PUC-Rio – Software Básico – INF1018
Prova 1 – Turma 3WA – 10/10/2017

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 Ficha {
   char i;
   struct Ficha *v;
   int p;
};
int main (void) {
   struct Ficha f[2] = {{'e', f, -3}, {3 << 4, &f[0], 1028}};
   dump (&f, sizeof(struct Ficha)*2);
   return 0;
}
```

   Descreva o que este programa irá imprimir quando executado, *justificando* os valores exibidos mostrando as contas e outras informações usadas para chegar ao resultado. Suponha que a máquina de execução seja *little-endian* com as convenções de alinhamento do Linux no IA-64 vistas em sala. Além disto considere que `f[0]` seja alocado na posição de memória 0x7fffe717c640 e que o valor do caracter 'a' na tabela ASCII seja 97 (decimal). Coloque **PP** nas posições correspondentes a *padding*.

2. (2,5 pontos) Implemente em C uma função *gravachars* que armazena apenas os elementos do tipo char de uma struct.
   `void gravachars(void* valorStruct, char* descritor, char* saida)`

   A função *gravaschars* recebe um ponteiro para a struct *valorStruct*, uma descrição dos campos desta struct em *descritor* e um ponteiro para um vetor de chars *saida* para gravação dos chars da struct seguidos do valor 0x00 (zero). A cadeia descritor representa o tipo de cada um dos campos da struct (na ordem) que podem ser um char ou um inteiro apenas, mais precisamente:
   'c' - descrevendo um char
   'i' - descrevendo um inteiro
   Por exemplo, para a struct seguinte

```
struct Alguma
{
   char c1;
   int i;
   char c2
} ss = {'b', 2, 'c'};
```

   Uma chamada a *gravachars* passando o descritor *"cic"* grava na saída o vetor com os dois chars 'b' e 'c', seguidos de 0x00 (zero). Considere que o espaço em memória para o vetor saida já está reservado antes da chamada à função *gravachars*.

3. Traduza as funções `bum` e `fooba` 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 as funções fazem, apenas traduza-as literalmente.)

   Comente seu código!

   (a) (2,5 pontos)

```
int mod(int i);

int bum(int a[], int n, int t)
{
   int r = 0; int i = 0;
   for (i=0; i< n; i++)
   {
      if (a[i] == t)
      r = r + mod(a[i]);
   }
   return r;
}
```

   (b) (2,5 pontos)

```
struct Cad
{
   int val;
   struct Cad* op1;
   struct Cad* op2;
};

int fooba(struct Cad* v)
{
   if (v == 0)
      return 0;
   else
   {
      if (v->val < 0)
        return 1 + fooba(v->op1);
      else
        return 1 + fooba(v->op2);
   }
}
```

Boa Prova!