Shafaqat Iqbal Assignment 2/ task 2 **6th April 2021**
PMR75L
pmr75l@inf.elte.hu
Group 1

TASK

At every competition of the National Angling Championship, the results of the competitors were recorded and put into a text file. Every line of the file contains the name of the angler the ID of the competition (string without spaces), and the species and the size of the caught fishes (pair of a string without spaces and a natural number). Data is separated by space or tab. The lines of the file are ordered by the name of the anglers. The file is assumed to be in the correct form. Sample line in the file:

```
PETER LAC0512 carp 45 carp 53 catfish 96
```

(1) Give the angler who has caught the most fishes on one contest. Give the ID of the contest and the number of caught fishes, too.
(2) How many contests there are where no fishes were caught?

(1) FIRST PART

Main Program:

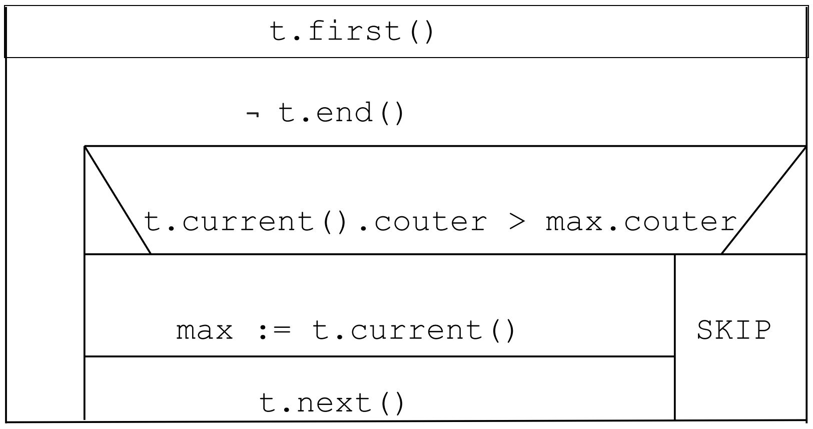
```
A = (f: infile(Line), l: L max:Contest*)
     Line = rec(contest: Contest*, catches: catch*)
     Contest =rec(angler_name: String, contest_id: String, counter:
Int)
     catch = rec(fish:String, Size: Int)
```

NOTE: We don't really use a recored for the catch*, we just read them directly from the file and work with the data itself, without actually storing it somewhere.

New State Space:

A = (t:enor(Contest), max:Contest)
Pre: (t=t')

Post: max = $\mathbf{MAX}_{e \in t'}$ (e.counter)



```
t.first()
¬ t.end()
    t.current().couter > max.couter
        max := t.current()  |  SKIP
    t.next()
```

Anology: Maximum Search(custom Enumerator)

```
t:enor(E) ~ x:infile(Contest) st,e,x:read
f(e)      ~ e.counter
H, <      ~ Z, <
```

CONTEST ENOUMERTOR (ENOR(1))

enor(Contest*)	first(), next(), current(), end()

f:infile(Line)
m_current: ContestEnor
m_end: L
	first() ~ next()
	next() ~ see below
	current() ~ m_current
	end() ~ m_end

Status:(norm, anbnorm)

$A^{next()}$ = (f:infile(Line), m_current: ContestEnor, m_end:L)

$Pre^{next()}$ = (f := f')

$Post^{next()}$ = (sf,df, f := read(f') ^ (m_end := sf=abnorm) ^ ¬(m_end) → m_curren.angler_name = df.angler_name ^ m_current.contest_id = df.contest_id ^

$$m\_current.counter = \sum_{i=0}^{|df.catches|} 1$$

```
sf, df, f:read
m_end, m_current.couter:=sf=abnorm, 0
¬m_end
    m_current.angler_name, m_current.contest_id := df.angler_name, df.contest_id
    i:1..|df.catches|
        m_current.counter := m.current.counter+1
else
    SKIP
```

```
Analogy: Summation(on interval)
t:[m..n] ~ [1.. | df.catches |]
f(i)     ~ 1
s        ~ m_current.counter
H + 0    ~ Z + 0
```

NOTE: In actual c++ implementaion, the enumerator is in a separate compilation unit. The Class name is called Contest.

(2) Second PART

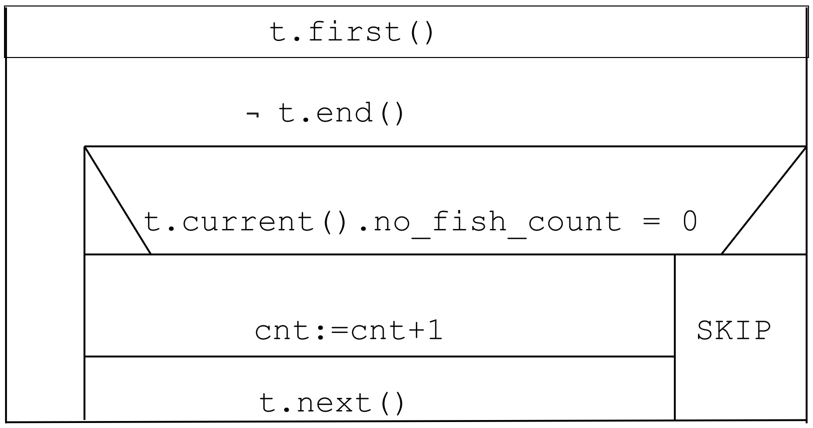
```
A = (f:infile(Line*), cnt:N)
     Line:rec(angler_name:String, contest_id, cates:Catch*)
     Catch=rec(fiss:String, size:N)
```

New State Space:

A = (t:enor(Angler*), cnt: N)
angler_data = rec(id:String, no_fish_count:N)
Pre = (t:=t') ^ t is sorted according to contest_id

Post = (cnt := $\sum_{e\in t'} 1$)
e.no_fisht_count = 0

cond:= e.no_fisht_count = 0 (where no fish were caught)



```
Anology: Count (custom enumerator)
```

```
t:enor(E) ~ x:infile(Line) st,e,x:read
cond(e)  ~  (e.no_fish_count = 0)
```

CONTEST ENOUMERTOR (ENOR(2))

enor(Angler*)	first(), next(), current(), end()

m_contest:Contest
m_curren:Angler
m_end: L
	first() ~ next()
	next() ~ see below
	current() ~ m_current
	end() ~ m_end

Contest=rec(angler_name:String, contest_id:String,counter:Int)

A$^{\text{next()}}$= (m_contest:enor(Contest), m_current:angler, m_end:L)
Pre$^{\text{next()}}$ = (m_contest:=m_contest')
Post$^{\text{next()}}$ = (m_end:=m_contest'.end()) ^ ¬m_end→
m_current.id:=m_contest.current().contest_id ^

m_current.count:= $\sum_{e \in (\text{m\_contest'.current())}}$ e.counter
m_current.id = e.current().contest_id

NOTE: In actual c++ implementation, the custom enor is named Angler and is in a seprate translation unit.

```
m_end := m_contest.end()
¬m_end
    m_current.count, m_current.id := 0,
    m_contest.current().contest_id
    ¬m_contest.end() and
          m_current.id = m_contest.current().contest_id
        m_current.count := m_current.count +
                           m_contest.current().couter
        m_contest.next()
else
    SKIP
```

(3) TESTING:

TAKS ONE:

1. EMTPY FILE:

It should fail. It can also be a possibility that the file is empty and the line is read, but the data in the containers (like angler_name,contest_id, and counter) is empty or zero.

2. CORRECT CASE:

The program must return the expected ouptut. The correct answer must be found out by hand.

3. INCORRECT FILE NAME:

The Program must throw and error stating, “The file you are retrieving data from doesn’t exit.” File name suggestion: “idk.txt”

4. ALL ANGLERS HAVE CAUGHT THE SAME NUMBER OF FISH:

The program must return the first angler.

5. ALL ANGLER HAVE CAUGHT ZERO NUMBER OF FISH

The program must return the first angler.

TASK TWO:

1. CORRENT CASE:

The program must return the expected result. The correct answer must be found out by hand.

2. INCORRECT FILE NAME:

The Program must throw an error stating, “File error!”. File name suggestion: “idk.txt”.

3. On All Contests One Or More Fish Were Caught:

The program must return the count as zero.

4. On All The Contests, No Fish Was Caught:

The count which the program returns must be equal to the number of the contest that happened (or are inside the file).