7th practice

1. Find the longest word in a given text file that contains character 'w'. Words are separated by spaces from each other.

Specification:

S = (x:infile($\mathbb{K}$), l:$\mathbb{L}$, longest:$\mathbb{S}$)
Pre = ($x = x_0$)

Idea:

Enumerate the words along with their length and a boolean value indicating whether the word contains 'w'.

New specification:

S = (t:enor(WORD), l:$\mathbb{L}$, longest:$\mathbb{S}$)
WORD = rec(word:$\mathbb{S}$, w:$\mathbb{L}$)
Pre = ($t = t_0$)

Conditional maximum search

f(e)	~	|e.word |
cond(e)	~	e.w
H, >	~	$\mathbb{N}$, >

Post = ((l, max, elem) = $\underset{e.w}{\mathbf{MAX}_{e\in t_0}}$ |e.word | ∧ l → longest =elem.word)

Algorithm:

```
l := false; t.first()
¬t.end()
    ¬ t.current().w:
        —
    t.current().w ∧ l:
        |t.current().word | > max
            max, longest := |t.current().word |, t.current().word
        else
            —
    t.current().w ∧ ¬l:
        l, max, longest := true, |t.current().word |, t.current().word
    t.next()
```

Enumerator:

t:enor(WORD)   WORD = rec(word:$\mathbb{S}$, w:$\mathbb{L}$)

WORD *	first()	next()	current() : WORD	end() : $\mathbb{L}$

x : infile($\mathbb{K}$)
dx : $\mathbb{K}$
sx : Status
curr : WORD
end : $\mathbb{L}$
	sx,dx,x:read
	next()
		see below	**return** curr	**return**
				end

next() method

S = (x:infile($\mathbb{K}$), dx:$\mathbb{K}$, sx:Status, curr:WORD, end:$\mathbb{L}$)
Pre = (x = x’ ∧ dx = dx’ ∧ sx = sx’)
Post = ((dx”,(sx”,dx”,x”))=$\mathrm{SELECT}_{dx\in(dx',x')}$ (sx=abnorm ∨ dx≠’ ’)
∧ end =(sx”=abnorm)
∧ (¬end → (curr.word, (sx,dx,x)) = $\overset{dx\neq' '}{\bigoplus}_{dx\in(dx'',x'')}$<dx> ∧ (curr.w, (sx,dx,x)) = $\overset{dx\neq' '}{\bigvee}_{dx\in(dx'',x'')}$dx=’w’))

Remark: the subproblem (decide whether a word contains 'w'), instead of linear search will be determined by OR’ing the boolean values. The reason for this is that the whole current word has to be processed and OR’ing can be merged into the same loop with the concatenation of the word.

Selection

t:enor(E) ~ x:infile(𝕂) (sx,dx,x:read)
without first()
cond(e) ~ sx=abnorm ∨ dx≠’ ’

Two summations (concatenation and OR’ing)

t:enor(E) ~ x:infile(𝕂) (sx,dx,x:read)
without first(), cond: dx≠’ ’
f(e) ~ (<dx>, dx=’w’)
s ~ (curr.word, curr.w)
H, +, 0 ~ (𝕂*, 𝕃), (⊕, ∨), (<>, false)

```
sx=norm ∧ dx=’ ’
    sx,dx,x:read
end := sx=abnorm
¬end
    curr.word, curr.w := <>,                          | –
    sx=norm ∧ dx≠’ ’                                  |
        curr.word, curr.w :=                          |
        curr.word ⊕ <dx>,                             |
        curr.w ∨ (dx=’w’)                             |
        sx,dx,x:read                                  |
```

2. Given a file containing data of huntings. Each line of the file consists of the name of the hunter, the date of the hunting, the species and weight of the animal shot by the given hunter at the given hunting. The file is sorted by hunter and then by date.
   Decide, whether every hunter has shot a bear at any of his/her hunting.

Specification:

S = (x:infile(Trophy), l:$\mathbb{L}$)

Trophy = rec(name:$\mathbb{S}$, date:$\mathbb{S}$,
species:$\mathbb{S}$, weight:$\mathbb{N}$)

Pre = ($x=x_0 \wedge x\nearrow_{(name,date)}$)

New specification:

S = (t:enor($\mathbb{L}$), l:$\mathbb{L}$)

Pre = ($t=t_0$)

Post = ($l = \forall\text{SEARCH}_{e\in t_0}\, e$)

Optimistic linear search

cond(e) ~ e

Idea:

Enumerate as many boolean values as the number of the hunters, a boolean value is true in case the related hunter has shot a bear.

Algorithm:

```
l := true
t.first()
$l \wedge \neg$t.end()
    l := t.current()
    t.next()
```

Enumerator:

t:enor($\mathbb{L}$)

$\mathbb{L}$*	first()	next()	current() : $\mathbb{L}$	end() : $\mathbb{L}$
x : infile(Trophy) dx : Trophy sx : Status curr : $\mathbb{L}$ end : $\mathbb{L}$	sx,dx,x:read next()	see below	**return** curr	**return** end

Trophy = rec(name:$\mathbb{S}$, date:$\mathbb{S}$, species:$\mathbb{S}$, weight:$\mathbb{N}$)

next() method

S = (x:infile(Trophy), dx: Trophy, sx:Status, curr:$\mathbb{L}$, end:$\mathbb{L}$)

Pre = ($x = x' \wedge x\nearrow_{(name,date)} \wedge dx = dx' \wedge sx = sx'$) dx.name = dx'.name

Post = (end = (sx'=abnorm) $\wedge$ ($\neg$end $\rightarrow$ (curr, (sx,dx,x)) = $\bigvee_{dx\in(dx',x')}$ curr.species="bear"))

Summation (OR'ing)

t:enor(E) ~ x:infile(Trophy) (sx,dx,x:read)
without first(),
cond: dx.name=dx'.name

f(e) ~ dx.species="bear"

s ~ curr

H, +, 0 ~ $\mathbb{L}$, $\vee$, false

```
end := sx=abnorm
$\neg$end
    curr := false
    n := dx.name
    sx=norm $\wedge$ dx.name=n
        curr := curr $\vee$ (dx.species="bear")
        sx,dx,x:read
    –
```

3. Given a file containing data of huntings. Each line of the file consists of the name of the hunter, the date of the hunting, the species and weight of the animal shot by the given hunter at the given hunting. The file is sorted by hunter and then by date.
Count the hunters who shot a rabbit at each of his/her huntings and the total weight of animals shot by the hunter is greater than 250 kg.

Specification:

S = (x:infile(Trophy), c: $\mathbb{N}$)

Trophy = rec(name:$\mathbb{S}$, date:$\mathbb{S}$,
species:$\mathbb{S}$,weight:$\mathbb{N}$)

Pre = ($x=x_0 \wedge x\nearrow_{(name,date)}$)

New specification:

S = (t:enor(Hunter), c: $\mathbb{N}$)

Hunter = rec(name: $\mathbb{S}$, r:$\mathbb{L}$, totalw:$\mathbb{N}$)

Pre = ($t=t_0$)

Post = ($c = \sum_{\substack{e \in t_0 \\ e.r \wedge e.totalw>250}} 1$)

Counting

cond(e) ~ e.r ∧ e.totalw>250

Idea:

Enumerate the hunters (name, boolean value indicating whether the hunter shot a rabbit at each of his/her huntings, total weight of animals shot by the hunter).

Algorithm:

```
c := 0
t.first()
¬t.end()
    t.current().r ∧ t.current().totalw>250
        c := c + 1
    else
        –
    t.next()
```

Enumerator:

Idea: In order to create the enumerator t:enor(Hunter) mentioned above, let us suppose that we have an enumerator y:enor(Aggregation) enumerating the total achievement of hunters per huntings. y:enor(Aggregation) is sorted by the name of the hunter and then by the date of the hunting, and also includes whether the hunter has shot any rabbit at the given hunting and the total weight of the hunted animals at the given hunting.

t:enor(Hunter) Hunter = rec(name: $\mathbb{S}$, r:$\mathbb{L}$, totalw:$\mathbb{N}$)

Hunter*	first()	next()	current() : Hunter	end() : $\mathbb{L}$

y : enor(Aggregation)
curr : Hunter
end : $\mathbb{L}$
	y.first()
	next()
		see below	**return** curr	**return** end

Aggregation = rec(name: $\mathbb{S}$, date:$\mathbb{S}$, shotr:$\mathbb{L}$, totalw:$\mathbb{N}$)

next() method of t:enor(Hunter)

S = (y:enor(Aggregation), curr:Hunter, end:$\mathbb{L}$)

Pre = ($y = y' \wedge y\nearrow_{(name,date)}$)

Post = (end = y'.end() ∧ (¬end → curr.name = y'.current().name

$$\wedge\ (curr.r, y) = \bigwedge_{e\in(y'.current(),y')}^{e.name = curr.name} e.shotr$$

$$\wedge\ (curr.totalw, y) = \sum_{e\in(y'.current(),y')}^{e.name\ =\ curr.name} e.totalw\ \ )\ \ )$$

Two summations (AND'ing and addition) on the same enumerator

t:enor(E)	~	y:enor(Aggregation) without first() cond: (y.current().name = curr.name)
f(e)	~	y.current().shotr
s	~	curr.r
H, +, 0	~	𝕃, ∧, true
f(e)	~	y.current().totalw
s	~	curr.totalw
H, +, 0	~	ℕ, +, 0

```
end := y.end()
¬end
    curr := (y.current().name, true, 0)
    ¬y.end() ∧  y.current().name=curr.name                –
        curr.r := curr.r ∧ y.current().shotr
        curr.totalw :=
            curr.totalw + y.current().totalw
        y.next()
```

Enumerator:

y:enor(Aggregation)   Aggregation = rec(name: 𝕊, date:𝕊, shotr:𝕃, totalw:ℕ)

Aggregation*	first()	next()	current() : Aggregation	end() : 𝕃
x : infile(Trophy) dx : Trophy sx : Status curr : Aggregation end : 𝕃	sx,dx,x:read next()	see below	**return** curr	**return** end

Trophy = rec(name:𝕊, date:𝕊, species:𝕊, weight:ℕ)

*next() method of y:enor(*Aggregation*)*

S = (x:infile(Trophy), dx:Trophy, sx:Status, curr: Aggregation, end:𝕃)

Pre = (x = x' ∧ x↗$_{(name,date)}$ ∧ dx = dx' ∧ sx = sx')

Post = (end = (sx'=abnorm) ∧ (¬end → curr.name=dx'.name ∧ curr.date=dx'.date ∧

$$\wedge\ (curr.shotr, (sx,dx,x)) = \bigvee_{dx\in(dx',x')}^{dx.name=curr.name\ \wedge\ dx.date=curr.date} dx.species=\text{"rabbit"}\ \wedge$$

$$\wedge\ (curr.totalw, (sx,dx,x)) = \sum_{dx\in(dx',x')}^{dx.name=curr.name\ \wedge\ dx.date=curr.date} dx.weight\ \ )\ \ )$$

Two summations (OR'ing and addition) with a common enumerator processing the same file

t:enor(E)	~	x:infile(Trophy) sx,dx,x:read, without first(), cond: dx.name=curr.name ∧ dx.date=curr.date
f(e)	~	dx.species="rabbit"
s	~	curr.shotr
H, +, 0	~	𝕃, ∨, false
f(e)	~	dx.weight
s	~	curr.totalw
H, +, 0	~	ℕ, +, 0

```
end := sx=abnorm
¬end
    curr :=  (dx.name, dx.date, false, 0)
    sx=norm ∧ dx.name=curr.name                           –
        ∧ dx.date=curr.date
        curr.shotr :=
            curr.shotr ∨ (dx.species="rabbit")
        curr.totalw :=  curr.totalw + dx.weight
        sx,dx,x:read
```