

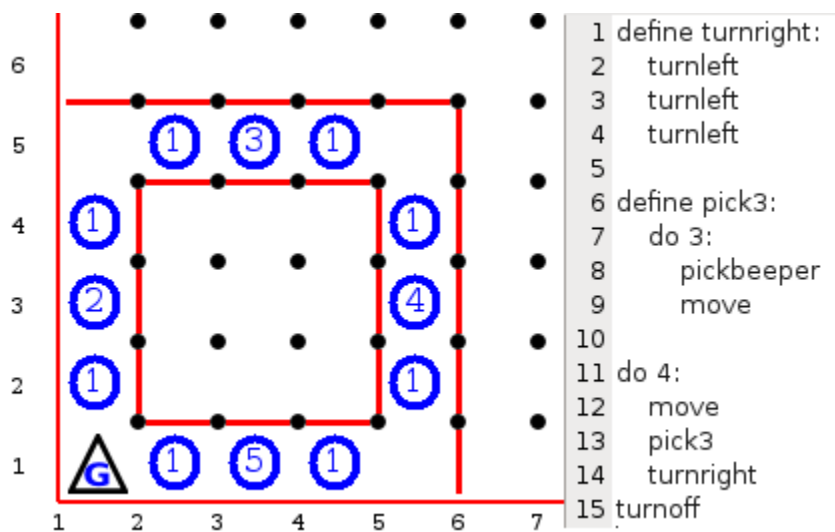
Guido van Robot och olika UNIX-kommandon

Till denna session är det tänkt att ni ska ha arbetat en del med både *UNIX*-kommandon och börjat med *Guido van Robot*. Vi ska nu studera dessa båda saker med hjälp av *Peer Instruction*, en alternativ undervisningsform där vi arbetar med nyckelfrågeställningar som illustrerar viktiga begrepp och testar att man förstår dessa begrepp.

Begreppet *föreläsning* är egentligen ett förlegat arv från medeltiden då man inte kunde trycka böcker, då fanns inte så många böcker och av ekonomiska skäl var man tvungen att ha en person som läste för de andra, det "lästes för", därav namnet *föreläsning*. På engelska heter det *lecture* som bara betyder läsning. Idag har vi alla varsinn bok och vi ska därför inse detta och börja arbeta med en annan undervisningsform som utnyttjar att ni har eget studiematerial. Vi förutsätter idag alltså att ni har arbetat med materialet innan och ägnar denna dag åt frågor i en speciell form som kallas *Conceptests* – begreppstester.

Vi börjar med *Guido van Robot*.

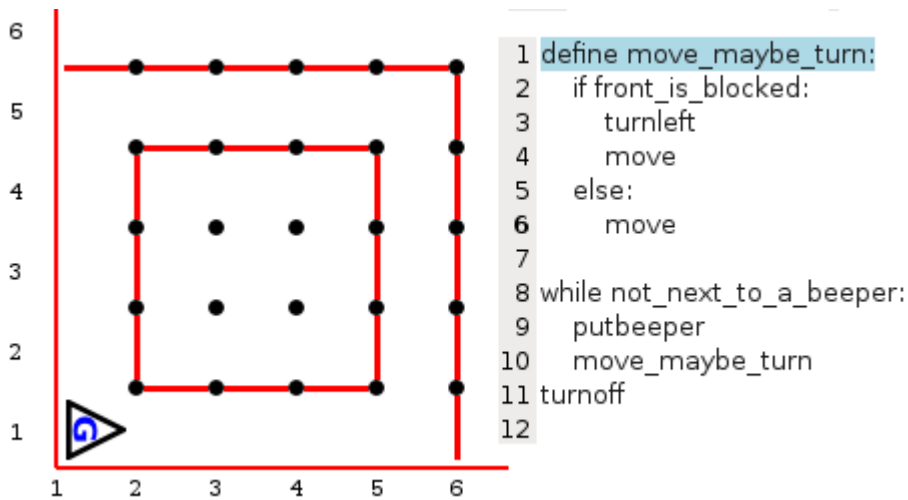
Begreppstest 1. Givet följande utgångsläge:



Vad blir resultatet då *Guido* har kört alla sina instruktioner?

- A. *Guido* står tittandes österut, med alla pipare och inga pipare är kvar.
- B. *Guido* står i tittandes västerut, med alla pipare och inga pipare är kvar.
- C. *Guido* står i samma ställning som nu (tittandes norrut), med 12 pipare och kvar är 10 pipare.
- D. *Guido* står i samma ställning som nu (tittandes norrut), med alla pipare och inga pipare är kvar.

Begreppstest 2. Givet följande utgångsläge: (*Guido* har 100 pipare från början.)



Koden till *Guido* ser till att *Guido* går runt i hela den kvadratiska strukturen och placerar pipare ända tills han stöter på en pipare som han tidigare lagt ned, hela gången fylls alltså med 16 pipare. Vi intresserar oss nu för olika varianter av ovanstående program. Frågan är vilka av följande tre varianter som fungerar på precis samma sätt som programmet ovan:

A	B	C
<pre> define move_maybe_turn: if front_is_blocked: turnleft move while not_next_to_a_beeper: putbeeper move_maybe_turn turnoff </pre>	<pre> define move_maybe_turn: if front_is_blocked: turnleft move while not_next_to_a_beeper: putbeeper move_maybe_turn turnoff </pre>	<pre> define maybe_turn: if front_is_blocked: turnleft while not_next_to_a_beeper: putbeeper move_maybe_turn move turnoff </pre>

- A. Alla alternativ fungerar precis som det givna programmet.
- B. Alternativen A & B fungerar som det givna programmet, men inte C.
- C. Alternativen A & C fungerar som det givna programmet, men inte B.
- D. Alternativen B & C fungerar som det givna programmet, men inte A.

Vi tar nu ett par begreppstester på *UNIX*-kommandon.

Begreppstest 4. Vi antar att vi gör följande kommandon vid en *UNIX*-prompt:

```
$ ls -l
total 12
-rw-r--r-- 1 johnny johnny 6 May 16 11:25 fil1
-rw-r--r-- 1 johnny johnny 7 May 16 11:25 fil2
-rw-r--r-- 1 johnny johnny 8 May 16 11:25 fil3
$ mkdir dir1 dir2
$ mv fil1 dir1
$ mv -t dir2 fil2 fil3
$ mv dir1 fil1
$ mv dir2 fil2
$ ls -l
```

När vi nu gör `ls -l`, efter alla kommandon, vad ser vi då för resultat?

Här är ett utdrag ur manualsidan för `mv`:

NAME

`mv` - move (rename) files

SYNOPSIS

```
mv [OPTION]... [-T] SOURCE DEST
mv [OPTION]... SOURCE... DIRECTORY
mv [OPTION]... -t DIRECTORY SOURCE...
```

DESCRIPTION

Rename `SOURCE` to `DEST`, or move `SOURCE(s)` to `DIRECTORY`.

Mandatory arguments to long options are mandatory for short options too.

`--backup[=CONTROL]`

make a backup of each existing destination file

`-t, --target-directory=DIRECTORY`

move all `SOURCE` arguments into `DIRECTORY`

A. Vi ser två filer.

B. Vi ser två kataloger.

C. Vi ser återigen de filer som fanns i arbetskatalogen från början (`fil1`, `fil2` och `fil3`).

D. Ingetdera av alternativen A-C är korrekta.

Begreppstest 5. Vi antar att vi kör virtuellt med katalogen `/media/sf_me` som delad katalog. Vi antar att vi gör följande kommandon vid en *UNIX*-prompt:

```
$ pwd
/home/me/test
$ ls -l
total 12
-rw-r--r-- 1 me me 6 May 16 11:25 fil1
-rw-r--r-- 1 me me 7 May 16 11:25 fil2
-rw-r--r-- 1 me me 8 May 16 11:25 fil3
$ mv fil1 /media/sf_me
$ cp fil2 /media/sf_me
$ cp fil3 /media/sf_me
$ rm fil3
```

Här är då alltså en katalog i värdsystemet monterad på `/media/sf_me` via *VirtualBox Guest Additions*.

Vilket påstående är sant?

A. Alla filer, `fil1`, `fil2` och `fil3` finns i separata kopior både i värdsystemet och den virtuella maskinen, om vi till exempel redigerar i `fil1` på värdsystemet så avspeglas förändringarna i den kopia som finns i den virtuella maskinen i `/home/me/test`.

B. Filen `fil2` finns i två separata kopior eftersom vi gjorde en `cp` på den, om vi ändrar i `/home/me/test/fil2` så avspeglas *inte* dessa förändringar i `/media/sf_me/fil2` eftersom det är två separata filer.

C. De två sista kommandona som utfördes,

```
$ cp fil3 /media/sf_me
$ rm fil3
```

har inte samma effekt på `fil3` som kommandot `mv fil1 /media/sf_me` har på `fil1`, filen `fil3` får nämligen ett nytt namn vid kopiering från `/home/me/test` till `/media/sf_me`, om man använder `mv` byts inte namnet på filen, den *bara* flyttas.

D. Ingetdera av alternativen A-C är sanna.