Ordenação por intercalação: *MergeSort*

Considere a implementação em C da função classificação por trocas *MergeSort* para resolver os exercícios a seguir.

```
void intercala(int p, int q, int r, int v[]){
     int i, j, k, *w;
     w = malloc( (r - p) * sizeof (int) );
     i = p;
     j = q;
     k = 0;
     while (i < q && j < r){
           if (v[i] <= v[j])
              w[k++] = v[i++];
           else
               w[k++] = v[j++];
     }
     while (i < q) w[k++] = v[i++];
     while (j < r) w[k++] = v[j++];
     for (i = p; i < r; i++)
         v[i] = w[i - p];
     free(w);
}

void ordenarMergeSort(int p, int r, int v[]){
     if (p < r - 1){
        int q = (p + r) / 2;
        ordenarMergeSort(p, q, v);
        ordenarMergeSort(q, r, v);
        intercala(p, q, r, v);
     }
}

int main(){
    int numeros[] = {5, 22, 7, 9, 25};
    ordenarMergeSort(0, 5, numeros);
}
```

01. Considere a função `ordenarMergeSort`.

a) Qual a sequencia de invocações se tivermos um vetor com 4 elementos.

b) Qual a sequencia de invocações se tivermos um vetor com 5 elementos.

02. O algoritmo *MergeSort* é **estável**? *Lembrando que um algoritmo de ordenação estável é aquele que não altera a ordem relativa das chaves iguais.*

( ) Sim ( ) Não

03. Qual o melhor caso para o algoritmo *MergeSort*?

04. Compare o pior caso com o caso médio do algoritmo *MergeSort*. Existe muita diferença entre esses dois casos?

05. O que acontece com a execução com a execução do *MergeSort* quando todos os elementos do vetor possuem o mesmo valor?

06. Explique sucintamente o que acontece com as seguintes implementações da função ordenarMergeSort:

Algoritmo	Explicação

`void mergesort1 (int p, int r, int v[])`
`{`
`  if (p < r-1) {`
`  int q = (p + r) / 2;`
`  mergesort1 (p, q, v);`
`  mergesort1 (q, r, v);`
`  intercala (p, q+1, r, v);`
`  }`
`}`	

`void mergesort2 (int p, int r, int v[])`
`{`
`  if (p < r) {`
`  int q = (p + r) / 2;`
`  mergesort2 (p, q, v);`
`  mergesort2 (q, r, v);`
`  intercala (p, q, r, v);`
`  }`
`}`	

`void mergesort3 (int p, int r, int v[])`
`{`
`  if (p < r-1) {`
`  int q = (p + r - 1) / 2;`
`  mergesort3 (p, q, v);`
`  mergesort3 (q, r, v);`
`  intercala (p, q, r, v);`
`  }`
`}`	

`void mergesort4 (int p, int r, int v[])`
`{`
`  if (p < r-1) {`
`  int q = (p + r) / 2;`
`  mergesort4 (p, q-1, v);`
`  mergesort4 (q-1, r, v);`
`  intercala (p, q-1, r, v);`
`  }`
`}`	