Problema 1 (5 punts)

Primera part:

```
node_llista *ant;
bool creixent = true;
int cont = 0;

ant = prim;                                   // caixa 1
act = prim→seg;

while (act≠NULL and creixent) {

  if (ant→info > act→info) creixent = false;  // caixa 2
  else {
    ant = act;
    act = act → seg;
  }
  ++ cont;
}
```

Aquesta és la solució standard. Hi ha solucions equivalents on només canvia l'ordre de les instruccions. Les solucions amb més d'un *if* (o un *if* amb més d'una condició) són en general una mica més dolentes. Existeix una versió una mica més bona on *ant* només intervé quan canvia *creixent*, però això comporta altres canvis bastant significatius per assolir l'estat que volem.

En aquest punt, *act* ha d'apuntar al primer element de la segona escala de P (o és NULL, si P només té una escala); *ant* ha d'apuntar a l'element de P anterior a l'apuntat per *act* (o a l'últim element de P, si *act* és NULL); per tant *ant* $\neq$ NULL i apunta a l'últim element de la primera escala de P; *cont* és la mida de la primera escala de P.

A l'exemple, *ant* apuntaria al primer 2 i *act* al segon 1; *cont* seria igual a 2.

Segona part:

```
if (act ≠ NULL) {

  ult = ant;                                  // caixa 3: la primera escala
  l.prim = l.ult = ant = act;                 // acaba en ant i la segona
  l.prim → ant = NULL;                        // comença en act; avancem
  act = act → seg;

  longitud = cont;
  l.longitud = 1;
  bool senar = false;
```

INVARIANT DEL SEGON BUCLE:

1) *act* apunta a un element del p.i. o és NULL; si *act* no és NULL *ant* apunta a l'element del p.i. o de *l* que era l'anterior d'*act* a P; (*)

2) entre *prim* i *ult* hi són totes les escales senars que apareixien a P abans que *act*, i el tros fins a *ant* (inclòs) de la que contenia *ant*, si aquesta és senar, totes en el seu ordre original;

3) entre *l.prim* i *l.ult* hi són totes les escales parells que apareixien a P abans que *act*, i el tros fins a *ant* (inclòs) de la que contenia *ant*, si aquesta és parell, totes en el seu ordre original;

4) *senar* indica si *ant* apareixia a una escala senar de P;

5) *longitud* és el nombre d'elements de P anteriors a *act* que pertanyien a escales senars; *l.longitud* és el nombre d'elements de P anteriors a *act* que pertanyien a escales parells

(*) als punts 2-5 de l'invariant, on apareix un punter volem referir-nos a *l'element apuntat per* aquest punter

```
    while (act ≠ NULL) {
      if (senar) {    // ant pertany a una escala senar
        if (ant→info ≤ act→info) {    // act i ant pertanyen a la mateixa escala

          ult = act;                          // caixa 4: act es queda al p.i.
          ++longitud;
        }
        else {    // act i ant NO pertanyen a la mateixa escala

          l.ult → seg = act;                  // caixa 5: act passa a l
          act → ant = l.ult;
          l.ult = act;
          ++l.longitud;
          senar = false;
        }
      }
      else {    // ant pertany a una escala parell
        if (ant→info ≤ act→info) {    // act i ant pertanyen a la mateixa escala

          l.ult = act;                        // caixa 6: act passa a l
          ++l.longitud;
        }
        else {    // act i ant NO pertanyen a la mateixa escala

          ult → seg = act;                    // caixa 7: act es queda al p.i.
          act → ant = ult;
          ult = act;
          ++longitud;
          senar = true;
        }
      }

      ant = act;                               // caixa 8: avancem el bucle
      act = act → seg;
    }
    ult→seg = l.ult→seg = NULL;
    l.act = l.prim;
  }
  act = prim;
}
```

Aquesta és la solució standard. Existeix una solució amb la meitat de comparacions de camps *info*, que seria més bona per a llistes d'objectes grans, però requereix fer servir *creixent* amb molta cura; a més, aquesta variable hauria de participar a l'invariant. Recordeu que demanem solucions que compleixin els requeriments marcats per l'esquema donat (invariant, comentaris al codi, etc) i només continguin assignacions. Això també afecta a les solucions amb *new i delete*, que a més són ineficients; encara pitjor, si fan servir *new* però no *delete*, llavors o no compleixen la postcondició o bé generen *memory leak*. Noteu, per últim, que *l.act* no apareix a cap caixa i el seu ús seria redundant en el millor dels casos.

Observacions particulars:

caixa 5: un error típic és suposar que quan s'entra en aquest *else* es pot fer servir *ant* en comptes de *l.ult*

caixa 7: anàlogament, aquí *ant* no es pot fer servir en comptes d'*ult*

caixes 5 i 6: existeix la temptació de refer els enllaços per mantenir connectat *act->seg* amb el p.i.; això simplificaria la caixa 7 però implica un risc de *segmentation fault* que s'hauria d'evitar amb *if*s; noteu que s'hauria de fer el mateix a l'inicialització del bucle.