Am ales sa fac cate un fisier header pentru fiecare tip de structura folosita (coada, stiva, bst, avl) + pentru functiile de citire a informatiilor echipei si jucatorilor.

Unele nume de functii difera (ex print / display) pentru ca am facut prima parte a temei prin aprilie si a doua parte la mijlocul lui mai.

**Cerinta 2**

Numarul de echipe care trebuie sa ramana in joc trebuie sa fie o putere a lui 2, deci din numarul de echipe scad diferenta pana la cea mai apropiata putere a lui 2. 🡺 (*NoToDelete* = numarul de echipe care trebuie sterse)

*deleteTeamByScore()* – sterg echipele cu scorul minim parcurgand intreaga lista de echipe, tinand minte numarul curent si precedentul sau. Tratez cazul in care nodul de sters este = headTeam si cazul in care e un nod oarecare.

**Cerinta 3**

Am ales ca meciurile sa fie reprezentate de o coada cu 2 elemente de tip *dataTeam,* in care se retin echipele care intra in meci. Echipe sunt puse in coada parcurgand lista de echipe din 2 in 2.

Numar rundele cu variabila *round,* pe care o transmit ca parametru. *round* se incrementeaza cu 1 la fiecare intrare in bucla. Dupa fiecare meci numarul de echipe se injumatateste (*NoTeams /=2*)

*getWinnersAndLosers()* – imi creeaza cele 2 stive de invisi si pierzatori in acelasi timp, scotand elementele din coada *matches*.

createMatchesFromStack() – creeaza urmatorul set de meciuri folosind stiva de castigatori.

**Cerinta 4**

In cerinta 3, cand ajung la *NoTeams = 8*, mi se creeaza BST-ul, in functie de scor, parcurgand toata stiva de castigatori.

*printBst()* – pentru a avea punctajele in ordine descrescatoare parcurg BST-ul in DRS.

**Cerinta 5**

populateAvl() – imi populeaza AVL-ul inserand fiecare element al BST-ului in arbore.