Practice of Information Processing
Final Assignment
2024/11/13

1. Japanese football league ranking program: J_score.c

Create a program J_score.c that generates a standings table based on the score data, J_dataset.csv, of the Japanese football league and outputs it to a text file. This program follows the specifications below.

＜Specifications＞

- Each line of the CSV file is listed in the order of "0: team name 1: number of wins 2: number of draws 3: number of loses 4: total goals scored (GF: goals for) 5: total goals conceded (GA: goals against)" separated by commas. Currently, the data is in ascending order by team names.
- The teams with the higher “win points” should be ranked higher. If the "win points" are the same, the team with the largest "goal difference" will be ranked higher. If there is still a tie, the team with the highest "total goals scored (GF: goals for)" will be ranked higher.
- “Win points” are calculated based on 3 points for winning, 1 point for drawing, and 0 points for losing.
- “Goal difference” is the “total goals scored (GF: goals for)” minus the “total goal conceded (GA: goals against)”.
- In addition to the original contents of the CSV file, add "ranking", "win points", and "goal difference" to the table, and sort them by "ranking" in the output file. It is desirable to output in an easy-to-read format.

＜Submissions＞

- (Mandatory) Create the above program using the template, J_score_template1.c. Refer to read_csv_sample.c on how to read the CSV file. Choose bubble or selection sort for sorting algorithm. To get credit, you must submit
  - Program code: J_score.c
  - Execution result: J_score.txt
- (Optional) Create the same program using the template, J_score_template2.c, which sorts a pointer array instead of the data array.
- (Optional) Instead of writing the macro definition,

  #define TEAM_NUM 18

  count the number of teams in the CSV file and allocate the corresponding size of arrays by using malloc.

- (Optional) In the ranking function rank_score (), use more efficient algorithm such as heap sort.
- (Optional) Any other improvements or modifications.

2. Root finding algorithm: bisection.c

Create a program bisection.c that finds the roots (i.e., zeros) of a continuous function f(x), using the following bisection method.

① If f(a) and f(b) have opposite signs, there must be at least one zero on the interval a<x<b.

② Evaluate f(c) at c=(a+b)/2.

③ If f(a) and f(c) have opposite signs, there must be at least one zero on the interval a<x<c. Go to ② after substituting a=a and b=c.

If f(a) and f(c) are the same sign, there must be at least one zero on the interval c<x<a. Go to ② after substituting a=c and b=b.

<Submissions>

- (Mandatory) Calculate a root of f(x)=x/12 + sin(x) by repeating the above iteration until the error gets smaller than 10^{-8}. To get credit, you must submit
  - ✧ Program code: bisection.c
  - ✧ Execution result: a root of f(x)=x/12 + sin(x)
- (Optional) Try to find all roots of f(x)=x/12 + sin(x).
- (Optional) Use recursive call. (However, it would be better to count and output the number of iterations.)
- (Optional) Use different root finding algorithm such as Newton's method.
- (Optional) Make any other effort to find roots of various functions.

** If you want to get higher grades such as A and AA, try as many optional assignments as possible and submit your codes. Please summarize what you have done in a PDF file* report2_(Student ID).pdf

Deadline: November 30th