PUC-Rio – Software Básico – INF1018
Prova 1 – Turma 3wb – 16/10/2018

1. (3,0 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 S {
  short cs;
  int   ci;
} s = {0x55aa, -2050};

struct S1 {
  struct S *ps;
  long  cl1;
  short cs1;
} s1;

int main (void) {
  s1.ps = &s;
  s1.cl1 = (long) s.ci;
  s1.cs1 = (s.cs >> 8) & 3;
  dump (&s1, sizeof(s1));
  return 0;
}
```

Sabendo das informações abaixo e supondo que a máquina de execução é *little-endian* com as convenções de alinhamento do Linux no IA-64 (vistas em sala), **mostre o que esse programa irá imprimir quando executado**. Coloque **PP** nas posições correspondentes a *padding*. (Justifique os valores exibidos: valores sem contas **NÃO** valem ponto!).

endereço de `s` na memória	0x55d060f70010
endereço de `s1` na memória	0x55d060f70030

2. Traduza as funções `foo` e `boo` abaixo para assembly IA-64, utilizando as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/Linux. (Não se preocupe em entender o que as funções fazem, apenas traduza-as literalmente.)

   Atenção! **Traduza o mais diretamente possível o código de C para assembly**.

   (a) (2,5 pontos)

```
int foo(long v1[], int *pi, int n) {
  int i;
  for (i = 0; i < n; i++, pi++) {
     if (*pi < n) {
       v1[*pi] = 0;
     }
  }
}
```

(b) (2,5 pontos)

```
#define NULL 0

struct X {
  int val;
  long valc;
  struct X *next;
};

int abs(int j);
long calc(int x);

void boo(struct X *px, int *v) {
  while (px != NULL) {
    px->val = *v;
    px->valc = calc(abs(*v));
    px = px->next;
    v++;
  }
}
```

3. (2,0 pontos) No nosso primeiro trabalho, utilizamos *strings* para descrever as estruturas, utilizado os caracteres abaixo para cada tipo de campo:

   - 'c' - char
   - 's' - short int
   - 'i' - int
   - 'l' - long int

   Por exemplo, dada a declaração

   ```
   struct X {
     char c;
     long l;
     int i;
   }
   ```

   a *string* que descreve a estrutura será `"cli"`.

   Escreva uma função que receba a descrição de uma estrutura e retorne o tamanho necessário (número de bytes) para armazenar essa estrutura na memória. Esse tamanho deverá atender os requisitos de alinhamento para um sistema Linux executando numa plataforma IA-64 (ou seja, o tamanho deve incluir os bytes de *padding* necessários para os alinhamentos dos campos).

   O protótipo da função é dado a seguir:

   ```
   int tamanho_struct(char *campos);
   ```

Boa Prova!