Algoritmos e Complexidade – 8 de Novembro de 2017 – Duração: 90 min

1. Considere a seguinte variante da função `partition` usada no algoritmo de quick sort, que particiona o array em três zonas, em que na do meio todos os elementos são iguais (ao pivot).

```
void partition (int v[], int N,
                int *ep, int *eq){
  int i, p, q, t;
  // PRE: N > 0
  i=p=q=0;
  // ...
  while (i<N-1) {
      // ...
      if (v[i] < v[N-1]) {
         t = v[i]; v[i] = v[q];
         v[q] = v[p]; v[p] = t;
         p++; q++;}
      else if (v[i] == v[N-1]) {
         t = v[i]; v[i] = v[q];
         v[q] = t; q++;
      }
      i++;
  }
  t=v[i]; v[i] = v[q];
  v[q] = t; q++;
  // POS: p < q
  *ep = p; *eq = q;
}
```

   (a) Complete a anotação da função de forma a provar que no final, a zona do meio tem pelo menos um elemento (i.e. que `p < q`).

   Apresente as condições de verificação correspondentes à prova da correcção **parcial** do código entre `//PRE:` e `//POS:`, **omitindo as que dizem respeito à preservação do invariante**.

   (b) Apresente uma definição da função `qsort` que tira partido do resultado desta variante da função `partition`.

   Mostre que essa função tem um melhor caso que é linear no tamanho do array argumento. Assuma para isso que a função `partition` tem um comportamento linear no tamanho do array argumento.

2. Considere a seguinte função que *remove* de um array de inteiros os elementos consecutivamente repetidos.

```
int noRep (int v[], int N) {
  int i;
  i=0;
  while (i<N-1)
     if (v[i] == v[i+1]) {
         shift(v+i, N-i);
         N--;
     } else i++;
  return N;
}
```

```
void shift (int v[], int N) {
  int i;
  for (i=0;i<N-1;i++)
     v[i]=v[i+1];
}
```

   (a) Apresente um variante do ciclo da função `nRep` e mostre que ele decresce em cada iteração.

   (b) Identifique o melhor e pior casos da execução da função `noRep` em termos do número de escritas no array.

   Para o pior caso identificado calcule qual o número de escritas no array em função do tamanho do array argumento.

3. Considere a definição recursiva da função `minSort` ao lado.
Admitindo que num array aleatório com N elementos a probabilidade de o elemento na posição 0 ser o menor de todos é $1/N$, apresente uma recorrência que traduza o número médio de trocas (`swap`) efectuado por esta função.
Apresente uma solução dessa recorrência (não precisa de simplificar os *somatórios* envolvidos nessa solução).

```
void minSort (int v[], int N) {
  int m;
  if (N>0) {
     m = minInd (v,N);
     if (m!=0) swap (v,0,m);
     minSort (v+1,N-1);
  }
}
```