Checklista för defensiv programmering

- Skyddar sig funktionen mot **dåliga indata**?
- Används **assertions** för att **dokumentera** omständigheter som aldrig borde uppstå, inklusive **pre- och postvillkor**?
- Används **assertions enbart** för att dokumentera omständigheter som **aldrig borde uppstå?**
- Används tekniker för att **minska skadan från fel** och för att **minska mängden kod** som måste "**bry sig om**" felhantering?
- Används **informationsgömningsprincipen** för att kapsla in interna förändringar?
- Har **hjälpfunktioner** implementerats i **utvecklingskoden** som hjälper till vid felsökning och debuggning?
- Är **mängden** defensiv programmering adekvat – varken för mycket eller för lite?
- Används **offensiva programmeringstekniker** för att minska risken att fel inte uppmärksammas under utveckling?

Adapterad från Steve McDonnell's utmärkta "Code Complete"

UU