FOOTBALL MANAGER GAME SIMULATOR

COLLEGE OF ENGINEERING, GUINDY

INFORMTION TECHNOLOGY

TEAM MEMBERS:

PRATEEK KUMAR SRIRANGAN

ASHOK G

DHILEEPAN S

ABOUT THE PROJECT:

This project makes use of data-structures like splay trees and graphs in C++ to simulate a football game. We have 4 structures namely Player, Team, Splayer and Manager and a class Player\_splay. Player structure is used to hold all the details related to a player. Team structure holds all the information related to a team and Manager structure is used to contain information about the manager and his team. Splayer and Player\_splay is used to contain players of a team in a splay tree.

It is managerial game where you will be asked to create and manage your own club. You can buy or sell players from the market of your choice abiding to two conditions:

1. You must have players from all ten positions to start a match
2. Your purchase cost must not exceed the initial budget given to you.

These players that you do buy or sell are being updated in the splay tree. After having a proper team, you can now start a championship where the rest of the teams are inbuilt.

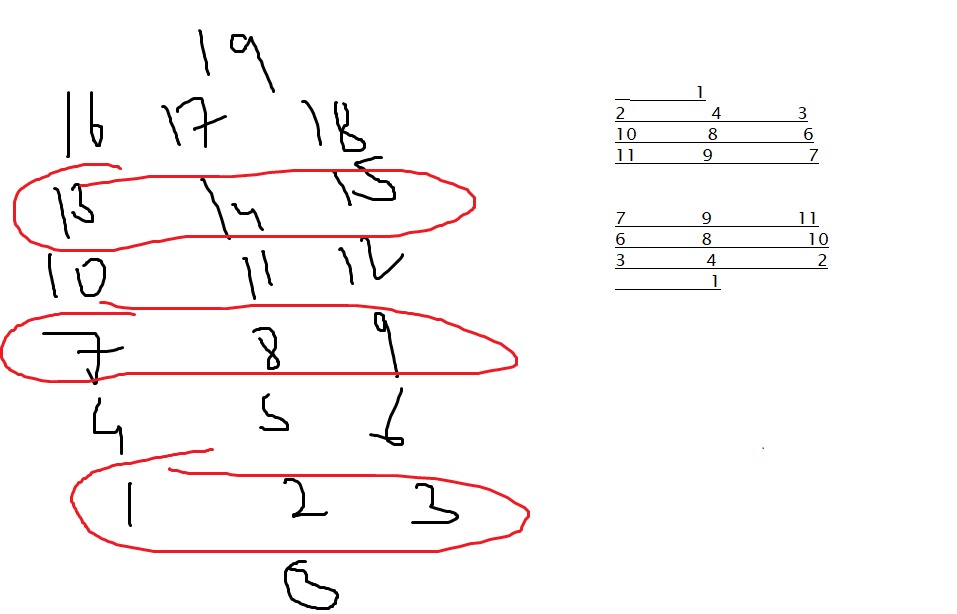
There are 4 types of players:

1. Goal Keeper GK
2. Defender LB,CB,RB(Left-Back, Centre Back, Right-Back)
3. Midfield LM,CM,RM(Left-Mid, Centre Mid, Right-Mid)
4. Attacker LW,CF,RW(Left-Wing, Centre Forward, Right-Wing)

You can only pass according to the hierarchy, a GK can pass only to a defender of his choice and he in int turn can only pass to a midfielder of his choice, who in turn can pass only to attacker of his choice. The attacker can shoot the ball at the opponent GK. Now how interception by opponent occurs is that we compare the rating of players(opponent) present in between the passing lanes and if their rating exceeds that of the current players, they can intercept the ball and start playing from that line. Example: A pass from GK to defender can not be intercepted by the opposition. A pass from defender to midfielder can be intercepted by the nearest midfielder of the opposition team, if he does intercept, he will start passing it to his attacker.

The match is played only for 1 goal. First goal wins, no second chances. For a win you get 3 points and a point for draw and none for a loss.

Below picture shows how the players are arranged in the vector<int> playground.



Football Playfield (graph vertices number)

DATA STRUCTURES USED:

* STRUCTURES
* SPLAY TREES
* GRAPHS
* CLASSES have also been used.

TIME COMPLEXITY:

* Splay trees have a time complexity of O(log n) time for all operations as they depend only on splaying. Here n is number of players, it ranges from 0 -10 in a team.
* Worst case would be 2.33 units of time.(log 10 =2.33)
* Graphs have a time complexity of O(V+E). Here we have 20 vertices as players and edges can range from 2 to 4 for each player.
* Worst case would be 12 units of time.

USER DEFINED FUNCTIONS:

PLAYER:

player(string n,string pos,int id,int pno,float orat,float dr,float value,int playing):

Constructor for the structure player

void show\_player():

A function that prints all the data members of structure player.

void display\_playing();

This function displays Yes or No to whether a player is playing in the field or not.

float return\_value();

This function returns the market value in millions for a player

int return\_pid();

This function returns the ID of a particular player.

int return\_player\_position\_no();

This function returns the position number of a particular player. (We have different numbers for different positions on the field)

void assign\_to\_team();

This function assigns 1 to bool playing. It tells us that this player is playing in the team

float return\_or();

This function gives the offensive rating of a particular player.

float return\_dr();

This function gives the defensive rating of a particular player.

TEAM:

team(int,int,string,vector<player>);

This is a constructor to the structure team.

float return\_player\_offense\_rating(int pno);

This function uses float return\_or from player to return a player’s offensive rating in a team.

float return\_player\_defense\_rating(int pno);

This function uses float return\_dr from player to return a player’s defensive rating in a team.

void display\_player\_offense\_rating(int pno);

This function uses float return\_or from player to print a player’s offensive rating in a team.

void display\_player\_defense\_rating(int pno);

This function uses float return\_dr from player to return a player’s defensive rating in a team.

void return\_player\_name(int pno);

This function uses int return\_pid from player to return a player’s name in a team.

int calpoints();

This function returns the total points obtained by a team in a tournament.

MANAGER:

manager();

This is a default constructor for structure manager.

float return\_budget();

This function returns the total budget allotted for a team when they register.

float update\_budget\_buy(float);

This function updates the budget of a team once they buy a player.

float update\_budget\_sell(float);

This function updates the budget of a team once they sell a player.

bool budget\_check(float);

This function checks whether the club and manager have enough money to buy a particular player from the market.

PLAYER\_SPLAY and SPLAYER:

All the players bought are put into a splay tree according to their id  
player\_splay();

This is a constructor to the class player\_splay.

splayer \*RR\_Rotate(splayer \*k2);

This function is used to splay the element to the root.

splayer \*LL\_Rotate(splayer \*k2);

This function is used to splay the element to the root.

splayer \*Splay(float rating, splayer \*root);

This function actually splays the element to the root.

splayer \*New\_Node(float rating);

This function creates a object of datatype player\_splay.

splayer \*Insert(float rating, splayer \*root);

This function is used to insert or buy players into the splay tree.

splayer \*Delete(float rating, splayer \*root);

This function is used to delete or sell players.

void inorder(splayer \*root,vector<player> p);

This function is used to print in order traversal of splay tree.

bool if\_player\_already(vector<player> p,int pno);

This function makes sure you cannot buy a player twice.

MAIN:

float return\_total\_rating\_pid(int pid,vector<player> p);

This function returns total rating of a player in the field.

void return\_player(float rating,vector<player> p);

This function prints the details of a player int the team using show\_player() of player.

void add\_players\_to\_field();

This function adds the player of a team inside the field.

int toss(int coin);

This function is initialise the toss of match.

void match(int mno, manager &m, team &t);

This function displays the actual match that is shown in the output. From here all other functions if our game run.

int return\_index(vector<player>p,int pid);

This function returns the index of player in vector whose pid is passed.

void check\_team(vector<player> &p);

This function checks whether the player is playing in the team going into the match.

bool check\_key\_players(vector<player> p);

This function checks whether all the players required for a team to play a match exist or not.

void swap\_players(vector<player> &p,int pid);

This function is used to swap players of team.(Only if they belong to the same position)

void pass(int,int,manager& m,team &t2);

This function is where the game runs, here passes are made to other players as well they are intercepted. It is called recursively until it is intercepted or reaches a goal.

void return\_position\_name(int);

This function prints the name of the positions of all the players of a team when called.

bool intercept(int from,int to);

This function checks whether a pass is intercepted by the opponent player.

int return\_posno\_verticeno(int);

This function returns the position of player inside the field vector playground.

int goal\_chance(team t);

This function calculates the chances of a goal between the striker of one team and goal keeper of another.

void simulate\_matches(team &t1, team&t2);

This function updates the points table after a match over.

void championship\_winners(team t1,team t2,team t3,team t4);

This function displays the winners of a tournament based on the points table.

bool if\_championship\_winner(team,team,team,team);

Returns true if manager team has won the championship or else false.

**Header Files**

<vector>

<cstdlib>

<math.h>

**MODULES:**

There are 3 modules for our Project

1.fb.h

2.fb.cpp

3.football.cpp

OUTPUT:

