PUC-Rio – Software Básico – INF1018
Prova Final – 11/12/2018

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 X {
   int i;
   char c;
   double d;
   float a;
  } x = {-12, 'e', -3.5, 16.75};

int main (void) {
   dump (&x, sizeof(struct X));
   return 0;
}
```

   Supondo que x seja armazenado no endereço de memória 0x601040, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores. Considere que a máquina de execução é *little-endian*, que as convenções de alinhamento são as do Linux no IA-64 e que o valor do caracter 'a' na tabela ASCII é 97, em decimal. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

2. (2,0 pontos) Construa uma função C que converte um array de inteiros sem sinal para uma representação *big endian*. A função recebe um array de inteiros sem sinal, o tamanho desse array e um buffer de tamanho suficiente para armazenamento do resultado. A função deve preencher o buffer com a mesma sequência de inteiros do array original, porém com cada inteiro armazenado em *big endian*. Suponha, se necessário, que sua função vai executar em uma máquina *little-endian*.

```
void toBig (unsigned int *origem, int tam, unsigned char *dest)
```

3. Traduza as funções `vaimais` e `boo` abaixo para assembly IA-64 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/linux. Traduza o mais diretamente possível o código de C para assembly. (Não se preocupe em entender o que cada função faz, apenas traduza-as literalmente.)

   (a) (2,5 pontos)

```
int ajuste (int i);

void vaimais (int *v, int n) {
   int i;
   for (i=0; i<n; i++) {
      v[i] = v[i] + ajuste(v[i]);
   }
}
```

   (b) (3,0 pontos)

```
struct T {
   char c;
   double d;
   struct T *prox;
};

double foo(double d);

double boo(struct T *boa, float bias) {
   double soma = 0.0;
   while (boa) {
      soma += boa->d + bias;
      boa = boa->prox;
   }

   return foo(soma);

}
```