Q1.

(*i*)
```
/* Assumindo-se n >= 0 */
int fr( int n )
{
    if (n <= 1) return n;
    else return fr(n-1) + fr(n-2);
}
```

(*ii*)
```
int fib( int n)
{
    int i, z, x = 1, y = 0;

    for (i = 1; i <= n; i++)
    {
        z = y;
        y = x + y;
        x = z;
        /* Neste ponto y é o i-ésimo número de Fibonnaci */
       /* e x é o (i-1)-ésimo número de Fibonacci. */
    }
    return y;
}
```

(*iii*) A versão iterativa é a mais rápida.

(*iv*) Pois na versão recursiva o mesmo subproblema é resolvido diversas vezes.

$\|$

Q2.

$(1) \equiv \texttt{mergesort}(\langle\texttt{M,E,R,G,E,z}\rangle)$ devolve $(13) \equiv \langle\texttt{E,E,G,M,R,z}\rangle$

  $(2) \equiv \texttt{mergesort}(\langle\texttt{M,E,z}\rangle)$ devolve $(5) \equiv \langle\texttt{E,M,z}\rangle$

    $(3) \equiv \texttt{mergesort}(\langle\texttt{M,z}\rangle)$ devolve $\langle\texttt{M,z}\rangle$

    $(4) \equiv \texttt{mergesort}(\langle\texttt{E,z}\rangle)$ devolve $\langle\texttt{E,z}\rangle$

    $(5) \equiv \texttt{merge((3), (4))}$ devolve $\langle\texttt{E,M,z}\rangle$

  $(6) \equiv \texttt{mergesort}(\langle\texttt{R,G,E,z}\rangle)$ devolve $(12) \equiv \langle\texttt{E,G,R,z}\rangle$

    $(7) \equiv \texttt{mergesort}(\langle\texttt{R,z}\rangle)$ devolve $\langle\texttt{R,z}\rangle$

    $(8) \equiv \texttt{mergesort}(\langle\texttt{G,E,z}\rangle)$ devolve $(11) \equiv \langle\texttt{E,G,z}\rangle$

      $(9) \equiv \texttt{mergesort}(\langle\texttt{G,z}\rangle)$ devolve $\langle\texttt{G,z}\rangle$

      $(10) \equiv \texttt{mergesort}(\langle\texttt{E,z}\rangle)$ devolve $\langle\texttt{E,z}\rangle$

      $(11) \equiv \texttt{merge((9), (10))}$ devolve $\langle\texttt{E,G,z}\rangle$

    $(12) \equiv \texttt{merge((7), (8))}$ devolve $\langle\texttt{E,G,R,z}\rangle$

  $(13) \equiv \texttt{merge((2), (6))}$ devolve $\langle\texttt{E, E, G, M, R, z}\rangle$

||

Q3.

(i)

```
         9
        / \
       5   6
      / \ / \
     1  4 2  3
```

(ii)

```
         6
        / \
       5   3
      / \ /
     1  4 2
```

(iii) Pois a altura de um *heap* de n elementos é $O(\lg n)$.

(iv) Para a criação do *heap* são realizadas $\lfloor n/2 \rfloor$ iterações cada uma executanto $O(\lg n)$ comparações e para ordenar o vetor resultante são realizadas $n-1$ iterações cada uma executando $O(\lg n)$ comparações. Portanto, o número total de comparações para ordenar um vetor de n elementos é $O(n \lg n)$.

Q4.

```
Elemento *produto( Elemento *p, Elemento *q )
{
    Elemento inicio;
    Elemento *final = &inicio, *r, *s;

    inicio.prox = NULL;
    /* inicio.prox aponta para o ínicio da lista que contém o produto de p e q */
    for (r = p; r; r = r->prox)
        /* final aponta para o final da lista cuja cabeça é inicio.prox */
        for (s = q; s; s = s->prox)
        {
            final->prox = (Elemento *)malloc(sizeof(Elemento));
            final = final->prox;
            final->chave = (char *) malloc(sizeof(char) *
                           (strlen(r->chave) + strlen(s->chave) + 1));
            strcpy(final->chave, r->chave);
            strcat(final->chave, s->chave);
            final->prox = NULL;
        }
    return inicio.prox;
}
```

Q5.

(*i*) Como `v = a[r]` e `ind_particao` só é chamada se `l < r`, segue que, o índice `i` em `ind_particao` satisfaz `i` $\leq$ `r`. Considere a seguinte chamada `quicksort(a, 1, n)` com o vetor `a` satisfazendo `a[n] < a[j]` para todo `j`, `1` $\leq$ `j < n`. Então, como o valor inicial de `j` em `ind_particao` é `r` e `v = a[r]`, o laço `while (a[--j]>v)` poderá acessar um elemento fora do vetor. Daí a necessidade de uma sentinela. Fazendo-se, então, `a[0] = INT_MIN` garantimos que `0` $\leq$ `j < n` e estamos nos limites do vetor. Mais ainda, devido a corretude de `ind_particao` (que mostraremos em (*ii*)), segue que, `l-1` $\leq$ `j < r` em qualquer chamada recursiva de `QS(a, l, r)` com `l < r`.

```
void quicksort(int a[], int l, int r)
{
    a[0] = INT_MIN;
    QS(a, l, r);
}

void QS(int a[], int l, int r)
{
    int i;

    if (l < r) {
        j = ind_particao(a, l, r);
        QS(a, l, i - 1);
        QS(a, i + 1, r);
    }
}
```

(*ii*) Vamos mostrar que no ínicio de cada iteração do laço `while (1)` a seguinte propriedade é válida:

$$\texttt{a[k]} \leq \texttt{v} \leq \texttt{a[m]} \text{ para } \texttt{l} \leq \texttt{k} \leq \texttt{i} \text{ e } \texttt{j} \leq \texttt{m} \leq \texttt{r} \tag{1}$$

No ínicio da primeira iteração `i = l - 1` e `j = r` e (1) é válida, pois o primeiro intervalo é vazio e o segundo contém somente `r` e `v = a[r]`. Suponha que (1) seja válida no início de uma iteração. Sejam `i'` e `j'` os valores de `i` e `j` imediatamente após a execução dos laços `while (a[++i] < v)` e `while (a[--j] > v)` respectivamente. Como observado no item (*i*) temos que `i'` $\leq$ `r` e `j'` $\geq$ `l - 1` e as seguintes relações são válidas:

$$\texttt{a[k]} \leq \texttt{v} \text{ para } \texttt{l} \leq \texttt{k} \leq \texttt{i},\ \texttt{a[k]} < \texttt{v} \text{ para } \texttt{i + 1} \leq \texttt{k} < \texttt{i'} \text{ e } \texttt{a[i']} \geq \texttt{v} \tag{2}$$

$$\texttt{a[k]} \geq \texttt{v} \text{ para } \texttt{j} \leq \texttt{k} \leq \texttt{r}, \texttt{a[k] > v} \text{ para } \texttt{j' < k} \leq \texttt{j - 1} \text{ e } \texttt{a[j']} \leq \texttt{v} \qquad (3)$$

Temos dois casos:

Caso 1 `i' < j'`.

Seja `a'` o vetor obtido de `a` trocando-se os elementos `a[i']` e `a[j']`. Então, devido à (2) e (3), (1) vale com `a'` no lugar de `a`, `i'` no lugar de `i` e `j'` no lugar de `j`.

Caso 2 `i' = j'`.

Então `a[i'] = a[j'] = r` e devido a (2) e a (3), no final do algoritmo, o vetor `a` e o índice `i'` satisfazem

$$\texttt{a[k]} \leq \texttt{a[i']} \leq \texttt{a[m]} \text{ para } \texttt{l} \leq \texttt{k < i'} \text{ e } \texttt{i' < m} \leq \texttt{r}$$

Caso 3 `i' > j'`

Então, devido a (2) e (3), temos que, `j' = i' - 1`, `a[j'] < v` e `a[i']` $\geq$ `v`. Seja `a'` o vetor obtido trocando-se `a[i']` e `a[r]`. Então, no final do algoritmo, `a'` e `i'` satisfazem

$$\texttt{a'[k]} \leq \texttt{a'[i']} \leq \texttt{a'[m]} \text{ para } \texttt{l} \leq \texttt{k < i'} \text{ e } \texttt{i' < m} \leq \texttt{r}$$

(*iii*) Como dito em (*i*) o caso em questão é aquele em que `a[n] < a[j]` para todo `j, l` $\leq$ `j < n` numa chamada `quicksort(a, 1, n)`.

$\|$