PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3WA – 29/11/2018

1. (2,5 pontos)

```
#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 {
   float f;
   double d;
   char c;
   float a;
} x = {-5.625, 1024.5, -5, 4.6};

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*, e que as convenções de alinhamento são as do Linux no IA-64. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

2. (1,5 pontos) Considere o seguinte código em C:

```
#include <stdio.h>

int n = 1;
int mal (int t);
int boa (int t);

int verif(int val) {
   int r;
   if (val > 0) r = mal(val);
   else r = boa(val);

   return r;
}

int geracc(char off, int cum, int *v) {
   int i; int tot = cum;
   n += off;
   printf("n = %d\n", n);
   for (i=0; i<n; i++)
      tot += verif(v[i]);

   return tot;
}
```

Liste quais símbolos do módulo objeto do código acima seriam importados e exportados, ou seja apareceriam como **D** (símbolo na área de dados inicializados), **T** (símbolo na área de código) e **U** (símbolo indefinido) na saída do programa `nm`.

3. Traduza as funções `acumula` e `hist` abaixo para assembly IA-64 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados de C/Linux. Traduza o mais diretamente possível o código de C para assembly e comente seu código.

   (a) (2,5 pontos)

```
double verif(double v);

double acumula (int n, double *vals) {
   double t = 0.0; int i;

   for (i=0; i < n; i++) {
      if (verif(vals[i]))
      t += vals[i];
   }
   return t;
}
```

   (b) (2,5 pontos)

```
double coef(int r);

double hist(int n, int *v)
{
   int i, j; int h[10];
   double pesof = 0.0;

   for (i=0; i<10; i++) h[i] = 0;

   for (j=0; j<n; j++) h[v[j]] += 1;

   for (i=0; i<10; i++) pesof += h[i] * coef(i);

   return pesof;
}
```

4. (1,0 ponto) Escreva o código assembly equivalente ao código de máquina que a função *gera_codigo* do segundo trabalho de software básico geraria para o trecho de código *SBF* a seguir:

```
function
v0 = p0 + $1
ret v0
end
```

Boa Prova!