PUC-Rio – Software Básico – INF1612
Prova Final – 29/6/06

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 um {
  double f;
  char c;
  int i;
  short s;
};
struct um a =  {39.375, 21, -2150, -33};
int main (void) {
  dump (&a, sizeof(a));
  return 0;
}
```

   Considerando que `a` seja alocado na posição de memória `0xbffffcd0`, diga o que esse programa irá imprimir quando executado, explicando como você chegou aos valores exibidos (*mostre suas contas!!*). Suponha que a máquina de execução é Pentium-Linux, ou seja, a representação de dados é a vista no curso.

2. (2,5 pontos) A codificação *uuencode* foi muito utilizada para transmitir arquivos binários como texto. A codificação tem algumas etapas, mas a principal delas consiste na própria criação de sequências de caracteres ASCII. Cada sequência de 24 bits é transformada em 4 caracteres de 8 bits, ASCII, seguindo a seguinte regra. A sequência de 24 bits, é dividida em 4 grupos de 6 bits, cada um deles representando um número entre 0 e 63. Soma-se o decimal 32 a cada um desses números e cria-se um caracter de 8 bits contendo o valor resultante.

   Escreva uma função em C que realiza essa conversão. A função deve ter o seguinte cabeçalho:

```
void codifica (char *binario, char* buffer);
```

   onde `binario` é um ponteiro para uma sequência de três bytes (24 bits) a ser transformada e `buffer` é um ponteiro para uma região previamente alocada, onde devem ser depositados os quatro caracteres ASCII resultantes.

   Como exemplo do comportamento de `codifica`, suponha que seja feita a chamada:

```
char entrada[3] = {'C', 'a', 't'};
char buffer[4];
codifica (entrada, buffer);

        C           a           t
    0100 0011 | 0110 0001 |  0111 0100     valores recebidos como entrada
      separa em grupos de seis bits:
    010000 110110    000101    110100
      expande para 8 bits:
    00010000 00110110  00000101 00110100
      soma 32:
    00110000 01010110  00100101 01010100   valores retornados na saída
```

(*DICA:* Antes de começar a separar os caracteres, junte os 24 bits iniciais em um inteiro!)

3. Traduza as funções `bar` e `foo` abaixo para assembly IA-32 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados em C. Comente seu código.

   (a) (2,5 pontos)

```
struct X {
  int val;
  struct X * next;
};
int max (struct X * lista) {
  int lm;
  if (lista == 0)
    return 0
  else {
    lm = max (lista-> next))
    if (lm < lista->val)
      return lista->val;
    else
      return lm;
  }
}
```

   (b) (2,5 pontos)

```
struct X {
  double val; int prox;
};
double soma (struct X lista[], int inicio) {
  int corr = inicio; double soma = 0.0;
  while (corr != 0) {
    soma += lista[corr].val;
    corr = lista[corr].prox;
  }
  return soma;
}
```