```
2Ø FOR z = 1 TO 5 STEP 3/2
```

z prehaja skozi vrednosti 1, 2.5 in 4. Opazili boste, da ni nujno uporabljati cela števila ter da ni nujno, da kontrolna variabla doseže mjeno vrednost (zanka bo delovala, dokler kontrolna variabla ne bo manjša ali enaka mejni vrednosti).

Poizkusite sedaj ta program narediti tako, da bodo števila od 1 do 1Ø v obrnjenem zaporedju:

```
1Ø FOR n = 1Ø TO 1 STEP -1
2Ø PRINT  n
3Ø NEXT n
```

Rekli smo, da se ta program izvršuje, dokler je kontrolna variabla manjša ali enaka mejni vrednosti. Če se zamislite, kaj bi to pomenilo v tem primeru, boste opazili, da je to nesmiselno. Prejšnje pravilo je potrebno modificirati: če je korak negativen, se zanka izvršuje, dokler ni kontrolna variabla večja ali enaka mejni vrednosti. Posebno pazljivi morate biti, kadar delate z dvema FOR-NEXT zankama, ki se nahajata ena v drugi. Poizkusite napr viti program, ki izpisuje števila kompletnega seta šestih domin:

```
1Ø FOR m = Ø TO 6
2Ø FOR n = Ø TO m
3Ø PRINT m; ": "; n; "";
4Ø NEXT n
5Ø PRINT
6Ø NEXT m
```

(2Ø–3Ø: n-zanka; 1Ø–6Ø: m-zanka)

Kot vidite, je n-zanka popolnoma objeta z m-zanko. Tisto, kar moramo preprečiti, sta dve FOR-NEXT zanki, ki se pokrivata, vendar nista popolnoma ena v drugi:

```
5 REM ta program je napačen
1Ø FOR m = Ø TO 6
2Ø FOR n = Ø TO m
3Ø PRINT m; ": "";
```

(1Ø–3Ø: m-zanka)