caracteres têm tamanho variável. Um caracter tem tamanho mínimo de 8 bits, porém, se seu código necessitar de mais espaço para ser representado, o formato UTF-8 pode utilizar mais de um byte para representá-lo (até um máximo de quatro bytes). Abaixo é apresentada a faixa dos números e o número de bytes necessários para representar cada faixa do código UNICODE (onde x é o valor real do bit):

```
Código UNICODE          Representação UTF-8
U+0000 - U+007F            0xxxxxxx
U+0080 - U+07FF            110xxxxx 10xxxxxx
U+0800 - U+FFFF            1110xxxx 10xxxxxx 10xxxxxx
U+10000 - U+10FFFF         11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Escreva, em C ou em assembly IA-32 usado nos laboratórios, uma função *utf8strlen*, com o protótipo:

```
int utf8strlen (FILE *entrada);
```

Essa função recebe um arquivo (já aberto) contendo uma sequência de caracteres UTF8 e retorna seu comprimento *em caracteres*, ou seja, o número de caracteres que estão representados no arquivo.