

Obs. Nas questões abaixo, suporemos que temos

```
typedef struct node *link;
struct node { int item; link next; };
```

1. [3 pontos]

- (a) Descreva precisamente o efeito da atribuição `x = delete(x,v)`, onde `x` aponta para a primeira célula de uma lista ligada (sem cabeça e sem cauda) e `v` é um inteiro dado, onde a função `delete()` é implementada como segue.

```
link delete(link x, int v)
{
    if (x == NULL) return NULL;
    if (x->item == v)
        { link t = x->next; free(x); return t; }
    x->next = delete(x->next, v);
    return x;
}
```

- (b) Altere o código em (a) apenas levemente para que todas as ocorrências de `v` sejam eliminadas.

2. [3 pontos] Considere as seguintes funções:

```
int partition(int a[], int l, int r)
{ int i = l-1, j = r; int v = a[r];
  for (;;) {
    while (a[++i] < v) ;
    while (v < a[--j]) if (j == l) break;
    if (i >= j) break;
    exch(a[i], a[j]);
  }
  exch(a[i], a[r]);
  return i;
}

void print(int a[], int l, int r, int i)
{
  int k;
  printf("%3d: %2d %3d %3d\n", a[i], l, i, r);
  for (k = 0; k < l; k++) printf("    "); /* 4 espaços */
  for (k = l; k <= r; k++) printf("%3d ", a[k]);
  printf("\n");
}

selection(int a[], int l, int r, int k)
{ while (r > l) {
  int i = partition(a, l, r);
  print(a, l, r, i);
  if (i >= k) r = i-1;
  if (i <= k) l = i+1;
}
}
```

Simule a execução da chamada `selection(a, 0, 10, 5)`, quando o conteúdo de `a[]` é

```
333 111 444 111 555 999 222 555 333 555 999
```

Nesta questão, você deve dizer claramente qual é a saída.

Bônus: diga o que ocorre se trocamos o trecho

```
if (i >= k) r = i-1;
if (i <= k) l = i+1;
```

em `selection()` por

```
if (i > k) r = i-1;
if (i < k) l = i+1;
```

3. [4 pontos] Descreva cuidadosamente o projeto de um programa que recebe uma cadeia de caracteres T como entrada e devolve o trecho repetido mais comprido neste texto. Por exemplo, se o texto de entrada for

```
aacabcabcabcbabac
```

então a saída de seu programa deve ser `abcbab`.

O seu projeto deve supor que a entrada T pode ser grande; por exemplo, T poderia um livro todo. Um programa implementado de acordo com o seu projeto deve ser tal que,

- (a) sob hipóteses razoáveis, ele leva tempo basicamente proporcional a $n \log n$, onde n é o número de caracteres em T ,
- (b) ele gasta uma quantidade de memória proporcional ao número de caracteres em T .

Você deve argumentar por que o programa teria estas propriedades (em particular, você deve explicitar as hipóteses que você usa em sua análise). [*Sugestão.* Um *sufixo* de uma cadeia de caracteres T é um 'segmento final' dela. Os sufixos de `abcde` são

```
abcde
bcde
cde
de
e
```

e a cadeia vazia, com 0 caracteres (6 sufixos no total). Considere os sufixos de T .]