Nome:			RA:				
Assinatura:	1	2	3	4	5		
MC102 6+0 **Prova 2 GABARITO 2007a** **Prof. Rogério Drummond**							

Use lapis de preferência. Envolva a resposta de cada questão com um retângulo e indentifique claramente a questão. Nenhum outro papel além desta prova é permitido durante a aplicação deste teste. Calculadoras, telefones, radios, etc também não são permitidos.

1. Implemente a função `parImpar()` que dado um vetor `v[]` de inteiros terminado pelo valor `0` (zero), copia os valores pares para o vetor `par[]` e os valores impares para o vetor `impar[]`. Os vetores `par[]` e `impar[]` devem ser terminados pelo valor `0` (zero).

```
void parImpar(int v[], int par[], int impar[]) {
    int k=0, i=0, p=0;
    while (v[k] != 0)
        if (v[k]%2 == 0)
            par[p++] = v[k++];
        else
            impar[i++] = v[k++];
    par[p] = impar[i] = 0;
}
```

2. Implemente a função `embaralha()` que dado dois vetores `v1[]` e `v2[]` de `n` inteiros, copia os valores de `v1` e `v2` para o vetor `r[]` de forma que `r[]` contenha os valores alternados de `v1` e `v2`. O tamanho final de `r[]` será `2*n`.

```
void embaralha(int v1[], int v2[], int n, int r[]) {
    int i;
    for (i=0; i<n; i++) {
        r[2*i] = v1[i];
        r[2*i+1] = v2[i];
    }
}
```

3. Implemente a função `calcMedia()` que dado os vetores `nota1[]`, `nota2[]` e `sub[]` de `n` reais, calcula a média das notas no vetor `media[]`. Cada índice contêm as informações de um aluno. A média é calculada com a `nota1` tendo peso `1` e a `nota2` tendo peso `2`. A `sub` é substitutiva da `nota2` e pode-se escolher o maior valor entre `nota2` e `sub`.

```
void calcMedia(float nota1[], float nota2[], float sub[], float media[], int n){
    int i;
    for (i=0; i<n; i++)
        if (nota2[i] > sub[i])
            media[i] = (nota1[i] + 2*nota2[i]) / 3;
        else
            media[i] = (nota1[i] + 2*sub[i]) / 3;
}
```

Usando o operador `?:`, a solução seria:

```
void calcMedia(float nota1[], float nota2[], float sub[], float media[], int n){
    int i;
    for (i=0; i<n; i++)
        media[i] = (nota1[i] + 2*(nota2[i]>sub[i] ? nota2[i] : sub[i])) /3;
}
```

4. Implemente a função `invertev()` que dado um vetor `v[]` de tamanho `n`, inverte seus elementos. Você não pode usar outro vetor como auxiliar. Após a chamada `invertev(ra[], 3)` onde `ra[0]=100`, `ra[1]=20` e `ra[2]=5`; o vetor `ra[]` teria os seguintes valores: `ra[0]=5`, `ra[1]=20` e `ra[2]=100`.

```
void invertev(int v[], int n) {
    int i=0; n--;
    while (i < n)
        trocav(v, i++, n--);
}

void trocav(int v[], int a, int b) {
    int aux = v[a];
    v[a] = v[b];
    v[b] = aux;
}
```

```
Solução sem usar a função trocav():
void invertev(int v[], int n) {
    int i=0, aux;
    n--;
    while (i < n) {
        aux = v[i};
        v[i++] = v[n];
        v[n--] = aux;
}
```

5. Escolha uma das funções recursivas abaixo e mostre de forma clara quantas vezes ela é chamada quando o parâmetro `n` da chamada inicial é `7`. O número total de chamadas sem descrição de como se chegou a este valor não tem qualquer significado.
Faça esta questão por último. Tem uma forma de resolver na força bruta que dá MUITO trabalho e que não dá para resolver na prova. Pense bem antes de começar a resolver esta questão.

5A.
```
int fib(int n) {
    if (n<2)
        return n;
    return fib(n-2) + fib(n-1);
}
```

5B.
```
void hannoi(int a int b, int c, int n) {
  if (n==1)
    printf("Move de %d para %d\n", a, b);
  else {
    hannoi(a, c, b, n-1);
    hannoi(a, b, c, 1);
    hannoi(c, b, a, n-1);
  }
}
```

A chave para a solução desta questão é definir de forma recorrente a fórmula do número de chamadas, e aplicá-la para valores crescentes de `n`. A forma da força bruta é seguir operacionalmente a execução da função, que resulta numa árvore de chamadas exponencial. A questão foi proposta para que vocês tivessem a oportunidade de descobrir e raciocinar com estratégias não operacionais.

Seja `nf(n)` e `nh(n)` funções que determinam o número de chamadas de `fib(n)` e `hannoi(…, n)`. Note que os 1ºs parâmetros de `hannoi()` não tem qualquer influência no número de chamadas.

```
nf(n) = 1, se n=0
        1, se n=1
        1 + nf(n-2) + nf(n-1), se n>1
```
No caso geral (`n>1`), são 2 chamadas com `n-2` e `n-1`, mais a chamada original para `n`.

```
nh(n) = 1, se n=1
        2 + 2 * nh(n-1), se n>1
```
No caso geral (`n>1`), são 2 chamadas com `n-1`, uma chamada com `n=1` (que custa `1`), mais a chamada original para `n`.

A tabela abaixo é construída iniciando com `n=0` e depois com valores crescentes de `n`.

n	nf(n)	nf(n)	nh(n)	nh(n)
0	1	1	--	--
1	1	1	1	1
2	1+1+1	3	2+2*1	4
3	1+1+3	5	2+2*4	10
4	1+3+5	9	2+2*10	22
5	1+5+9	15	2+2*22	46
6	1+9+15	25	2+2*46	94
7	1+15+25	41	2+2*94	190