Soft(ware)Ball

Goal

Play as a team to get the maximum score in a 6-innings game.

Playing the Game

The goal of each inning is to solve a ball course puzzle by using programmable components. Each component is loaded with a single program.
To check that a proposed solution works, the components are impersonated by the members of the team.

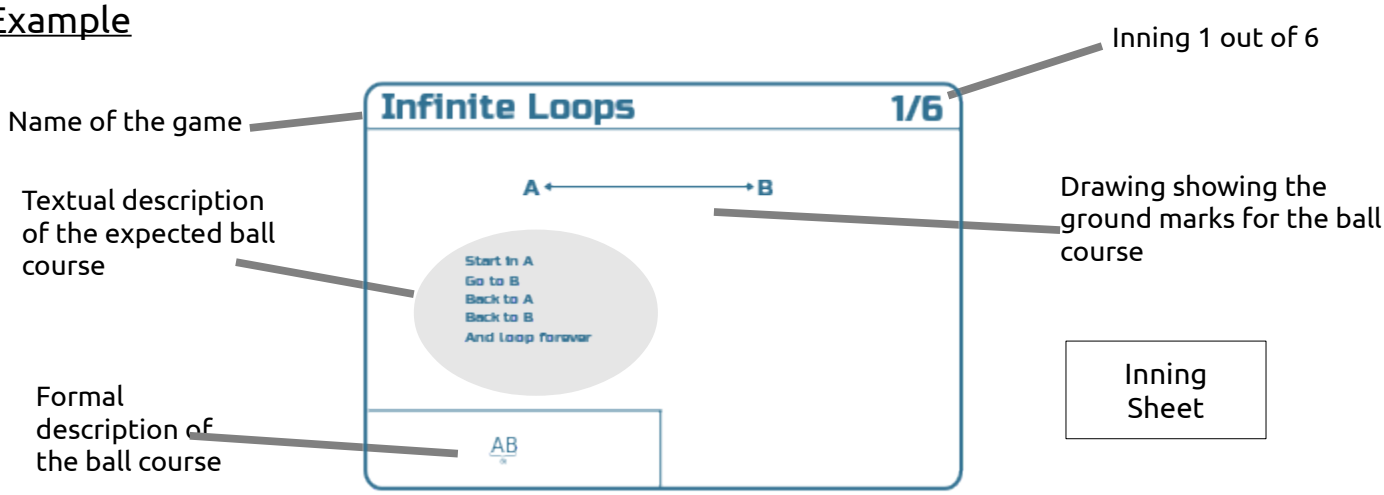
When the team members agree on a good solution, they close the inning and count the score.
With each solved puzzle, the team gets 10 points minus the cost of the programs that were used.

Cost of a program = number of events + number of actions + number of states

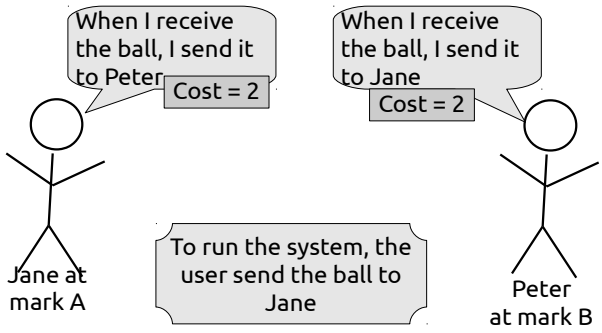
This cost can be quickly estimated by counting the number of verbs in the program.

The programs that were paid in a previous inning can be reused in the following innings without any additional cost. However, these program must be reused as-is without any modification.
Any complete program can be duplicated at zero cost.

Example



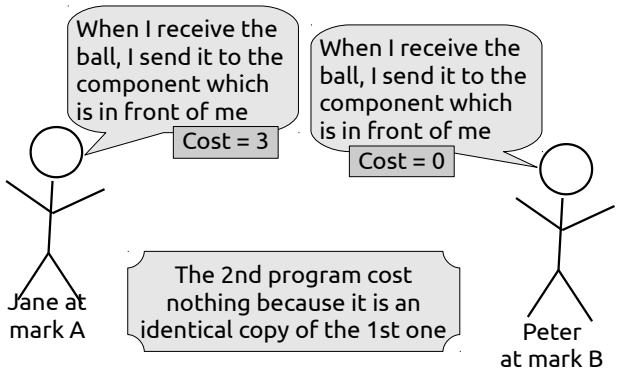
One possible solution



Score sheet after 1st inning

Inning	Points	Verbs	Score	Total
1	10	4	6	6
2	10			
3	10			
4	10			
5	10			
6	10			

Another solution



Score sheet after 1st inning

Inning	Points	Verbs	Score	Total
1	10	3	7	7
2	10			
3	10			
4	10			
5	10			
6	10			

Structure of a program

A program is a set of rules

```
<program> ::= <rule> [<program>]
```

Each rule describes an operation which is executed when an event occurs for one of the actors

```
<rule> ::= "When" <actor> <event> "," <operation>
```

An operation is an action executed by a programmed component. Many operations can be executed one after the other or upon a given condition

```
<operation> ::=
     "me" <action>
     | <operation> [ "and" <operation> ]
     | "if" <condition> "then" <operation> [ "else" <operation> ]
```

An actor is a component identified by a name or by its state

```
 <actor> ::= "me"|"it"|<name>|"the component which" <state>|"a component"
```

The name of the components are defined before the game and cannot be changed

```
<name> ::= "Peter" | "Jane" | ...
```

A condition is an observation of an actor in a given state. Conditions can be grouped using logical operators.

```
<condition> ::=
     <actor> <state>
     | <condition> "and" <condition> | <condition> "or" <condition>
```

An action is an unambiguous physical effort performed by a component. The listed actions are only examples. Players can ask the referee to add additional items to this list. However, players shall keep in mind that components cannot read, write, compute, or talk.

```
<action> ::=
     send the ball to <actor> | go to mark <position>
     | turn <actor> facing to <actor> | turn <actor> over
     | turn <actor> left | turn <actor> right | ...
```

A position is a mark on the ground

```
    <position> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | ...
```

An event is seeing or heraing something happening on the game spot. This includes all possible actions.

```
<event> ::=
     send the ball | hear <word> | <action>
     | receive the ball | receive the ball from <actor>
```

A word is a sound pronounced by the program user (because components cannot talk).

A state is a stable physical description of an actor. The listed states are only examples. Players can ask the referee to add additional items to this list.

```
<state> ::=
     be at mark <position> | be beside <actor>
     | be at the left of <actor> | be at the right of <actor> | ...
```

When a program is complete, it can be transformed into correct English for clarity purpose (using personal pronouns, possessive adjectives, appropriate tenses...) These adaptations do not change the cost of the program.