Hacettepe University

Computer Science and Engineering

BBM203 Programming Lab.

Assignment 3

Ahmet Faruk BAYRAK

21426716

25.11.2018

R.A. Merve OZDES

1. Software Using Documentation

1.1. Software Usage

The program runs on command prompt and the output will be printed out on the txt file. Program takes inputs as a command line argument. To run the program: firstly you go to the file path which contains main.cpp file. And write the “g++ -std=c++11 main.cpp” command. After that program is ready to start. There are 3 arguments: first one is information of footballers and goals (input.txt), second one is information of operations (operations1.txt), and the last argument is name of output txt (output1.txt). Example of command line argument is: ./a.out input1.txt operations1.txt output1.txt

1.2. Error Messages

Program does not have any error messages.

2. Software Design

2.1. Description of the Program

2.1.1. Problem

In this experiment, the problem that has been expected us to design a simple linked list structure which contains footballers and their informations. Program reads the information about footballers such as their name, team name, away team of scored, minute of goal and match ID. After that program creates linked list structure then reads operations.txt. Finally program prints the commands. Main aim of this assignment was usage of linked list and doubly linked list.

2.1.2. Solution

For the solution, first of all I took the arguments and assign them. Since program does not know the number of line in the input file, program finds out it while calling determine_line_size() function. After that program calls read_and_create_linked_list() function for reads the input.txt and create linked list. In the function firstly reads the input txt and adds them to temporary array. When the filling of array is finished, program stars to create linked list structure while calling create_footballer_node_sequential() function. Program firstly check that whether footballer is in the linked list to avoid duplicates. If footballer is already in linked list then program does not add football but add new goal information while calling insert_goal_node() function. After creation of linked list is finished program reads the operations.txt and starts to print the commands.

One problem in the creation of linked list was, adding the footballers sequentially. Every addition, program check the right position of footballer according to their names. The position might be head or tail or between them, that

was a little hard to determine.

There are 8 titles for print:

- 1)THE MOST SCORED HALF: In this command, program prints the most scored half. If the most scored half is first half, program prints 0 but if the most scored half is second half then program prints 1.

Program has two count for first and second half goals. According to these 2 count, program determine which half is most scored.

- 2)GOAL SCORER: In this command, program prints the most scored footballer name. There may be more than one footballer with the same goal count, in this case all should be printed.

Program keeps the number of goals of footballers. According to this information, program determines the goal scorer.

- 3)THE NAMES OF FOOTBALLERS WHO SCORED HAT-TRICK: In this command program prints the names of footballers who scored hat-trick. Hat-trick is the scoring of three goals in a game by one player.

To determine the players who scored hat-trick, first of all program adds match id of footballers to array. After that program checks the array and make count of match IDs. If the match IDs are equal then add count to one. Program does that for every match ID. After that checks that if count is greater than 2, it means in this match, player scored more than 2 which means hat-trick.

- 4)LIST OF TEAMS: In this command, program prints the names of teams. One problem was, program has to print them without duplicates. Firstly program adds the first team of footballer to array. After that every footballer program checks whether the team name of footballer is in the array. If the team name is already in the array then program does not add it.

- 5)LIST OF FOOTBALLERS: In this command, program prints the all of the footballers names. Program already created the linked list sequentially so there is nothing to do but prints them.

- 6)MATCHES OF GIVEN FOOTBALLER: In this command, program prints the matches of footballers which is the first line of operations.txt. This function split the line according to “,” and determine the footballers. After that prints their informations.

- 7)ASCENDING ORDER ACCORDING TO MATCH ID: In this command, program prints the matches of footballers which is the second line of operations.txt according to match ID ascending order. This function firstly split the line according to “,” and determine the footballers. After that for every footballers, add their match IDs to an array. For removing duplicates and sorting program uses vector functions

which are sort(), erase(), begin(), end() and unique()

- 8)DESCENDING ORDER ACCORDING TO MATCH ID: In this command, program prints the matches of footballers which is the second line of operations.txt according to match ID descending order. This function firstly split the line according to “,” and determine the footballers. After that for every footballers, add their match IDs to an array. For removing duplicates and sorting program uses vector functions which are sort(), erase(), begin(), end(), unique() and reverse().

When everything is done, program calls clear_linked_lists() function for deleting linked list for avoiding memory leak.

Here is the functions that I used in the program:

- **void** display_linked_list()

This function prints all of the linked list. I used this function for check whether my creation of linked list is right.

- **int** findByID(client *clients, **char** clientName, **int** clientNumber)

First parameter is clients. Second parameter is name of client. And the last argument is number of client.
This function returns the index of clients which name is second parameter.

- **void** create_footballer_node_sequential(string footballer_name, string footballer_team_name, string away_team_name, **int** minute_of_goal, **int** match_ID)

First parameter is the name of footballer. Second one is team name of footballer. Third one is away team name. Fourth one is minute of goal. Last one is match ID.
In this function, program creates linked list sequentially according to name of footballer. Function determines where to add in the linked list.

- **void** insert_goal_node(string footballer_name, string away_team_name, int minute_of_goal, int match_ID)

First parameter is the name of footballer. Second parameter is name of away team. Third parameter is the minute of goal. And the last argument is match ID
This function adds goal to footballer which is already in linked list.

- **bool** search_footballer_name(string footballer_name_searched)

Parameter is the name of footballer which will be searched in the linked list.

This function searches the given footballer name in the linked list and if the name is already in the linked list returns true otherwise returns false.

- **void** clear_linked_lists()

This function traverse in the linked list and delete them. Program calls this function when everything is done. This function aims to avoid memory leak.

- **int** determine_line_size(ifstream& inFileTemp)

This function calculates the number of lines in the input file.

- **void** most_scored_half(ofstream& outFile)

- **void** goal_scorer(ofstream& outFile)

- **void** hattrick(ofstream& outFile)

- **void** list_of_teams(ofstream& outFile)

- **void** list_of_footballers(ofstream& outFile)

- **void** matches_of_given_footballer(string line, ofstream& outFile)

- **void** ascending_order_match_id(string line, ofstream& outFile)

- **void** descending_order_match_id(string line, ofstream& outFile)

I explained these command functions above.

2.1.3. Algorithm

1) Program takes arguments and assign them.
2) Program calls determine_line_size() functions to determine number of line in the input.txt
3) Program calls read_and_create_linked_list() functions for reading input.txt and creating linked list sequentially according to name of footballers.
4) Program reads operations.txt.
5) Program calls most_scored_half() function.
6) Program calls goal_scorer() function.
7) Program calls hattrick() function.
8) Program calls list_of_teams() function.

9) Program calls list_of_footballers() function.
10)Program calls matches_of_given_footballer() function.
11)Program calls ascending_order_match_id() function.
12)Program calls descending_order_match_id() function.
13)Program calls clear_linked_lists() function for deleting linked list.