TYPES.H

tJuryName	Jury name (string)
tParticipantName	Participant name (string)
tNumVotes	Number of votes (integer)
tEUParticipant	Membership of the European Union (boolean)

PARTICIPANT_LIST.H

TYPE OF DATA

tListP	List of participants
itemP	Data of a participant: • participantName (tParticipantName) • numVotes (tNumVotes) • EUParticipant (tEUParticipant)
tPosP	Position of a participant in the participant list
NULLP	Represents a null position in the list of participants

OPERATIONS

- **CreateEmptyListP (tListP) -> tListP**
  - Aim: creates a list and initializes it.
  - Inputs: an uninitialized list.
  - Outputs: an empty list.
  - Postcondition: the list has no data.

- **IsEmptyListP (tListP) -> boolean**
  - Aim: Determines if the list is empty.
  - Inputs:
    - tListP: list to check.
  - Outputs: true if the list is empty and false otherwise.

- **firstP(tListP) -> tPosP**
  - Aim: returns the position of the first item in the list.
  - Inputs:
    - tListP: list to manipulate.
  - Outputs: position of the first element.

- Precondition: the list is not empty.

- **lastP(tListP) -> tPosP**
  - Aim: returns the position of the last item in the list.
  - Inputs:
    - tListP: list to manipulate.
  - Outputs: position of the last element.
  - Precondition: the list is not empty.

- **nextP(tPosP, tListP) -> tPosP**
  - Aim: returns the position of the element following the current one.
  - Inputs:
    - tPosP: position of the current element.
    - tListP: list to manipulate.
  - Outputs: position of the next element or NULLP if it is the last one.
  - Precondition: position is a valid position in the list.

- **previousP(tPosP, tListP) -> tPosP**
  - Aim: returns the position of the element before the current one.
  - Inputs:
    - tPosP: position of the current element.
    - tListP: list to manipulate.
  - Outputs: position of the previous element or null if it is the first.
  - Precondition: position is a valid position in the list.

- **insertItemP(tItemP, tPosP, tListP) -> tListP, boolean**
  - Aim: Inserts an item in the list based on the sort order on the tItemP field.
  - Inputs:
    - itemP: content of the element to insert.
    - tListP: list where to insert.
  - Outputs: list with the item inserted in the corresponding position according to its content and true if it could be inserted, false otherwise.
  - Precondition: the list is initialized.
  - Postcondition: The positions of the list items after the inserted item may have changed.

- **deleteAtPositionP(tPosP, tListP) -> tListP**
    - Aim: removes the element of the given position from the list.
    - Inputs:
        - tPosP: position of the element to delete.
        - tListP: list to modify.
    - Outputs: list without the element corresponding to the given position.
    - Precondition: position is a valid position in the list.
    - Postcondition: Positions of list items after the deleted position may have changed.

- **getItemP (tPosP, tListP) ->tItemP**
    - Aim: retrieves the item of the element of the given position of the list.
    - Inputs:
        - tPosP: position of the searched element.
        - tListP: list where to search.
    - Outputs: content of the position.
    - Precondition: position is a valid position in the list.

- **updateItemP(tItemP, tPosP, tListP) -> tListP**
    - Aim: modifies the item of a list element.
    - Inputs:
        - itemP: new content to assign to the element of the given position.
        - tPosP: position of the element that we want to modify.
        - tListP: list to modify.
    - Outputs: list with the content of the modified element.
    - Precondition: position is a valid position in the list.

- **findItemP(tParticipantName, tListP) -> tPosP**
    - Aim: finds the first element with certain content in the list.
    - Inputs:
        - tParticipantName: content of the searched element.
        - tListP: list where to search.
    - Outputs: position of the element found or null if not found.

JURY_LIST.H

TYPE OF DATA

tListJ	Juror List
itemJ	Jury details: • juryName (tJuryName) • totalVoters(tNumVotes) • validVotes(tNumVotes) • nullVotes(tNumVotes) • participantList(tListP)
tPosJ	Position of a juror on the juror list
NULLJ	Represents a null position in the jury list

OPERATIONS

- **createEmptyListJ(tListJ) -> tListJ**
  - Aim: creates a list and initializes it.
  - Inputs: an uninitialized list.
  - Outputs: an empty list.
  - Postcondition: the list has no data.

- **isEmptyListJ(tListJ) -> boolean**
  - Aim: Determines if the list is empty.
  - Inputs:
    - tListJ: list to check.
  - Outputs: true if the list is empty and false otherwise.

- **firstJ(tListJ) -> tPosJ**
  - Aim: returns the position of the first item in the list.
  - Inputs:
    - tListJ: list to manipulate.
  - Outputs: position of the first element.
  - Precondition: the list is not empty.

- **lastJ(tListJ) -> tPosJ**
  - Aim: returns the position of the last item in the list.
  - Inputs:

        - tListJ: list to manipulate.
    - Outputs: position of the last element.
    - Precondition: the list is not empty.

- **nextJ(tPosJ, tListJ) -> tPosJ**
    - Aim: returns the position of the element following the current one.
    - Inputs:
        - tPosJ: position of the current element.
        - tListJ: list to manipulate.
    - Outputs: position of the next element or NULLJ if it is the last one.
    - Precondition: position is a valid position in the list.

- **previousJ(tPosJ, tListJ) -> tPosJ**
    - Aim: returns the position of the element before the current one.
    - Inputs:
        - tPosJ: position of the current element.
        - tListJ: list to manipulate.
    - Outputs: position of the previous element or null if it is the first.
    - Precondition: position is a valid position in the list.

- **insertItemJ (tItemJ, tPosJ, tListJ) -> tListJ, boolean**
    - Aim: inserts an item in order in the list according to the sort criteria on the tItemJ field.
    - Inputs:
        - itemJ: content of the element to insert.
        - tListJ: list where to insert.
    - Outputs: list with the item inserted in the corresponding position according to its content and true if it could be inserted, false otherwise.
    - Precondition: the list is initialized.
    - Postcondition: The positions of the list items after the inserted item may have changed.

- **deleteAtPositionJ(tPosJ, tListJ) -> tListJ**
    - Aim: removes the element of the given position from the list.
    - Inputs:
        - tPosJ: position of the element to delete.

        - tListJ: list to modify.
    - Outputs: list without the element corresponding to the given position.
    - Precondition: position is a valid position in the list.
    - Postcondition: Positions of list items after the deleted position may have changed.

- **getItemJ (tPosJ, tListJ) ->tItemJ**
    - Aim: retrieves the item of the element of the given position of the list.
    - Inputs:
        - tPosJ: position of the searched element.
        - tListJ: list where to search.
    - Outputs: content of the position.
    - Precondition: position is a valid position in the list.

- **updateItemJ (tItemJ, tPosJ, tListJ) -> tListJ**
    - Aim: modifies the item of a list element.
    - Inputs:
        - itemJ: new content to assign to the element of the given position.
        - tPosJ: position of the element that we want to modify.
        - tListJ: list to modify.
    - Outputs: list with the content of the modified element.
    - Precondition: position is a valid position in the list.

- **findItemJ(tParticipantNameJ, tListJ) -> tPosJ**
    - Aim: finds the first element with certain content in the list.
    - Inputs:
        - tParticipantName: content of the searched element.
        - tListJ: list where to search.
    - Outputs: position of the element found or null if not found.

OPERATIONS.H

OPERATIONS

- **Create(string1, string2, tListJ) -> tListJ**
    - Aim: create a jury with the indicated number of voters.
    - Inputs:
        - string1: name of the jury.
        - string2: number of voters.
        - tListJ: list of jurors.
    - Outputs: list of jurors with one more jury.
    - Precondition: The juror list is initialized.

- **New (string1, string2, string3, tListJ) -> tListJ**
    - Aim: add a participant to the indicated jury.
    - Inputs:
        - string1: name of the jury.
        - string2: name of the participant.
        - string3: EU membership.
        - tListJ: list of jurors.
    - Outputs: list of jurors with a jury with one more participant.
    - Precondition: The juror list is initialized.

- **Vote(string1, string2, tListJ) -> tListJ**
    - Aim: add a participant to the indicated jury.
    - Inputs:
        - string1: name of the jury.
        - string2: name of the participant.
        - string3: EU membership.
        - tListJ: list of jurors.
    - Outputs: list of jurors with a jury with one more valid vote, and in turn with a list with a participant with one more vote.
    - Precondition: The juror list is initialized.

- **Disqualify(string, tListJ) -> tListJ**
    - Aim: disqualify a participant by removing him from all the lists of the juries, starting to count his votes as null.
    - Inputs:

        - string1: name of the participant.
        - tListJ: list of jurors.
    - Outputs: list of jurors with one less participant in all the lists of all the jurors.
    - Precondition: The juror list is initialized.

- **Remove(tListJ) -> tListJ**
    - Aim: remove all jurors without valid votes.
    - Inputs:
        - tListJ: list of jurors.
    - Outputs: list of jurors with one less jury.
    - Precondition: The juror list is initialized.

- **Stats (tListJ) -> (stats)**
    - Aim: Show Stats.
    - Inputs:
        - tListJ: list of jurors.
    - Outputs: statistics.
    - Precondition: The juror list is initialized.

- **Winners (tListJ) -> (winners)**
    - Aim: show the winners inside and outside Europe.
    - Inputs:
        - tListJ: list of jurors.
    - Outputs: winner inside and outside the EU and their respective votes.
    - Precondition: The juror list is initialized.