Part V

Multi-player Games: Auctions and Markets

Multi-player Games: Auctions and Markets

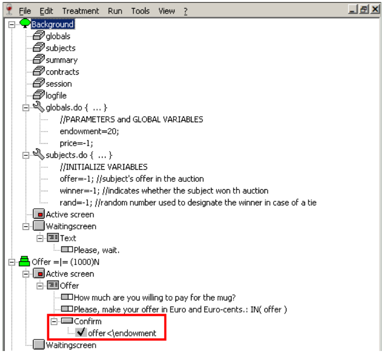
Second price sealed-bid auctions - I

Purpose: elicit the true value of a good (e.g. a mug) for each of the subjects.

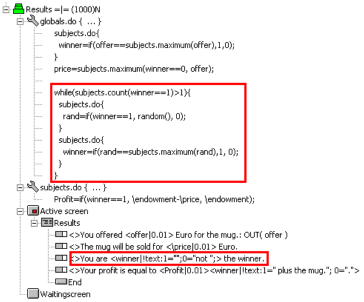
- each subject receives an initial endowment (20 euros)
- each subject makes an offer for the mug (possibly even above 20 euros). The offers are secret and simultaneous.
- the subject who makes the highest bid wins the mug and has to pay a price equal to the second highest bid.
- in case of a tie, the winner is drawn at random among the bidders who submitted the highest bid.

See second_price_auction.ztt

Second price sealed-bid auctions - II



```
Background
    globals
    subjects
    summary
    contracts
    session
    logfile
    globals.do { ... }
        //PARAMETERS and GLOBAL VARIABLES
        endowment=20;
        price=-1;
    subjects.do { ... }
        //INITIALIZE VARIABLES
        offer=-1; //subject's offer in the auction
        winner=-1; //indicates whether the subject won th auction
        rand=-1; //random number used to designate the winner in case of a tie
    Active screen
    Waitingscreen
        Text
            Please, wait.
Offer =|= (1000)N
    Active screen
        Offer
            How much are you willing to pay for the mug?
            Please, make your offer in Euro and Euro-cents.: IN( offer )
            Confirm
                ✓ offer<\endowment
    Waitingscreen
```

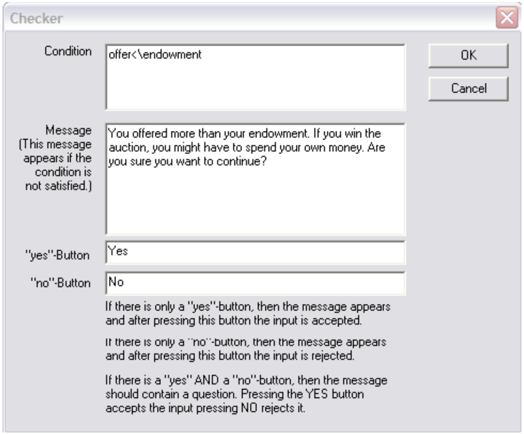


```
Results =|= (1000)N
    globals.do { ... }
        subjects.do{
         winner=if(offer==subjects.maximum(offer),1,0);
        }
        price=subjects.maximum(winner==0, offer);

        while(subjects.count(winner==1)>1){
         subjects.do{
          rand=if(winner==1, random(), 0);
         }
         subjects.do{
          winner=if(rand==subjects.maximum(rand),1, 0);
         }
        }
    subjects.do { ... }
        Profit=if(winner==1, \endowment-\price, \endowment);
    Active screen
        Results
            <>You offered <offer|0.01> Euro for the mug.: OUT( offer )
            <>The mug will be sold for <\price|0.01> Euro.
            <>You are <winner|!text:1="";0="not ";> the winner.
            <>Your profit is equal to <Profit|0.01><winner|!text:1=" plus the mug."; 0=".">
            End
    Waitingscreen
```

checkers

checkers are used to **verify the validity of an input**.

To create a checker, select the button you need to "check", then, from the menu:
Treatment→New checker



Condition: offer<\endowment

Message (This message appears if the condition is not satisfied.): You offered more than your endowment. If you win the auction, you might have to spend your own money. Are you sure you want to continue?

"yes"-Button: Yes

"no"-Button: No

If there is only a "yes"-button, then the message appears and after pressing this button the input is accepted.

If there is only a "no"-button, then the message appears and after pressing this button the input is rejected.

If there is a "yes" AND a "no"-button, then the message should contain a question. Pressing the YES button accepts the input pressing NO rejects it.

while statement - I

To sort the winner at random in case of a tie, we use the while statement:

```
while(subjects.count(winner==1)>1){
 subjects.do{
  rand=if(winner==1, random(), 0);
 }
 subjects.do{
  winner=if(rand==subjects.maximum(rand),1, 0);
 }
}
```

while statement - II

The general use is:

```
while(condition){
  program;
}
```

While the condition is TRUE, the program is executed.

Reminder: loops can be left with the key combination Ctrl+Alt+F5.

Variables into labels

In the `Active screen` of the `Results` stage, we tell each participants whether he is the winner.

In the input **label** we write: `<>You are <winner|!text:1="";0="not ";> the winner.`

This becomes:

- "You are the winner", if the variable `winner` is equal to 1
- "You are not the winner", if the variable `winner` is equal to 0

Do not forget the <> sign at the beginning.

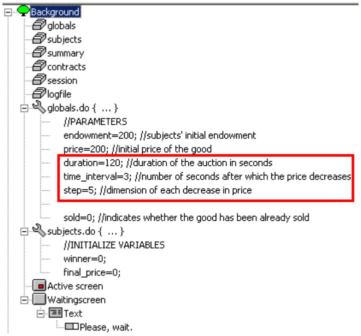
Dutch Auctions - I

A Dutch auction is an auction in which the auctioneer begins with a very high asking price, which is progressively lowered until some participant accepts the auctioneer's price, or until a predetermined time is over.

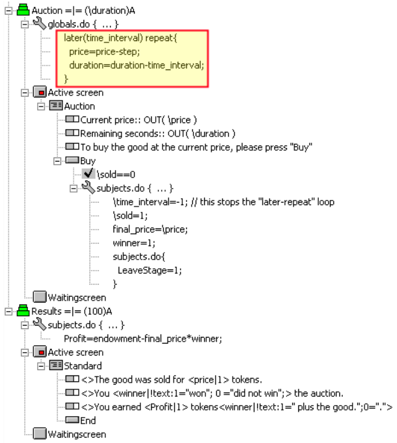
- all subject are buyers
- `global` variables:
  - initial asking price
  - duration of the auction (in seconds) $\rightarrow$ duration
  - step of decrease of the price $\rightarrow$ step
  - frequency of decrease of the price (in seconds) $\rightarrow$ time_interval

See dutch_auction.ztt

Dutch Auctions - II



```
Background
  globals
  subjects
  summary
  contracts
  session
  logfile
  globals.do { ... }
      //PARAMETERS
      endowment=200; //subjects' initial endowment
      price=200; //initial price of the good
      duration=120; //duration of the auction in seconds
      time_interval=3; //number of seconds after which the price decreases
      step=5; //dimension of each decrease in price

      sold=0; //indicates whether the good has been already sold
  subjects.do { ... }
      //INITIALIZE VARIABLES
      winner=0;
      final_price=0;
  Active screen
  Waitingscreen
    Text
      Please, wait.
```



```
Auction =|= (\duration)A
  globals.do { ... }
      later(time_interval) repeat{
        price=price-step;
        duration=duration-time_interval;
      }
  Active screen
    Auction
      Current price:: OUT( \price )
      Remaining seconds:: OUT( \duration )
      To buy the good at the current price, please press "Buy"
      Buy
        \sold==0
        subjects.do { ... }
            \time_interval=-1; // this stops the "later-repeat" loop
            \sold=1;
            final_price=\price;
            winner=1;
            subjects.do{
              LeaveStage=1;
            }
  Waitingscreen
Results =|= (100)A
  subjects.do { ... }
      Profit=endowment-final_price*winner;
  Active screen
    Standard
      <>The good was sold for <price|1> tokens.
      <>You <winner|!text:1="won"; 0 ="did not win";> the auction.
      <>You earned <Profit|1> tokens<winner|!text:1=" plus the good.";0=".">
      End
  Waitingscreen
```

The later statement

In the `Auction stage`, we run the following program:

```
later(time_interval) repeat{
  price=price-step;
  duration=duration-time_interval;
}
```

The **general form** is:

```
later(a) repeat{
  program;
}
```

Expression a is calculated. The resulting number of seconds later, the `program` is executed.

The later statement

In the `Auction stage`, we run the following program:

```
later(time_interval) repeat{
  price=price-step;
  duration=duration-time_interval;
}
```

The **general form** is:

```
later(a) repeat{
  program;
}
```

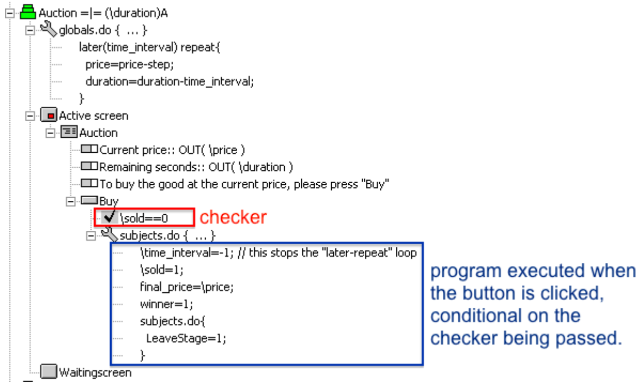
Expression a is calculated. The resulting number of seconds later, the `program` is executed.

Note: to exit the loop, set a to a negative value.

Programs into buttons

When a subject tries to buy the good:

- check that the good has not been sold so far
- run a program to assign the good to the subject



```
Auction =|= (\duration)A
  globals.do { ... }
    later(time_interval) repeat{
      price=price-step;
      duration=duration-time_interval;
    }
  Active screen
    Auction
      Current price:: OUT( \price )
      Remaining seconds:: OUT( \duration )
      To buy the good at the current price, please press "Buy"
      Buy
        \sold==0
        subjects.do { ... }
          \time_interval=-1; // this stops the "later-repeat" loop
          \sold=1;
          final_price=\price;
          winner=1;
          subjects.do{
            LeaveStage=1;
          }
  Waitingscreen
```

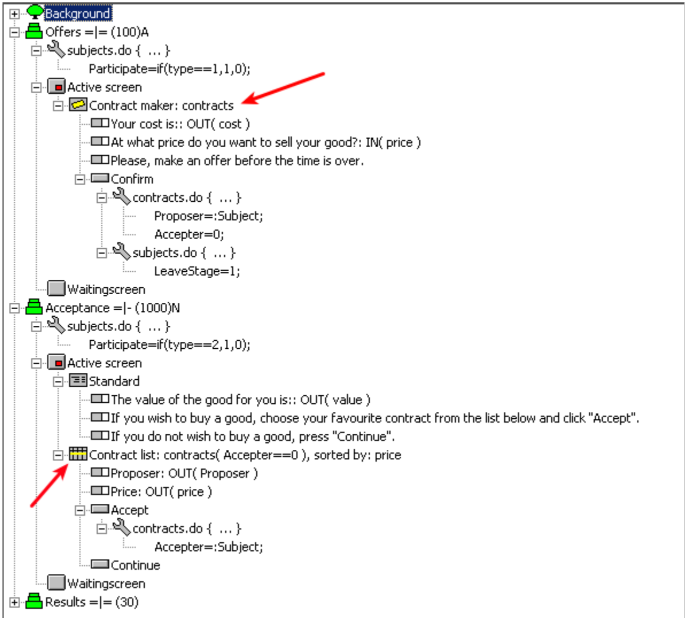
checker
program executed when the button is clicked, conditional on the checker being passed.

Posted offer markets - I

- subjects in the role of *buyers and sellers*
- each seller makes an offer, without knowing the offers made by other buyers.
- buyers act *sequentially*, in random order
- they can see all the sellers' offers that are still open, and choose which one to accept, if any.
- accepted offers are not visible anymore to subsequent buyers.

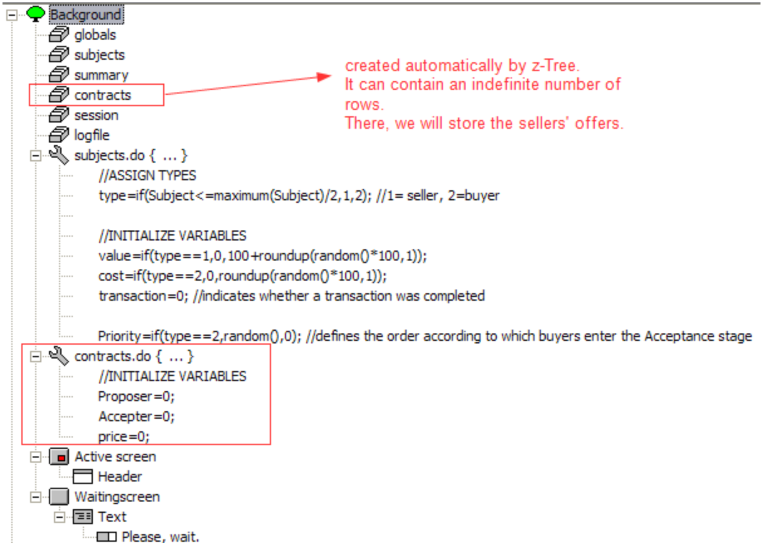
See posted_offers_markets.ztt

Posted offer markets - II



```
Background
Offers =|= (100)A
    subjects.do { ... }
        Participate=if(type==1,1,0);
    Active screen
        Contract maker: contracts
            Your cost is:: OUT( cost )
            At what price do you want to sell your good?: IN( price )
            Please, make an offer before the time is over.
            Confirm
                contracts.do { ... }
                    Proposer=:Subject;
                    Accepter=0;
                subjects.do { ... }
                    LeaveStage=1;
    Waitingscreen
Acceptance =|- (1000)N
    subjects.do { ... }
        Participate=if(type==2,1,0);
    Active screen
        Standard
            The value of the good for you is:: OUT( value )
            If you wish to buy a good, choose your favourite contract from the list below and click "Accept".
            If you do not wish to buy a good, press "Continue".
        Contract list: contracts( Accepter==0 ), sorted by: price
            Proposer: OUT( Proposer )
            Price: OUT( price )
            Accept
                contracts.do { ... }
                    Accepter=:Subject;
            Continue
    Waitingscreen
Results =|= (30)
```

The contracts table

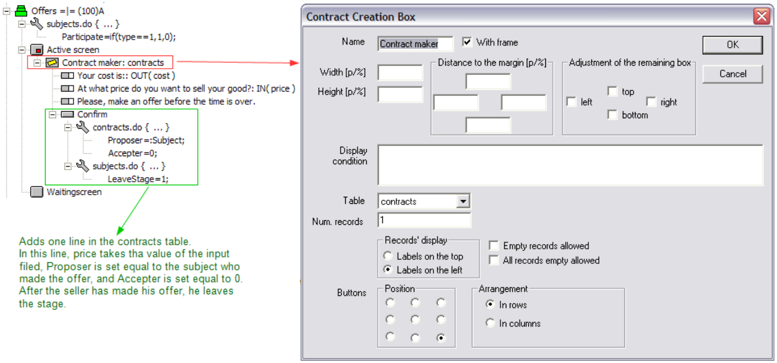


```
Background
    globals
    subjects
    summary
    contracts
    session
    logfile
    subjects.do { ... }
        //ASSIGN TYPES
        type=if(Subject<=maximum(Subject)/2,1,2); //1= seller, 2=buyer

        //INITIALIZE VARIABLES
        value=if(type==1,0,100+roundup(random()*100,1));
        cost=if(type==2,0,roundup(random()*100,1));
        transaction=0; //indicates whether a transaction was completed

        Priority=if(type==2,random(),0); //defines the order according to which buyers enter the Acceptance stage
    contracts.do { ... }
        //INITIALIZE VARIABLES
        Proposer=0;
        Accepter=0;
        price=0;
    Active screen
        Header
    Waitingscreen
        Text
            Please, wait.
```

created automatically by z-Tree.
It can contain an indefinite number of rows.
There, we will store the sellers' offers.

The contracts creation box

Records in the `Contracts` table are created via inputs in the `Contract Creation Box`.



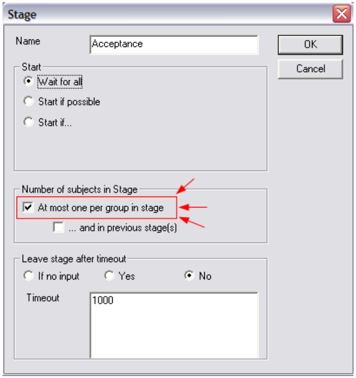
```
Offers =|= (100)A
  subjects.do { ... }
    Participate=if(type==1,1,0);
  Active screen
    Contract maker: contracts
      Your cost is:: OUT( cost )
      At what price do you want to sell your good?: IN( price )
      Please, make an offer before the time is over.
    Confirm
      contracts.do { ... }
        Proposer=:Subject;
        Accepter=0;
      subjects.do { ... }
        LeaveStage=1;
  Waitingscreen
```

Contract Creation Box

Name	Contract maker
With frame	✓
Width [p/%]	
Height [p/%]	
Distance to the margin [p/%]	
Adjustment of the remaining box	☐ left ☐ top ☐ right ☐ bottom
Display condition	
Table	contracts
Num. records	1
Records' display	Labels on the top [✓ Labels on the left]
Empty records allowed	☐
All records empty allowed	☐
Buttons: Position	bottom right ●
Buttons: Arrangement	[✓ In rows] In columns

Adds one line in the contracts table.
In this line, price takes tha value of the input filed, Proposer is set equal to the subject who made the offer, and Accepter is set equal to 0.
After the seller has made his offer, he leaves the stage.

Buyers enter the Acceptance stage one by one


Stage

Name: Acceptance

Start: [✓ Wait for all] Start if possible / Start if...

Number of subjects in Stage: [✓ At most one per group in stage] ... and in previous stage(s)

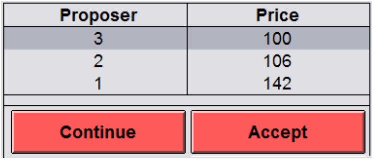
Leave stage after timeout: If no input / Yes / [✓ No]

Timeout: 1000

- the variable `Priority`, if set, defines the order according to which subjects enter the stage.
- if `Priority` is not set, subjects enter the stage in random order.

The contracts list box

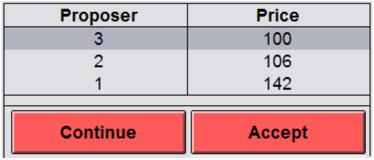
Available offers are displayed in a `Contract list box`.



Proposer	Price
3	100
2	106
1	142

The contracts list box

Available offers are displayed in a Contract list box.



Proposer	Price
3	100
2	106
1	142

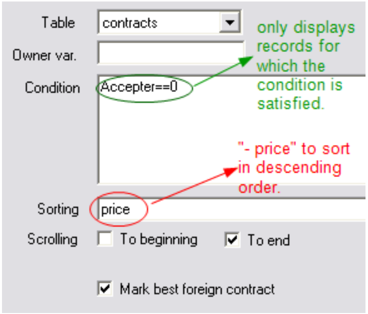
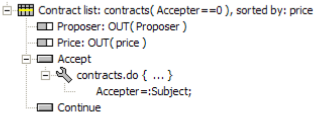


Table	contracts
Owner var.	
Condition	Accepter==0
Sorting	price
Scrolling	☐ To beginning ☑ To end
	☑ Mark best foreign contract

only displays records for which the condition is satisfied.
"- price" to sort in descending order.



```
Contract list: contracts( Accepter==0 ), sorted by: price
  Proposer: OUT( Proposer )
  Price: OUT( price )
  Accept
    contracts.do { ... }
      Accepter=:Subject;
  Continue
```

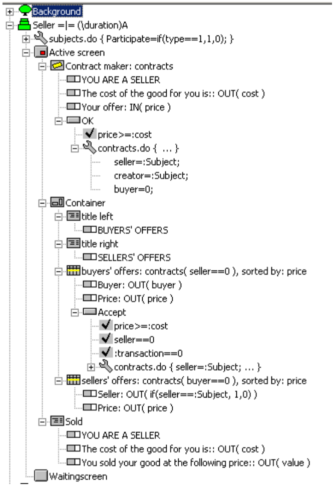
Double auctions - I

see the z-Tree Tutorial, pages: 57-66

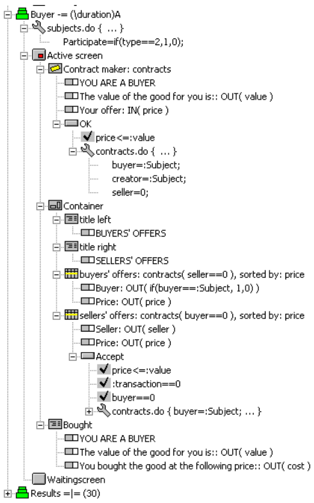
- subjects in the role of buyers and sellers.
- *both sellers and buyers* can make offers, *at the same time*.
- each seller and each buyer can make *more than one offer*.
- sellers and buyers can see *all* the open offers (made by buyers and by sellers)
- each seller can accept *only one* of the buyers' offers.
- each buyer can accept *only one* of the sellers' offers.

See double_auction.ztt

Double auctions - II

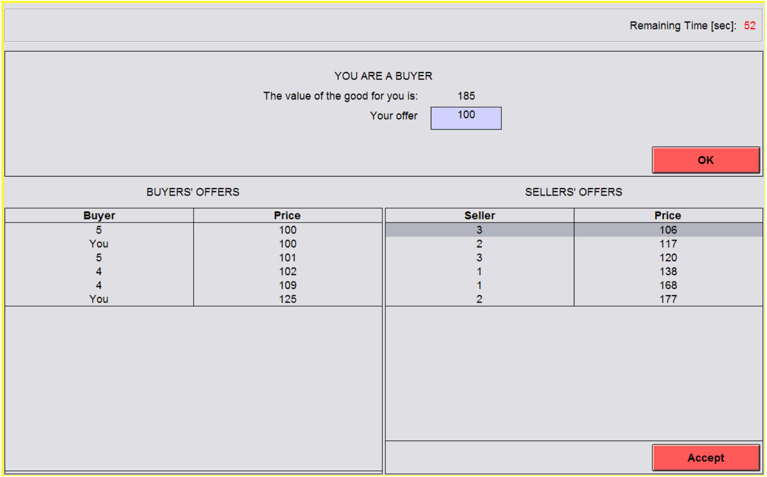


```
Background
Seller =|= (\duration)A
    subjects.do { Participate=if(type==1,1,0); }
    Active screen
        Contract maker: contracts
            YOU ARE A SELLER
            The cost of the good for you is:: OUT( cost )
            Your offer: IN( price )
            OK
                price>=:cost
                contracts.do { ... }
                    seller=:Subject;
                    creator=:Subject;
                    buyer=0;
        Container
            title left
                BUYERS' OFFERS
            title right
                SELLERS' OFFERS
            buyers' offers: contracts( seller==0 ), sorted by: price
                Buyer: OUT( buyer )
                Price: OUT( price )
                Accept
                    price>=:cost
                    seller==0
                    :transaction==0
                    contracts.do { seller=:Subject; ... }
            sellers' offers: contracts( buyer==0 ), sorted by: price
                Seller: OUT( if(seller==:Subject, 1,0) )
                Price: OUT( price )
        Sold
            YOU ARE A SELLER
            The cost of the good for you is:: OUT( cost )
            You sold your good at the following price:: OUT( value )
    Waitingscreen
```



```
Buyer -= (\duration)A
    subjects.do { ... }
        Participate=if(type==2,1,0);
    Active screen
        Contract maker: contracts
            YOU ARE A BUYER
            The value of the good for you is:: OUT( value )
            Your offer: IN( price )
            OK
                price<=:value
                contracts.do { ... }
                    buyer=:Subject;
                    creator=:Subject;
                    seller=0;
        Container
            title left
                BUYERS' OFFERS
            title right
                SELLERS' OFFERS
            buyers' offers: contracts( seller==0 ), sorted by: price
                Buyer: OUT( if(buyer==:Subject, 1,0) )
                Price: OUT( price )
            sellers' offers: contracts( buyer==0 ), sorted by: price
                Seller: OUT( seller )
                Price: OUT( price )
                Accept
                    price<=:value
                    :transaction==0
                    buyer==0
                    contracts.do { buyer=:Subject; ... }
        Bought
            YOU ARE A BUYER
            The value of the good for you is:: OUT( value )
            You bought the good at the following price:: OUT( cost )
    Waitingscreen
Results =|= (30)
```

Double auctions - screenshot



Remaining Time [sec]: 52

YOU ARE A BUYER

The value of the good for you is: 185

Your offer: 100

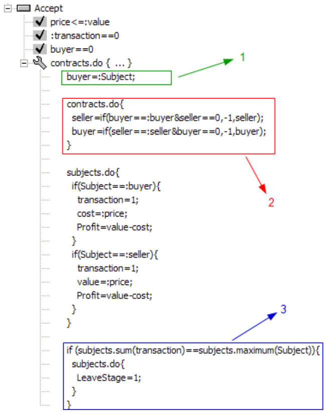
BUYERS' OFFERS

Buyer	Price
5	100
You	100
5	101
4	102
4	109
You	125

SELLERS' OFFERS

Seller	Price
3	106
2	117
3	120
1	138
1	168
2	177

Double auctions - accepting an offer



```
Accept
  price<=:value
  :transaction==0
  buyer==0
  contracts.do { ... }
    buyer=:Subject;

    contracts.do{
      seller=if(buyer==:buyer&seller==0,-1,seller);
      buyer=if(seller==:seller&buyer==0,-1,buyer);
    }

    subjects.do{
      if(Subject==:buyer){
        transaction=1;
        cost=:price;
        Profit=value-cost;
      }
      if(Subject==:seller){
        transaction=1;
        value=:price;
        Profit=value-cost;
      }
    }

    if (subjects.sum(transaction)==subjects.maximum(Subject)){
      subjects.do{
        LeaveStage=1;
      }
    }
```

1. set the ID of the accepter
2. close all other offers by the same proposer and by the same accepter
3. if all subjects have signed a contract, leave the stage

Note: the variable `LeaveStage` is preset in z-Tree.