6th practice

1. Integers are stored in a sequential input file sorted in ascending order. Count how many times each number occurs in the file and write the (number, count) records into a sequential output file.

Specification:

S = (x:infile($\mathbb{Z}$), y:outfile(Stat))

Stat = rec(num:$\mathbb{Z}$, count:$\mathbb{N}$)

Pre = ($x = x_0 \wedge x\nearrow$)

($x\nearrow$ denotes that x is sorted in an ascending order)

New Specification:

S = (t:enor(Stat), y:outfile(Stat))

Pre = ($t = t_0$)

Post = ($y = t_0$) = ($y = \bigoplus_{e \in t_0} \langle e \rangle$)

Summation (copying)

f(e) ~ <e>

s ~ y

H, +, 0 ~ Stat*, $\oplus$, <>

Idea:

Enumerate the (number, count) records and copy them into the output file.

Algorithm:

y := <>
t.first()
$\neg$t.end()
y : write(t.current())
t.next()

Enumerator:

t:enor(Stat) Stat= rec(num:$\mathbb{Z}$, count:$\mathbb{N}$)

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

next() method

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

Pre = ($x = x' \wedge x\nearrow \wedge dx = dx' \wedge sx = sx'$)

Post = ($end = (sx'=abnorm) \wedge (\neg end \rightarrow curr.num=dx' \wedge (curr.count, (sx,dx,x)) = \sum_{dx \in (dx',x')}^{dx = curr.num} 1))$

Note: Summation has two results: the count (curr.count); and the current state of the enumerator, which is the value of the variables sx,dx,x after the next() method has finished.

Summation (counting)

t:enor(E) ~ x:infile($\mathbb{Z}$) (sx,dx,x:read)

without first() , cond: dx=curr.num

f(e) ~ 1

s ~ curr.count

H, +, 0 ~ $\mathbb{N}$, +, 0

end := sx=abnorm	
$\neg$end	
curr.num, curr.count := dx, 0	
sx=norm $\wedge$ dx=curr.num	–
curr.count := curr.count + 1	
sx,dx,x:read	

2. We store the transactions of the customers of a bank in a sequential input file. A transaction contains the account number of the customer, the date of the transaction, and the amount of the transaction (integer, positive: deposit, negative: withdraw). The transactions are sorted by the account number of the customer. Write the account numbers and balances of customers who have less than -1000 Euros into a sequential output file.

Specification:

S = (x:infile(Customer), y:outfile(Balance))

Customer = rec(acc:$\mathbb{S}$, date:$\mathbb{S}$, am:$\mathbb{Z}$)

Balance = rec(acc:$\mathbb{S}$, bal:$\mathbb{Z}$)

Pre = ($x = x_0 \wedge x\nearrow_{acc}$)

($x\nearrow_{acc}$ denotes that x is sorted by the account number)

New specification:

S = (t:enor(Balance), y:outfile(Balance))

Pre = ($t = t_0$)

Post = ($y = \bigoplus_{e \in t_0} \langle e \rangle$)

e.bal<-1000

Summation (assortment)

f(e) ~ <e> if e.bal < -1000

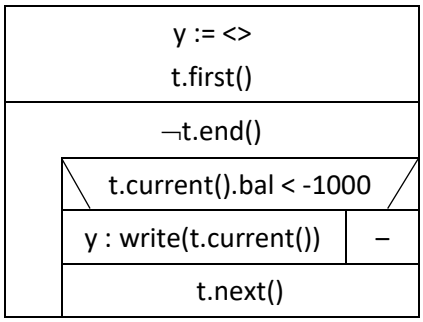
s ~ y

H, +, 0 ~ Balance*, $\oplus$, <>

Idea:

Enumerate the balances of customers by account number and pick the ones with balance less than -1000.

Algorithm:



Enumerator:

t:enor(Balance) Balance = rec(acc:$\mathbb{S}$, bal:$\mathbb{Z}$)

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

Customer = rec(acc:$\mathbb{S}$, date:$\mathbb{S}$, am:$\mathbb{Z}$)

next() method

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

Pre = ($x = x' \wedge x\nearrow_{acc} \wedge dx = dx' \wedge sx = sx'$)

Post = $(end=(sx'=abnorm) \wedge (\neg end \rightarrow curr.acc=dx'.acc \wedge (curr.bal, (sx,dx,x)) = \sum_{dx \in (dx',x')}^{dx.acc = curr.acc} dx.am))$

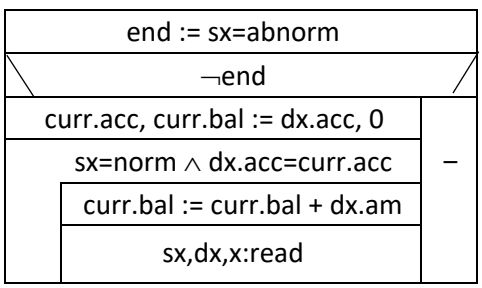
Summation

t:enor(E) ~ x:infile(Customer)(sx,dx,x:read)
without first(),
cond: dx.acc=curr.acc

f(e) ~ dx.am

s ~ curr.bal

H, +, 0 ~ $\mathbb{Z}$, + , 0



3. Count the number of words in a text file. The words that are longer than 12 characters should be counted twice. A word is delimited by spaces or the end of the file.

Specification:

S = (x:infile($\mathbb{K}$), c:$\mathbb{N}$)
Pre = (x = x_0)

New specification:

S = (t:enor($\mathbb{N}$), c:$\mathbb{N}$)
Pre = (t = t_0)
Post = (c = $\sum_{e \in t_0} \{2, \text{if } e>12; 1, \text{if } e \leq 12\}$)

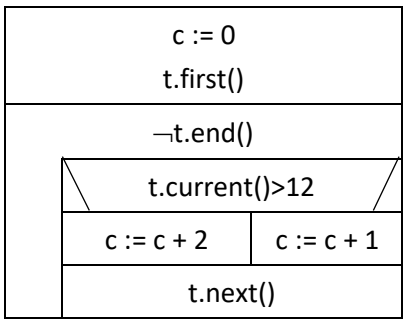
Summation

f(e) ~ {2, if e>12; 1, if e≤12}
s ~ c
H, +, 0 ~ $\mathbb{N}$, +, 0

Idea:

Enumerate the length of the words.

Algorithm:



Enumerator:

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

$\mathbb{N}$*	first()	next()	current() : $\mathbb{N}$	end() : $\mathbb{L}$
x : infile($\mathbb{K}$) dx : $\mathbb{K}$ sx : Status curr : $\mathbb{N}$ 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:$\mathbb{N}$, end:$\mathbb{L}$)
Pre = (x = x' ∧ dx = dx' ∧ sx = sx')
Post = ((dx",(sx",dx",x"))=SELECT$_{dx \in (dx',x')}$ (sx=abnorm ∨ dx≠' ') ∧ $_{dx \neq ' '}$
end=(sx"=abnorm) ∧ (¬end → (curr, (sx,dx,x))=$\sum_{dx \in (dx'',x'')} 1$))

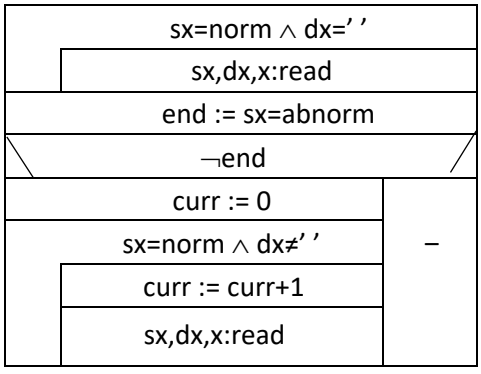
Note: The selection has two results: the space we have been looking for (dx'') and the state of the enumerator (sx", dx", x"). We could denote the result of the selection as just (sx",dx",x") without loss of understanding.

Selection

t:enor(E) ~ x:infile($\mathbb{K}$) (sx,dx,x:read) without first()
cond(e) ~ sx=abnorm ∨ dx≠' '

Summation (counting)

t:enor(E) ~ x:infile($\mathbb{K}$) (sx,dx,x:read) without first(), cond: dx≠' '
f(e) ~ 1
s ~ curr
H, +, 0 ~ $\mathbb{N}$, +, 0



4. Copy the text from a sequential input file into a sequential output file and keep only one space between the words.

Specification:

S = (x:infile($\mathbb{K}$), y:outfile($\mathbb{K}$))

Pre = (x = x_0)

Idea:

Enumerate the words and write them to the output file by putting a single space between them.

New specification:

S = (t:enor($\mathbb{S}$), y:outfile($\mathbb{K}$))

Pre = (t = t_0)

Post = (y = $\oplus_{e \in t_0}$(e $\oplus$ <' '>))

Summation (concatenation)

f(e) ~ e $\oplus$ <' '>

s ~ y

H, +, 0 ~ $\mathbb{K}^*$, $\oplus$, <>

Algorithm:

y := <> t.first()
¬t.end()
y : write(t.current())
y : write(' ')
t.next()

Enumerator

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

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

skip_spaces() method

S = (x:infile($\mathbb{K}$), dx:$\mathbb{K}$, sx:Status)

Pre = (x = x' $\wedge$ dx = dx' $\wedge$ sx = sx')

Post = ((sx,dx,x) =

SELECT$_{dx \in (dx', x')}$(sx=abnorm$\vee$dx≠' '))

Selection

t:enor(E) ~ x:infile($\mathbb{K}$) (sx, dx, x:read)

without first

cond(e) ~ sx=abnorm $\vee$ dx≠' '

sx=norm $\wedge$ dx=' '
sx,dx,x:read

next() method

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

Pre = (x = x' $\wedge$ dx = dx' $\wedge$ sx = sx' $\wedge$ (sx=norm $\rightarrow$dx≠' ')) $_{dx \neq ' '}$

Post = (end = (sx'=abnorm) $\wedge$ (¬end $\rightarrow$ (curr, (sx",dx",x"))=$\oplus_{dx \in (dx', x')}$<dx>

$\wedge$ (sx,dx,x)= SELECT$_{dx \in (dx'', x'')}$(sx=abnorm$\vee$dx≠' ')))

Summation (concatenation)

t:enor(E) ~ x:infile($\mathbb{K}$) (sx,dx,x:read)

without first(), cond: dx≠' '

f(e) ~ <dx>

s ~ curr

H, +, 0 ~ $\mathbb{K}^*$, $\oplus$, <>

end := sx=abnorm	
¬end	
curr := <>	–
sx=norm $\wedge$ dx≠' '	
curr := curr $\oplus$ <dx>	
sx,dx,x:read	
skip_spaces()	