SI-möte #11, Programkonstruktion och datastrukturer
Lösningsförslag

Elias Castegren & Kristiina Ausmees
elca7381@student.uu.se ‖ krau6498@student.uu.se

Övningar

1.

```
val a = 42;
val b = ref 42;
val c = a;
val d = b;
val f1 = (fn x => x + a);
val f2 = (fn x => x + !b);
f1 2;   → 44
f2 2;   → 44
val a = 13;
b := 99;
d := 23;   (OBS! d=b, alltså ändras b också)
f1 c;   → 84
f2(!d);   → 46
```

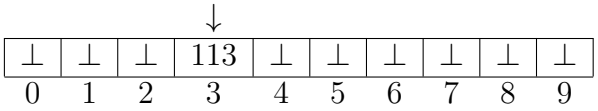
$a = 13, \quad b = ref\ 23, \quad c = 42, \quad d = b = ref\ 23$

2.

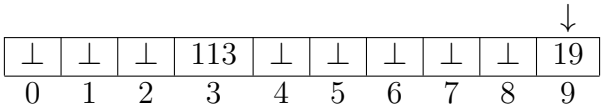
Hashvärdena för våra element:

k	$h(k) = k \ mod \ 10$
113	3
19	9
155	5
16	6
129	9
43	3
99	9
26	6
125	5
73	3

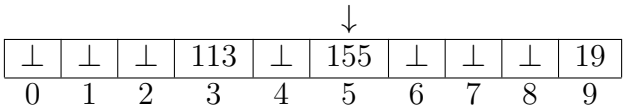
Sätt in 113



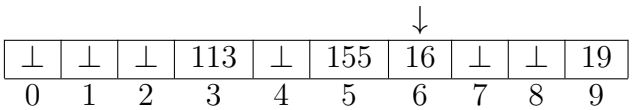
Sätt in 19



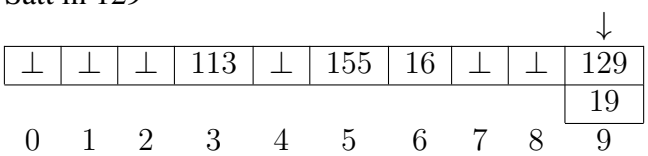
Sätt in 155



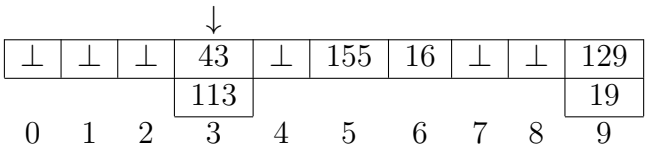
Sätt in 16



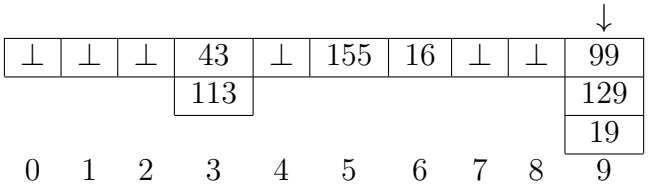
Sätt in 129



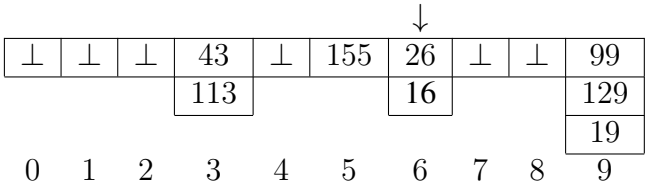
Sätt in 43



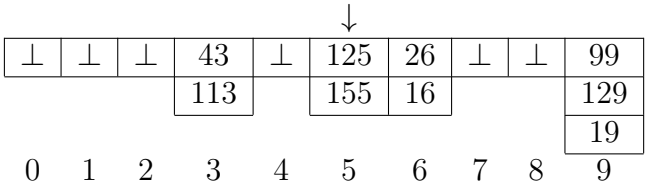
Sätt in 99



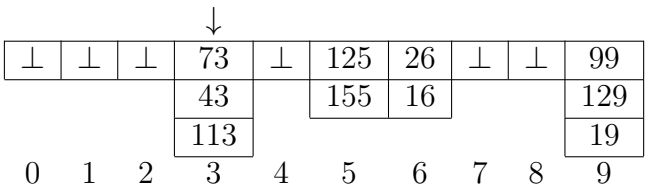
Sätt in 26



Sätt in 125



Sätt in 73



Borttagning av element går till på samma sätt som ovan. Man kan sluta leta efter element som inte finns när man når slutet av kedjan på motsvarande plats.

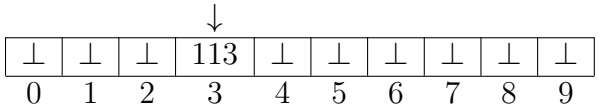
3.

Linjär probing

$h(k,i) = (k + f(i))\ mod\ 10 \qquad f(i) = i$

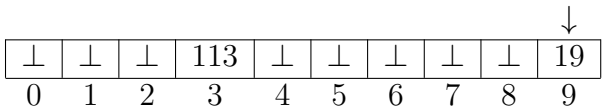
Sätt in 113

$i = 0$



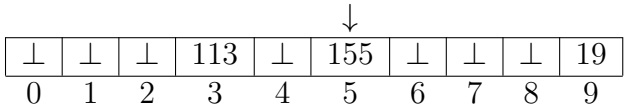
Sätt in 19

$i = 0$



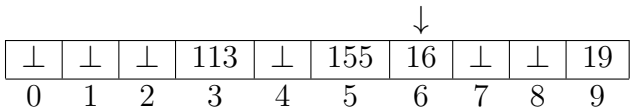
Sätt in 155

$i = 0$



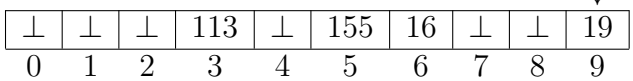
Sätt in 16

$i = 0$



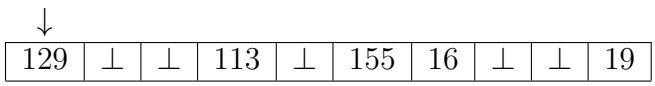
Sätt in 129

$i = 0$



	0	1	2	3	4	5	6	7	8	9
	$\perp$	$\perp$	$\perp$	113	$\perp$	155	16	$\perp$	$\perp$	19

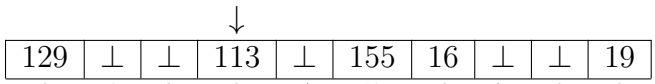
$i = 1$



	0	1	2	3	4	5	6	7	8	9
	129	$\perp$	$\perp$	113	$\perp$	155	16	$\perp$	$\perp$	19

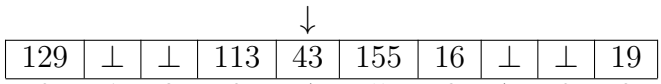
Sätt in 43

$i = 0$



	0	1	2	3	4	5	6	7	8	9
	129	$\perp$	$\perp$	113	$\perp$	155	16	$\perp$	$\perp$	19

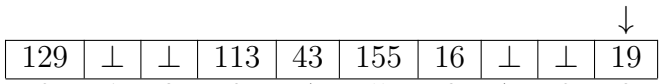
$i = 1$



	0	1	2	3	4	5	6	7	8	9
	129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19

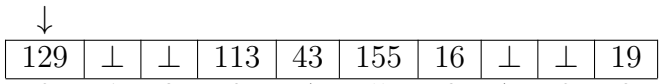
Sätt in 99

$i = 0$



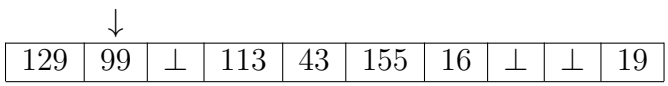
	0	1	2	3	4	5	6	7	8	9
	129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19

$i = 1$



	0	1	2	3	4	5	6	7	8	9
	129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19

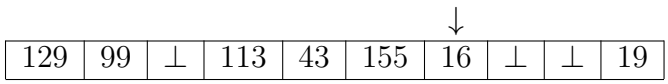
$i = 2$



	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	$\perp$	$\perp$	19

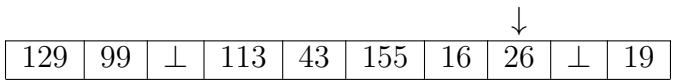
Sätt in 26

$i = 0$



	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	$\perp$	$\perp$	19

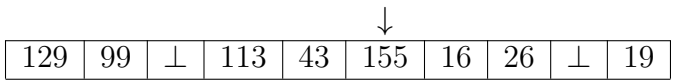
$i = 1$



	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	26	$\perp$	19

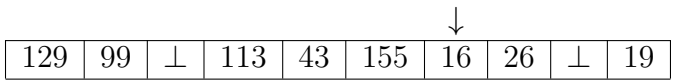
Sätt in 125

$i = 0$



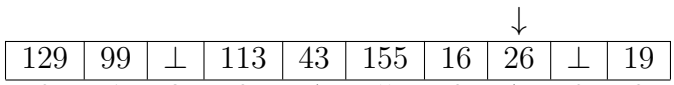
	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	26	$\perp$	19

$i = 1$



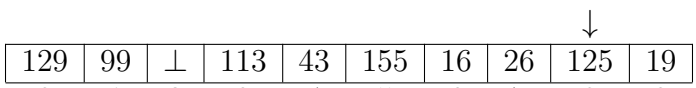
	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	26	$\perp$	19

$i = 2$



	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	26	$\perp$	19

$i = 3$



	0	1	2	3	4	5	6	7	8	9
	129	99	$\perp$	113	43	155	16	26	125	19

Sätt in 73

$i = 0$

			↓						
129	99	$\perp$	113	43	155	16	26	125	19
0	1	2	3	4	5	6	7	8	9

$i = 1$

				↓					
129	99	$\perp$	113	43	155	16	26	125	19
0	1	2	3	4	5	6	7	8	9

...

...

...

$i = 9$

		↓							
129	99	73	113	43	155	16	26	125	19
0	1	2	3	4	5	6	7	8	9

Man hittar rätt plats på samma sätt när man tar bort element. Ett borttaget element ersätts med Δ för att skilja det från en plats som aldrig har haft något element. Efter att ha tagit bort elementen $19, 26, 129$ och 73 ser hashtabellen ut så här:

Δ	99	Δ	113	43	155	16	Δ	125	Δ
0	1	2	3	4	5	6	7	8	9

Letar man efter ett element som inte finns måste man proba tills man hittar ett $\perp$, eller tills man har provat alla möjliga probningar (ovan, när $i = 9$)

Kvadratisk probing

$h(k, i) = (k + f(i)) \; mod \; 10 \qquad f(i) = i^2$

De sex första insättningarna blir identiska med de ovan (eftersom $0 = 0^2$ och $1 = 1^2$).

Sätt in 99

$i = 0$

								↓	
129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19
0	1	2	3	4	5	6	7	8	9

$i = 1$

↓									
129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19
0	1	2	3	4	5	6	7	8	9

$i = 2$

			↓						
129	$\perp$	$\perp$	113	43	155	16	$\perp$	$\perp$	19
0	1	2	3	4	5	6	7	8	9

$i = 3$

								↓	
129	$\perp$	$\perp$	113	43	155	16	$\perp$	99	19
0	1	2	3	4	5	6	7	8	9

Sätt in 26

$i = 0$

						↓			
129	⊥	⊥	113	43	155	16	⊥	99	19
0	1	2	3	4	5	6	7	8	9

$i = 1$

							↓		
129	⊥	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

Sätt in 125

$i = 0$

					↓				
129	⊥	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 1$

						↓			
129	⊥	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 2$

									↓
129	⊥	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 3$

				↓					
129	⊥	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 4$

	↓								
129	125	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

Sätt in 73

$i = 0$

			↓						
129	125	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 1$

				↓					
129	125	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 2$

							↓		
129	125	⊥	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

$i = 3$

		↓							
129	125	73	113	43	155	16	26	99	19
0	1	2	3	4	5	6	7	8	9

Samma principer gäller för borttagning som ovan. Man kan sluta leta efter ett element när man hittar ett $\perp$ eller när man har provat alla möjliga probningar (ovan när $i = 5$).

4.
Vi skriver en funktion som går igenom en lista med en hashtabell h och för varje element e kollar om e finns i h. Om e inte finns i h behåller vi elementet i listan och lägger in det i h. Om e finns i h så betyder det att e har förekommit tidigare i listan, därför tar vi bort det ur listan.

Eftersom vi bara är intresserade av att att snabbt lagra och hitta nycklar i tabellen så låter vi alla värden vi lägger in i hashtabellen vara `unit`.

```
(* killDuplicates l
   TYPE: string list -> string list
   PRE: ()
   POST: l med endast en förekomst av varje element
   EXAMPLES: killDuplicates ["a", "b", "a", "b", "c"] = ["a", "b", "c"]
   COMPLEXITY: Theta(|l|) i genomsnittliga fall
*)
fun killDuplicates(l) =
    let
        (* killDuplicates'(l,h)
           TYPE: string list*HashArray.hash -> string list
           PRE: ()
           POST: l med endast en förekomst av varje element och utan något
                 element som finns som nyckel i h
           EXAMPLES: killDuplicates'(["a", "b", "b", "c"], {("a"->())}) =
                          ["a", "b", "c"]
           COMPLEXITY: Theta(|l|) för en välbyggd hashtabell h
        *)
        fun killDuplicates'([], _) = []
          | killDuplicates'(f::r, h) =
            if isSome(HashArray.sub(h,f)) then
                killDuplicates'(r, h)
            else
                (HashArray.update(h,f,());
                 f::killDuplicates'(r, h))
        val h = HashArray.hash(length l)
    in
        killDuplicates'(l,h)
    end;
```

`killDuplicates` skulle kunna ha typen `''a list -> ''a list` om Poly/ML tillät polymorfa hashtabeller (som många andra programmeringsspråk och ML-dialekter gör).