Middle East Technical University
Department of Computer Engineering

CENG 111

Fall '07

Worksheet 3

REGULATIONS

Due date: 21 January, Monday *(Not subject to postpone)*

Submission: Electronically. You will be submitting your program source code through a file which you will name as `hw3.scm` and submit through the cow system. Resubmission is allowed (till the last moment of the due date), the last will replace the previous, provided you answer the interactive question positively.

Team: There is **no** teaming up. The homework has to be done/turned in individually.

Cheating: All parts involved (source(s) and receiver(s)) get zero.

PROBLEM

Wikipedia defines *tournament* as a competition involving a relatively large number of competitors, all participating in a sport or game. More specifically, A competition involving multiple matches, each involving a subset of the competitors, with the overall tournament winner determined based on the combined results of these individual matches. These are common in those sports and games where each match must involve a small number of competitors: often precisely two, as in most team sports, racket sports and combat sports, many card games and board games, and many forms of competitive debating. Such tournaments allow large numbers to compete against each other in spite of the restriction on numbers in a single match.

In a *single-elimination tournament*, only the top-ranked competitors in a fixture progress; in 2-competitor games, only the winner progresses. All other competitors are eliminated. This ensures a winner is decided with the minimum number of fixtures.

So at the start there are n number of contestants (in our problem this is a power of 2). These n contestants are paired for the first tour of matches.

Each match has a clear winner. So with the completion of that tour we have $n/2$ numbers of winners. The winners will participate the next tour. The losers are eliminated. This algorithm continues until one winner is left, *the champion*.

The whole tournament event can be represented by a binary tree. Here a parent node is holding the winner information of the match performed between its (two) children. With this representation the nodes at the same level are the contestants that enter a particular tour.

Given the 2^k many contestants of the first tour (the leaf of the tree) in a certain order (read off the tree in the left-to-right order) it is possible to set up a tournament tree. Before the first tour no winner is known and all non terminal nodes are unknown (we denote this with a question mark at the winner node). After the matches of the first tour are finished the winners are known. The tree is updated. So as the tournament progresses the question marks at the nodes are replaced by the winners. Therefore such a tree is called a *dynamic tournament tree*, namely a DTT.

In this homework you will write functions that

- construct a DTT,
- update it according to the the match results of a tour,
- return answers to questions about the current view of the tree.

SPECIFICATION

We define a DTT as a fully balanced complete binary tree with 2^k leaf nodes (k is a positive integer). Each node in the tree contains a *datum* of

- either a pair of information represented as a list of the form

  $(playerName \;\; score)$

  if this node is <u>not</u> the root
- or only the $playerName$ if the node is the root.

$playerName$ is either a word or a sentence, while $score$ is a non-negative integer value. If the information about the score, or the player name and its score, is not determined at some point, then the symbol '?' will be used to represent each of that missing info.

As said, a tournament is made up of a series of tours of matches. At a tour, each player in a node of the form

$(playerName \;\; ?)$

will play a match with the player in its sibling node. The winner's $playerName$ will replace the '?' mark in the $playerName$ field in its parent node. The tournament will end when the player name of the root node is determined, who will be the champion of that tournament.

You are expected to write the following functions

`(ddt-cons firstTourMatches)`

This function will construct the initial version of the tournament tree. It accepts one argument: a list of matches of the first tour. Here, each match is a list of the form

$(\; playerName_1 \;\; playerName_2 \;)$

The order of this list of matches has to correspond to the left-to-right reading of the leafs of the tournament tree.

As far as the tree representation is concerned, you are expected to use the below defined representation

$$Representation(tree) = \begin{cases} \textbf{(}\ datum(tree)\ \textbf{)} & \text{[for leaf nodes]} \\ \textbf{(}\ datum(tree) \quad leftchild(tree) \quad rightchild(tree)\ \textbf{)} & \text{[otherwise]} \end{cases}$$

As said previously, $datum(tree)$ has to be represented as

$(\; playerName \quad score \;)$

for all nodes except the root. the datum of the root has to be $playerName$. $leftchild(tree)$ and $rightchild(tree)$ are the left and right children, respectively, each of which itself complies with the definition of the tree representation.

Furthermore you shall implement and make use of the constructor and selector functions of the tree ADT (See Chapter 18 of the textbook). These functions are `make-node`, `datum`, `children`, `leaf?`.

`(ddt-update tournamentTree tourMatches)`

This function will update the tournament tree given the list of match results of some tour played. Each match result is a list of the form

$(\; playerName_1 \quad score_1 \quad playerName_2 \quad score_2 \;)$

The order of $playerName_1$ and $playerName_2$ is <u>unspecified</u> (the order does not have to comply with the left-right order of the tree).

`(dtt-matches tournamentTree)`

This function will return the list of the matches to be played in the coming tour for a given tournament tree. Each match is a list of the form:

$(\; playerName_1 \quad playerName_2 \;)$

The order of the list has to comply with the left-right order of the tree.

`(dtt-champ tournamentTree)`

This function will return the name of champion, if it is determined. Otherwise, it will return `()`.

`(dtt-match-with-max-score tournamentTree)`

This function finds and returns the list of match results of the form

$$(\; playerName_1 \quad score_1 \quad playerName_2 \quad score_2 \;)$$

with the maximum total score $(score_1 + score_2)$. The order of the list has to comply with the left-right order of the tree. It is quite possible that the list has only one match result as member (still the return value will be a list of that match result). If no match has been performed then `()` will be returned.

`(dtt-player-with-max-score tournamentTree)`

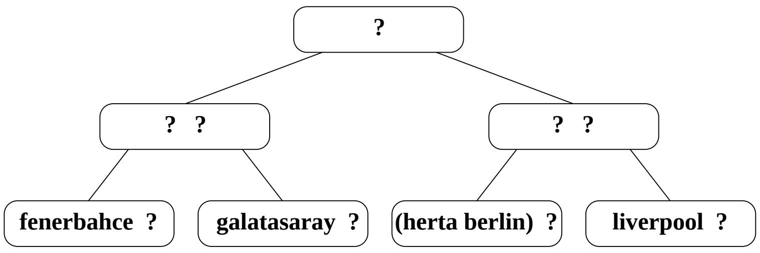
This function finds and returns a list which starts with the maximum total score obtained by one (or possibly more) players (in the whole tournament, so far) and continues with all the $playerName$s of the players that attained this maximum total score. The order of the $playerName$s has to comply with the left-right order of the tree. If no match has been performed then `()` will be returned.

There will be no erroneous input.

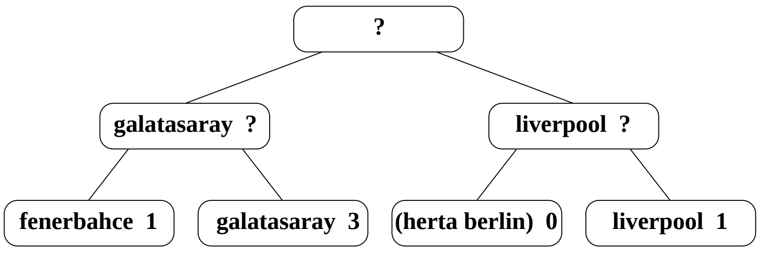
Example

(Below, we use the textbook's convention to differentiate lines of input from output. input lines are prefixed with a '>'. Furthermore indentations in outputs is introduced by us, for readability)

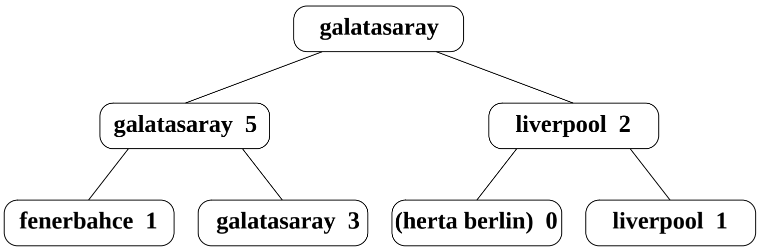
```
> (define tour1 (dtt-cons '((fenerbahce galatasaray)((herta berlin) liverpool))))
> tour1
(? ((? ?) ((fenerbahce ?)) ((galatasaray ?)))
   ((? ?) (((herta berlin) ?)) ((liverpool ?))))
```



```
> (dtt-matches tour1)
((fenerbahce galatasaray)((herta berlin) liverpool))

> (ddt-champ tour1)
()

> (define tour2 (dtt-update tour1
    '((fenerbahce 1 galatasaray 3) (liverpool 1 (herta berlin) 0))))
> tour2
(? ((galatasaray ?) ((fenerbahce 1)) ((galatasaray 3)))
   ((liverpool ?) (((herta berlin) 0)) ((liverpool 1))))
```



```
> (dtt-matches tour2)
((galatasaray liverpool))

> (ddt-champ tour2)
()

> (define tour3 (dtt-update tour2 ’((galatasaray 5 liverpool 2))))
> tour3
(galatasaray ((galatasaray 5) ((galatasaray 1)) ((galatasaray 3)))
             ((liverpool 2) (((herta berlin) 0)) ((liverpool 1))))
```



```
> (dtt-matches tour3)
()

> (ddt-champ tour3)
galatasaray

> (dtt-match-with-max-score tour3)
((galatasaray 5 liverpool 2))

> (dtt-player-with-max-score tour3)
(8 galatasaray)
```