EEEC10008(515169) S23

Programming Homework #0

Deadline 2023/02/12 23:59:59

INSTRUCTIONS:

1. **This homework is a prerequisite for the course. Please submit your zipped file on e3 by the deadline of 2023/02/12 23:59:59. A failure to submit will result in withdrawal from the course.**
2. **Your source code files should be placed in a folder called [StudentID]_hw0, compressed into a zip file (ex: 0810700_hw0.zip), and uploaded to e3 in a timely manner.**
3. **Note that the source code files should be named p1.cpp, p2.cpp, p3.cpp, p4.cpp and p5.cpp.**
4. **Testcases for each problem can be downloaded from e3 and online.**
5. **You may contact the Teaching Assistant (Jimmy Liu) at jimmy61210.ee10@nycu.edu.tw with any questions you may have.**
6. **Incorrect file names will deduct five points from your score per problem.**
7. **We strongly suggest to use online GDB (https://www.onlinegdb.com/) to wirte your C++ codes (Select C++ as the Language).**

Good luck!

PROBLEM 01

A puzzle is a game, problem, or toy that tests a person's ingenuity or knowledge. In a puzzle, the solver is expected to put pieces together (or take them apart) in a logical way, in order to arrive at the correct or fun solution of the puzzle. There are different genres of puzzles, such as crossword puzzles, word-search puzzles, number puzzles, relational puzzles, and logic puzzles. The academic study of puzzles is called enigmatology.

Puzzles are often created to be a form of entertainment but they can also arise from serious mathematical or logical problems. In such cases, their solution may be a significant contribution to mathematical research.

In this problem, you are given a list of puzzle pieces in random order and random orientation, just like a real puzzle. You need to write a c++ program that can combine these puzzle pieces into an image.

Each puzzle piece contains at most four "Tabs" and "Blanks" joints denoted by a random integer, and a partial image denoted by ASCII characters. The partial image has the same height and width. The input format, output format, and execution are shown below.

✓ The input file format:
- The first line indicates the size of the puzzle partial image.
- The following lines each represent a puzzle piece, with the following format
  ```
  <leftJoint> <topJoint> <rightJoint> <downJoint> "<image>"
  ```
- Every joint is unique, except for 0.
- If a joint is 0, it means that it is a flat side of the puzzle and cannot be connected to other puzzle pieces.
- If a joint is positive, it means it is a "Tab". If a joint is negative, it means it is a "Blank". A "Tab" and a "Blank" can be connected if they have the same absolute value. Ex, joint 3 and joint -3 can be connected.
- The image is flattened to a 1D string, by appending each row to the end of the previous row. Ex:
  ```
                    abc
            Image   def    is flattened to  "abcdefghj"
                    ghj
  ```
- The output file is multiple rows of strings, which is the combined image.
- Execution
  ```
  ./p1 [input_filename] [output_filename]
  ```
- Other constraints:
  - The joint id will be between -10000 to 10000.
  - There will be at most 10000 puzzle pieces.

- The size of the puzzle partial image will not exceed 100.
- **The order of the puzzle pieces is random.**
- **The orientation of the puzzle pieces is random, except for the first puzzle piece (line 2 of the input file). So you have to try to rotate some pieces for them to connect correctly.**
- **When rotating puzzle pieces, you must rotate the puzzle partial image too (rotate the characters' position only, you don't have to rotate the characters themselves).**
- **The orientation of the output image is the same as the first puzzle piece read in.**

✓ You don't have to implement any input error checking.

✓ Scoring (Total 30%):
  - Pass demo script (15%)
  - Need to pass the demo script, then show the output image to TA, and answer what the output image is. (15%, 5% each for 2.out, 3.out, 4.out)

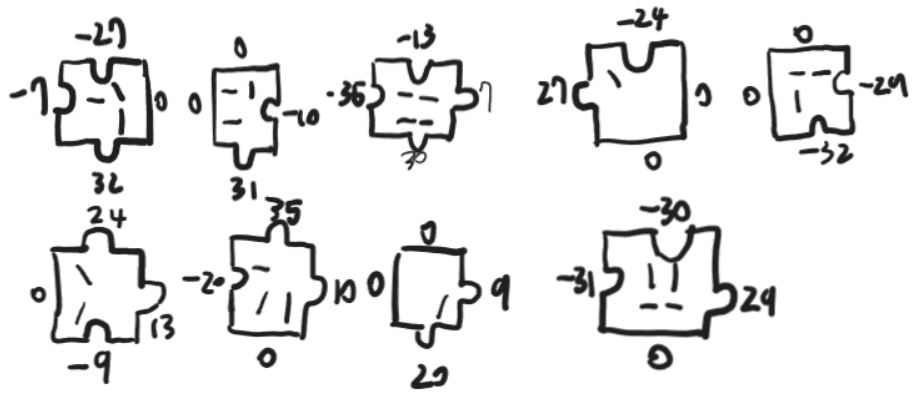
✓ **File name rule:**
  - cpp file name: `p1.cpp`
  - executable file name: `p1`

✓ Example:
  - Input file: 1.in

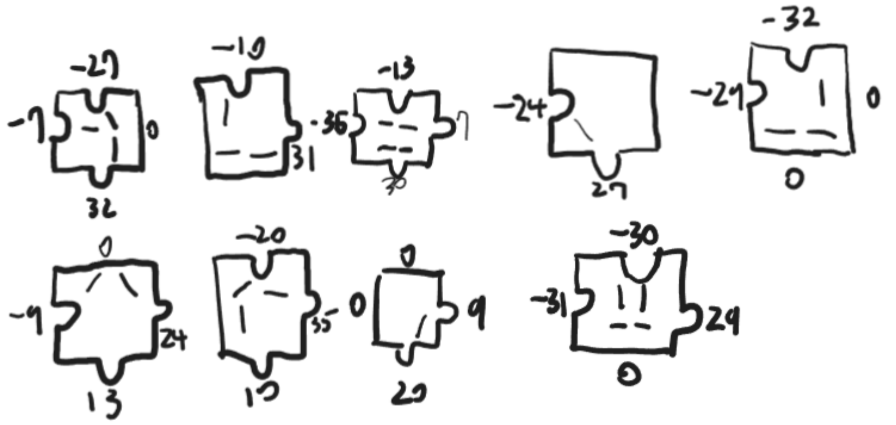
```
2
-7 -27 0 32 "_\ |"
0 0 -10 31 "-|- "
-35 -13 7 30 "____"
27 -24 0 0 "\   "
0 0 -29 -32 "--| "
0 24 13 -9 "\ / "
-20 35 10 0 "_ /|"
0 0 9 20 "   /"
-31 -30 29 0 "||--"
```

- Visualization

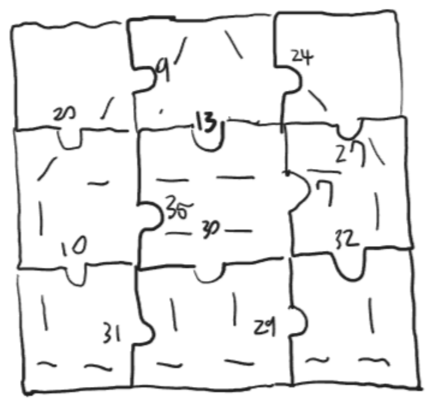
  Input



  Correct rotation



  Output



- Output file: 1.out

```
   /\
  /   \
/_____\
|  __  |
|  ||  |
------
```

PROBLEM 02

In the field of computer science and math optimization, one of the famous solving strategies is "metaheuristic" which is good at find, generate, or select a sufficient good solution to an optimization problem. Compared to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. Because the process of finding optimal solution is dependent on the set of random variables generated, many new metaheuristic algorithms are inspired by natural systems. Taking simulated annealing algorithm for example, the way finding optimal solution is inspired by the process of temperature-cooling in metallurgy. In addition, another metaheuristic algorithm is originated by the simplified society model and simulating the unpredictable exercise of flock of birds in a multi-dimensional space. In this problem, we are going to implement this algorithm and solve an optimization problem.

In this algorithm, we have the following variables

Name	Depiction
Dim	Dimension of the space which also means the variables' number in optimization problem
Num_b	Number of birds
$\mathbf{x_i}$	Represents the position of bird i and contains Dim numbers. ($\mathbf{x_i} = [x_{i,0}, x_{i,1}, \ldots\ldots, x_{i,(dim-1)}]$)
Max_boundary	The maximum value of each dimension. (max = $[x\_max_0, x\_max_1, \ldots\ldots, x\_max_{(dim-1)}]$
Min_boundary	The minimum value of each dimension. (min = $[x\_min_0, x\_min_1, \ldots\ldots, x\_min_{(dim-1)}]$
$f(\mathbf{x_i})$	A cost function that returns a score for judging how good the solution $\mathbf{x_i}$ is.
Max_cycle	Maximum cycles for iteration
$\mathbf{v_i}$	Represents the velocity of bird i and contains Dim numbers. ($\mathbf{v_i} = [v_{i,0}, v_{i,1}, \ldots\ldots, v_{i,(dim-1)}]$)
k	A coefficient of velocity maximum
$\mathbf{v^{max}}$	Represents the maximum velocity and contains Dim numbers. ($\mathbf{v^{max}} = [v_0^{\mathbf{max}}, v_1^{\mathbf{max}}, \ldots\ldots, v_{(dim-1)}^{\mathbf{max}}]$)
w(cycle)	A function of velocity iteration coefficient
C1, C2	Two acceleration-constant
R1, R2	Two random number arrays in size of (Max_cycle * Num_b * Dim)
$\mathbf{p_i}$	Represents the best position that bird i has ever been.
g	Represents the best position that this flock of the birds have ever been.

- ✓ Initialization
  - Initial position: $\mathbf{x_{i,j}} = x\_min_j + (x\_max_j - x\_min_j) * i / (Num\_b – 1)$
  - Initial velocity: $\mathbf{v_{i,j} = 0}\ \forall \boldsymbol{i}, \boldsymbol{j}$
- ✓ Setting
  - **k** = 0.5
  - $v_i^{max} = (x\_max_i - x\_min_i) * \mathbf{k} / 2$, v_i^{max} is a double number.
  - Dim, Num_b, Max_cycle, C1, C2, Max_boundary, Min_boundary: define in setting file (*.set)
  - R1: define in rand1 file (*.rand1)

  - R2: define in rand2 file (*.rand2)
- ✓ Iteration update (in the following order)
  - $w(cycle) = w_{max} – (w_{max} - w_{min}) * cycle / Max\_cycle$

    $cycle \in [0, Max\_cycle)$, $w_{min} = 0.85$, $w_{max} = 1.15$
  - $\mathbf{v_{i,j}}$ **(new)** = $\mathbf{v_{i,j}}$ **(old)*w(cycle) + C1*(**$\mathbf{p_{i,j} – x_i}$**)*R1**$_{\mathbf{cycle, i, j}}$ **+ C2*(**$\mathbf{g_j – x_i}$**)*R2**$_{\mathbf{cycle, i, j}}$
  - if $\mathbf{v_{i,j}} > v_j^{max}$, then $\mathbf{v_{i,j}} = v_j^{max}$.

    if $\mathbf{v_{i,j}} < -v_j^{max}$, then $\mathbf{v_{i,j}} = -v_j^{max}$
  - $\mathbf{x_i}$ (new) = $\mathbf{x_i}$ (old) + $\mathbf{v_i}$
  - if $\mathbf{x_{i,j}} > x_j^{max}$, then $\mathbf{x_{i,j}} = x_j^{max}$.

    if $\mathbf{x_{i,j}} < x_j^{min}$, then $\mathbf{x_{i,j}} = x_j^{min}$
  - update $\mathbf{p_i} \in i$ and update **g**
- ✓ after iteration in times of Max_cycle
  - **g** is the optimal solution we can get from this metaheuristic process.

The polynomial regression curve is in the format of ($a_o + a_1*x^1 + \ldots\ldots + a_{(Dim-1)}*x^{(Dim-1)}$). We call this polynomial regression curve R(x) and use the loss function to evaluate the R(x). In this problem, we use following loss function, Loss(R) = $\sum | y - R(x) |$ for all given points in the .data file. When the smaller value Loss (R) is, the better regression curve we get.

In this problem, you need to implement a C/C++ program to find the optimal solution of the polynomial regression curve using the metaheuristic mentioned above. The result of **g** in metaheuristic will represent the optimal solution of coefficients in polynomial regression curve ($a_o, a_1, \ldots\ldots, a_{(Dim-1)}$) as well.

- ✓ Input file
  - .set file: give the following value in each column. Dim(int), Num_b(int), Max_cycle(int), C1(double), C2(double), Max_boundary(Dim integers), Min_boundary(Dim integers).
  - .rand1 file: there are Max_cycle sections and a line contains one "=" between two part. Each part has Num_b columns and Dim double numbers in one line.
  - .rand2 file: there are Max_cycle sections and a line contains one "=" between two part. Each part has Num_b columns and Dim double numbers in one line.
  - .data file: In the first line, there is an integer number (N). Furthermore, there are N lines and two numbers in each line. In each line, the first number is an integer number and it means the point's x value and the second number is a double number

and it means the point's y value.

- ✓ Output
  - .out: Please show the best result of (a_0, a_1, ……, $a_{(Dim-1)}$) on the terminal and only one double number in one line.
- ✓ Execution command
  - **./hard-2 [.set file name] [.date file name] [.rand1 file name] [.rand2 file name]**
- ✓ Note: please use integer and double. Otherwise, there may be some error.

```
>vim 1.set
```

```
Dim 3
Num_b 7
Max_cycle 195
C1 1.0
C2 1.4
Min_boundary -26 -16 -8
Max_boundary 26 18 6
```

```
>vim 1.data
```

```
13
12 228.772
-20 721.726
25 1099.03
-22 848.368
18 532.991
23 825.856
-1 11.8333
2 9.08295
-14 389.738
-16 491.535
22 904.856
21 723.609
24 1091.15
```

```
>./p2 1.set 1.data 1.rand1 1.rand2
```

```
20.9489        //a0
3.48253        //a1
1.67649       //a2
```

- ✓ **File name rule:**
  - cpp file name: `p2.cpp`
  - executable file name: `p2`

PROBLEM 03

In homework 2, we have learned to use a two-dimensional array to implement a simple maze problem. But in many cases, two-dimensional arrays cannot make good use of memory space and express the relevance of data, so we learned linked list at the end of the semester. To be a good programmer, you must learn to use a suitable container. Hope you can have a deep understanding through this question.

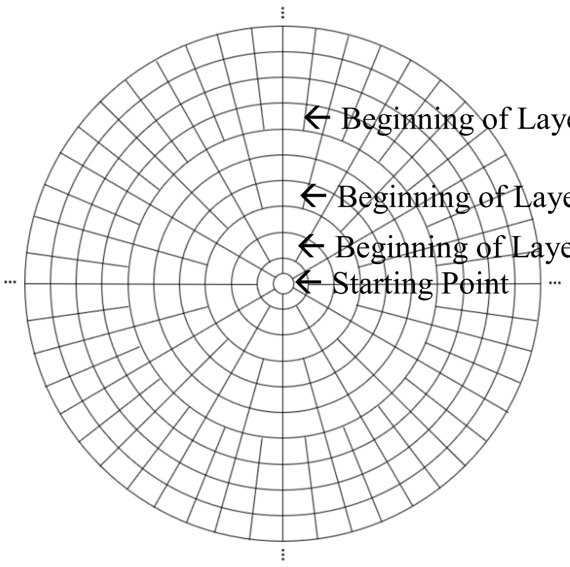
This problem is to solve the least cost path solution of the circular maze.

The structure of the maze is shown in the figure below. The maze is a structure of concentric circles. Starting from the center circle, the first circle is divided into 6 blocks, the 2nd to 3rd circles are divided into 12 blocks, and the 4th to 6th circles are divided into 24 blocks, and so on. (Note that the number of layers may not be exactly 1, 3, 6, 10, 15... there may be more)

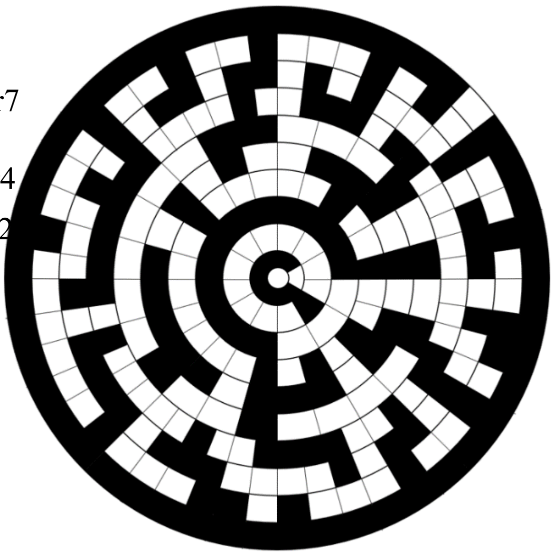
There is only one exit in the outermost layer of the maze, the rest will be walls. Except that, each block may be a wall or a road.

In the input file, the data will present the data of a layer of maze as a line, starting from the upper right block (marked in the figure), and displaying line by line from inside to outside. You need to read data from input file by using

```
./h04 <input>
```



(The structure of the maze)

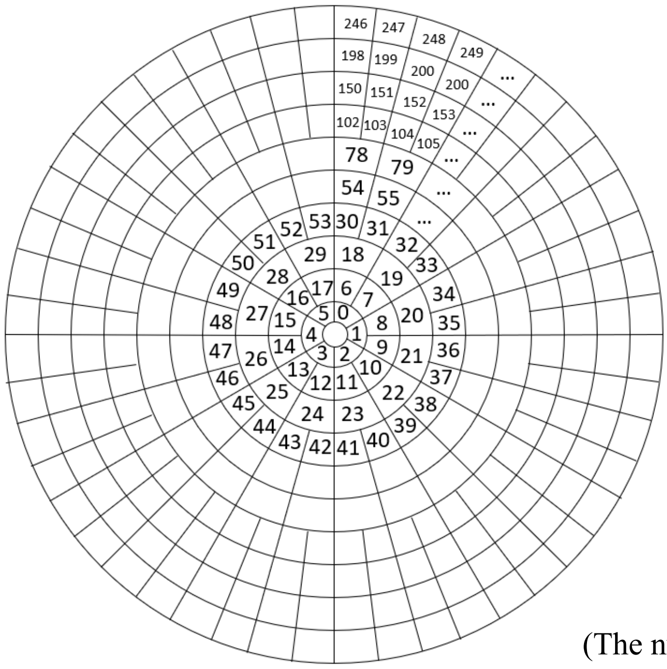


(The example maze)

From the example above, the input file might be

```
>cat example.in
0 1 0 0 0 0
1 1 1 1 0 1 1 1 1 1 1 1
0 0 0 1 1 1 0 0 0 0 0 0
1 1 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1
1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 1
1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0
1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1
1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 0 0
1 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

0 represents the wall and non-zero positive integer represents the cost of the road. You need to find a path from the starting point to the exit, and the sum of all costs on this path will have the minimum value. Finally, show the path on the terminal according to the following numbers, or print "fail" if there is no path to exit.



(The number of each blocks)

(Ex)

```
>./p3 example.in
1 -> 9 -> 21 -> 29 -> 39 -> … -> 252↲
>./p3 example2.in
fail↲
>
```

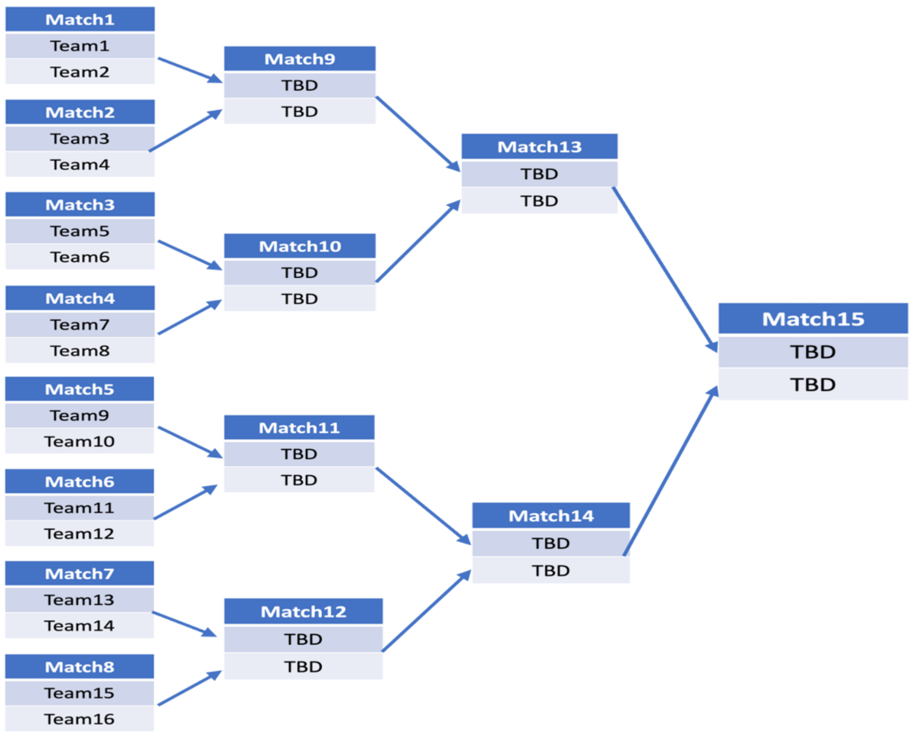
✓ **File name rule:**
- cpp file name: `p3.cpp`
- executable file name: `p3 [input_filename]`

PROBLEM 04

✓ In this exercise, you need to write a C/C++ program to read the input file which shows a part of information of the world cup's matches and write some specify information to the output file.

✓ The world cup has 32 teams. The match schedule has two parts.

1. The first part is the group matches. These 32 teams will be divided into 8 groups. Every team need to have one match with each of the rest of the teams in the same group. And then, pick two team which get the first two high points as the final 16.
   - The winner of each match gets 3 points. The loser of each match gets 0 points. If the match is a draw, both of the team get 1 point.
   - If two team has the same point, the priority of promotion will be determined by the summation of scores of every matches.
2. The second part is the knockout stage. Only the winner of matches can go through. It's showed by the following figure. There is not the draw in the knockout stage. If the score is equal, it will do the PK battle. The order of match is match1 -> match2 -> …… - > match 15. (TBD is "to be decided" which means the team is not decided until the previous match is ended.)



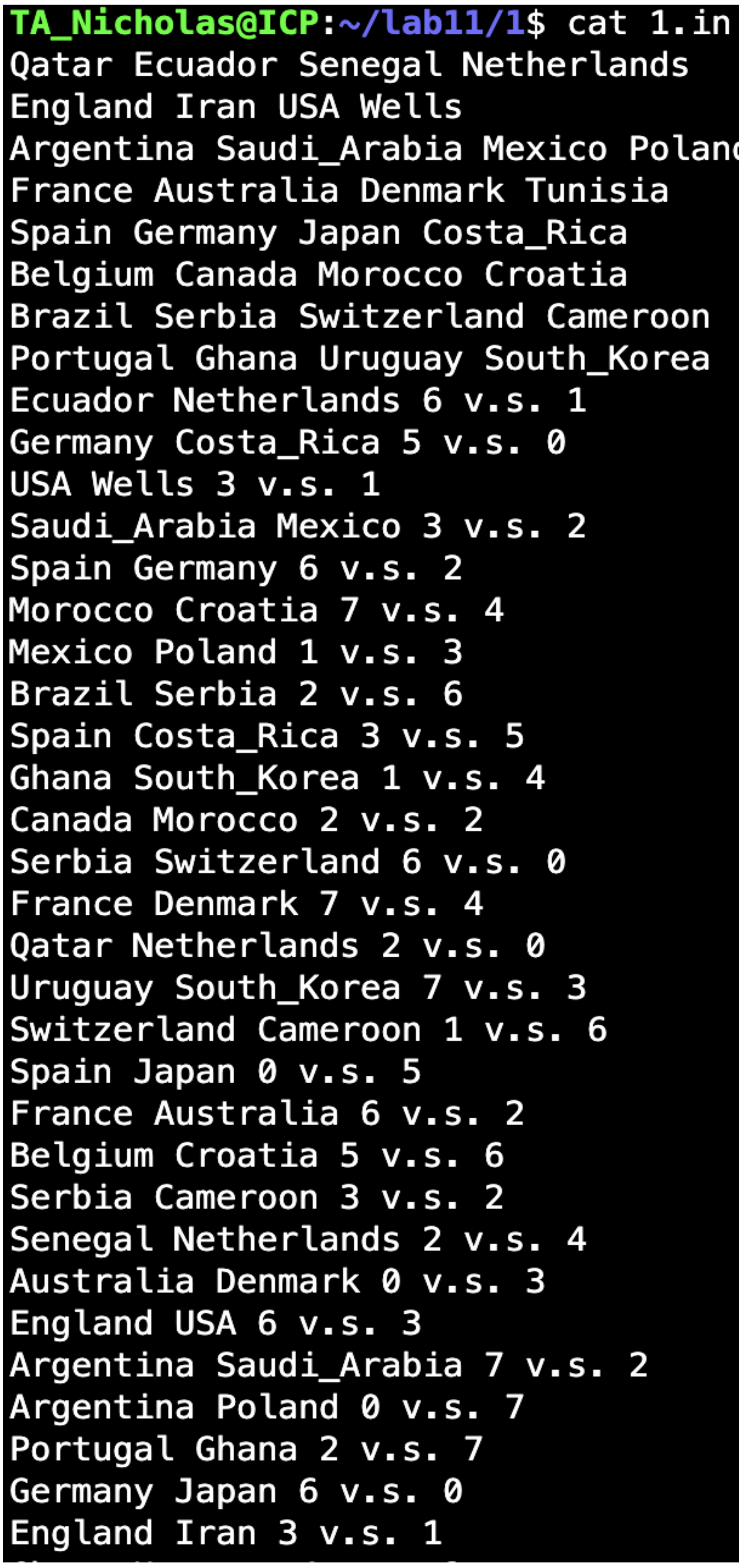
✓ In this program, you need to do

1. Read the input file which is entered by the users in command line. The input file contains 4 parts.
    - The first part contains 8 lines and there are four teams' name in each line. The first line means group 1 and so on.
    - The second part shows the result of every group matches. The format of each line is <Team 1> <Team 2> <Team 1's score> v.s. <Team 2's score>
    - The third part shows the result of matches in knockout stage. The first line of this part means the result of match 1 and so on. The format of each line is <Team 1's score> v.s. <Team 2's score>. However, if PK battle is held, the format is <Team 1'score>(<Team 1's PK score>) v.s. <Team 2'score>(<Team 2's PK score>). **The knockout stage may be incompleted.**
    - The end of the third part is followed by a line which contains a "=".
    - The fourth part means many cases of the drawing straw. One line means one cases and contains 16 numbers. The order of drawing straw is 1st in group 1 -> 1st in group2 -> …… -> 1st in group 8 -> 2nd in group 1 -> 2nd in group 2 -> …… -> 2nd in group8. The number means the serial number in the knockout stage. For example, if a team get number 8, this team is going to be the team 8 in the knockout stage.
2. Write the results to the output file.
    - The result of each case writes to the output file in order and separated by a line which contains a "=".
    - The result of each case should contain 15 lines which means the information of match1 to match15. Each line contains two parts and there is a **'\t'** between these two parts.
        - The first part's format is <Team 1> v.s. <Team 2>. If the team is not decided, you should use "TBD" as the team's name.
        - The second part is the result of this match. If this match is completed, please show "The winner is <winner's name>". Otherwise, please show "The game is coming soon...".

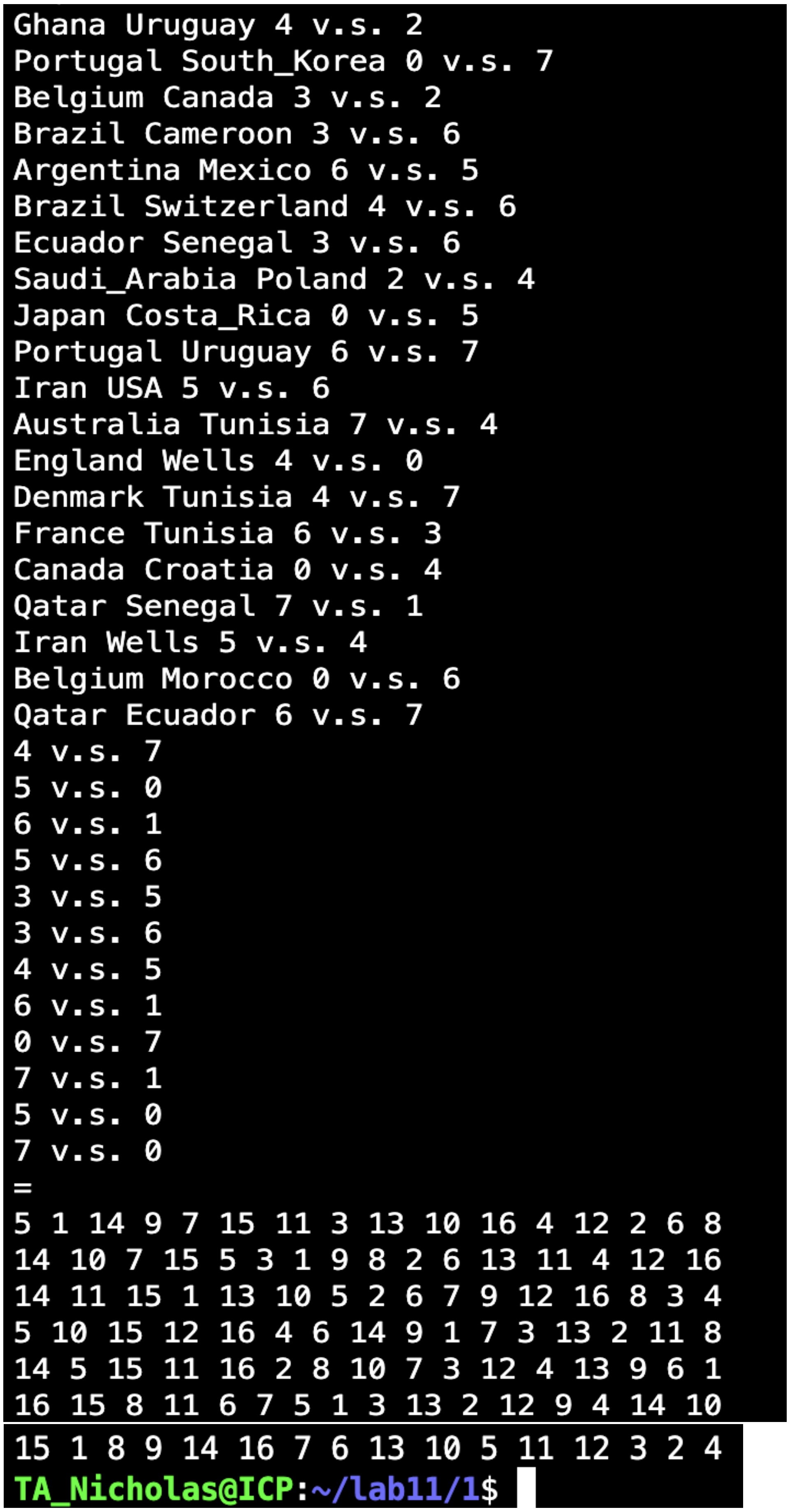
✓ File name rule:

- cpp file name: `p4.cpp`
- executable file name: `p4`
- execution command: `./p4 [input_filename] [output_filename]`

- ✓ For example
  - In the bellow picture, it is a picture that shows the input file(1.in).

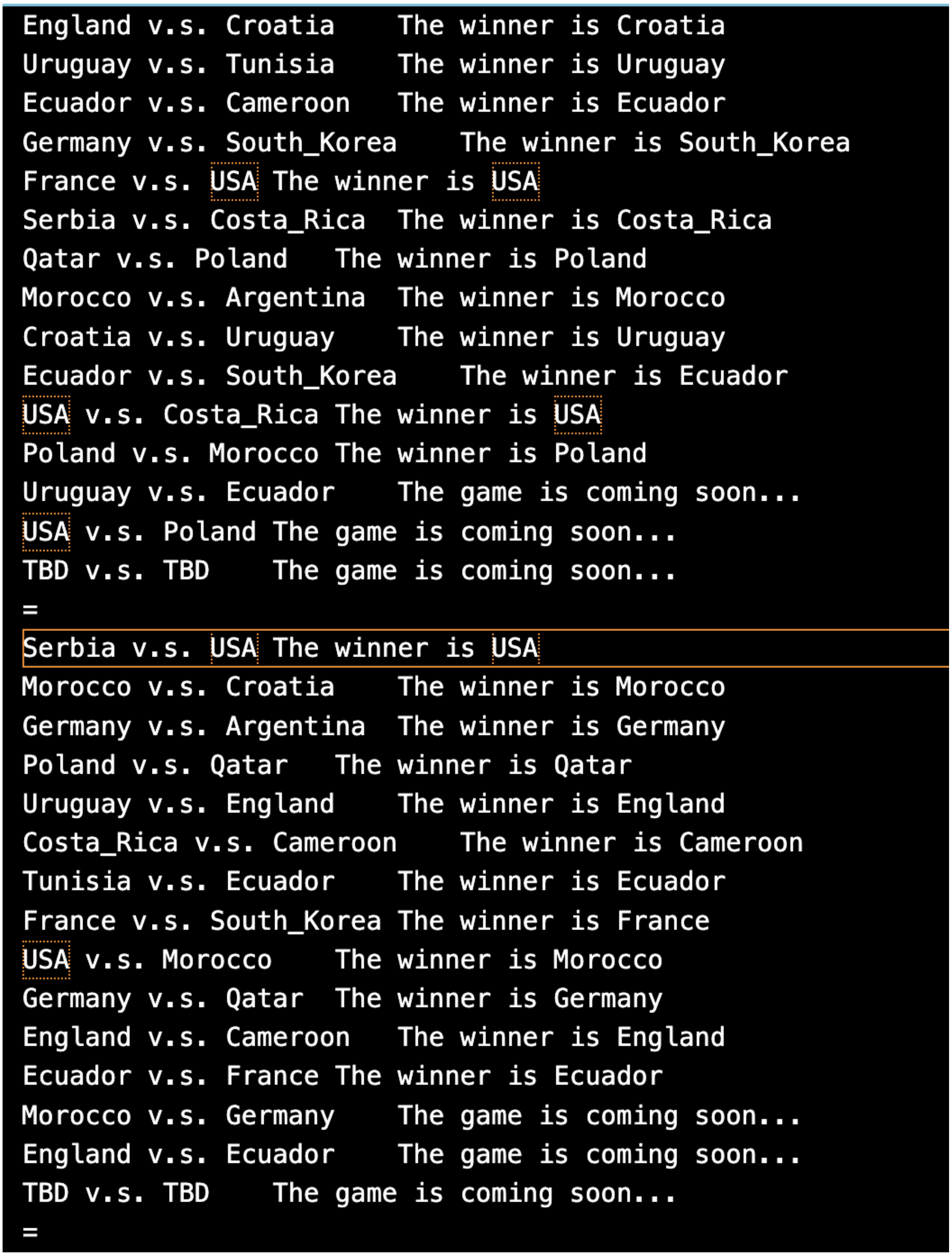


```
TA_Nicholas@ICP:~/lab11/1$ cat 1.in
Qatar Ecuador Senegal Netherlands
England Iran USA Wells
Argentina Saudi_Arabia Mexico Polan
France Australia Denmark Tunisia
Spain Germany Japan Costa_Rica
Belgium Canada Morocco Croatia
Brazil Serbia Switzerland Cameroon
Portugal Ghana Uruguay South_Korea
Ecuador Netherlands 6 v.s. 1
Germany Costa_Rica 5 v.s. 0
USA Wells 3 v.s. 1
Saudi_Arabia Mexico 3 v.s. 2
Spain Germany 6 v.s. 2
Morocco Croatia 7 v.s. 4
Mexico Poland 1 v.s. 3
Brazil Serbia 2 v.s. 6
Spain Costa_Rica 3 v.s. 5
Ghana South_Korea 1 v.s. 4
Canada Morocco 2 v.s. 2
Serbia Switzerland 6 v.s. 0
France Denmark 7 v.s. 4
Qatar Netherlands 2 v.s. 0
Uruguay South_Korea 7 v.s. 3
Switzerland Cameroon 1 v.s. 6
Spain Japan 0 v.s. 5
France Australia 6 v.s. 2
Belgium Croatia 5 v.s. 6
Serbia Cameroon 3 v.s. 2
Senegal Netherlands 2 v.s. 4
Australia Denmark 0 v.s. 3
England USA 6 v.s. 3
Argentina Saudi_Arabia 7 v.s. 2
Argentina Poland 0 v.s. 7
Portugal Ghana 2 v.s. 7
Germany Japan 6 v.s. 0
England Iran 3 v.s. 1
```



```
Ghana Uruguay 4 v.s. 2
Portugal South_Korea 0 v.s. 7
Belgium Canada 3 v.s. 2
Brazil Cameroon 3 v.s. 6
Argentina Mexico 6 v.s. 5
Brazil Switzerland 4 v.s. 6
Ecuador Senegal 3 v.s. 6
Saudi_Arabia Poland 2 v.s. 4
Japan Costa_Rica 0 v.s. 5
Portugal Uruguay 6 v.s. 7
Iran USA 5 v.s. 6
Australia Tunisia 7 v.s. 4
England Wells 4 v.s. 0
Denmark Tunisia 4 v.s. 7
France Tunisia 6 v.s. 3
Canada Croatia 0 v.s. 4
Qatar Senegal 7 v.s. 1
Iran Wells 5 v.s. 4
Belgium Morocco 0 v.s. 6
Qatar Ecuador 6 v.s. 7
4 v.s. 7
5 v.s. 0
6 v.s. 1
5 v.s. 6
3 v.s. 5
3 v.s. 6
4 v.s. 5
6 v.s. 1
0 v.s. 7
7 v.s. 1
5 v.s. 0
7 v.s. 0
=
5 1 14 9 7 15 11 3 13 10 16 4 12 2 6 8
14 10 7 15 5 3 1 9 8 2 6 13 11 4 12 16
14 11 15 1 13 10 5 2 6 7 9 12 16 8 3 4
5 10 15 12 16 4 6 14 9 1 7 3 13 2 11 8
14 5 15 11 16 2 8 10 7 3 12 4 13 9 6 1
16 15 8 11 6 7 5 1 3 13 2 12 9 4 14 10
15 1 8 9 14 16 7 6 13 10 5 11 12 3 2 4
TA_Nicholas@ICP:~/lab11/1$
```

- In the bellow picture, it is a picture that shows the part of output file after read the input file(1.in).



```
England v.s. Croatia    The winner is Croatia
Uruguay v.s. Tunisia    The winner is Uruguay
Ecuador v.s. Cameroon   The winner is Ecuador
Germany v.s. South_Korea    The winner is South_Korea
France v.s. USA The winner is USA
Serbia v.s. Costa_Rica  The winner is Costa_Rica
Qatar v.s. Poland   The winner is Poland
Morocco v.s. Argentina  The winner is Morocco
Croatia v.s. Uruguay    The winner is Uruguay
Ecuador v.s. South_Korea    The winner is Ecuador
USA v.s. Costa_Rica The winner is USA
Poland v.s. Morocco The winner is Poland
Uruguay v.s. Ecuador    The game is coming soon...
USA v.s. Poland The game is coming soon...
TBD v.s. TBD    The game is coming soon...
=
Serbia v.s. USA The winner is USA
Morocco v.s. Croatia    The winner is Morocco
Germany v.s. Argentina  The winner is Germany
Poland v.s. Qatar   The winner is Qatar
Uruguay v.s. England    The winner is England
Costa_Rica v.s. Cameroon    The winner is Cameroon
Tunisia v.s. Ecuador    The winner is Ecuador
France v.s. South_Korea The winner is France
USA v.s. Morocco    The winner is Morocco
Germany v.s. Qatar  The winner is Germany
England v.s. Cameroon   The winner is England
Ecuador v.s. France The winner is Ecuador
Morocco v.s. Germany    The game is coming soon...
England v.s. Ecuador    The game is coming soon...
TBD v.s. TBD    The game is coming soon...
=
```

PROBLEM 05

Considering the heavy load of every student in the ICP class, Prof. Charles and TAs decided to hold a Christmas Gift Exchange event during the final-exam week. Every participant will provide the following information (<name>, <whose gift I want to receive the most>, <my gift name>, <good gift or not>). However, the host, who is one of the participants, has the highest priority to meet his/herself requirements. Besides, friends of the host can also enjoy the welfare, and the priority depends on how close to the host he/she is. In brief, the number of participants, the participant information, the host name and the friends of the host (given by the descending order of how close to the host) are provided in the input file.

Depending on whether the received gift is good or not and the gift provider is his/her designated person or not, the gift receiver becomes happy or upset on different levels. Your goal is to create and maintain a 2D linked list to store participant information on different levels of mood points. As Fig. 2 shown, the mood_node storing the mood_point also served as the head of its corresponding student_nodes. Besides, the first mood_node is pointed by the list_node, and every mood_node is linked one by one in the ascending order of the mood_point (the last mood_node points to NULL). The student_nodes are also linked one by one in the ascending order of the student_name, whose data type is “string” (the last student_node points to NULL).

Before the gift exchange, every participant has a mood point of 0 initially. Please print the current list on the terminal for the first time. Since the participants haven’t received any gift, every name of the participant is followed by “(0)”, standing for “no gifts received yet”. For the gift exchange part 1, the host and his/her friend may receive the gift in order. However, it is possible for the host and his/her friend to not receive gifts if the gift is given away to someone else already. After the gift exchange part 1, please print the current list on the terminal for the second time. The received gift_name should be printed after the student name between parenthesis, e.g. Tina(Handmade_Tissue_Bomb). For the gift exchange part 2, the rest of the participants, including those who are not friends of the host and those who are friends of the host but his/her desired gift is taken away by someone else already in part 1. The details of the exchanging rules are in the following description. After the gift exchange part 2, please print the current list on the terminal for the third time.

Finally, after both gift exchange part 1 and part 2, please print all of the participants’ <name>, <received_gift_name> and <emoji> on the terminal (separated by a space “ “).

✔ Input File Description

- <student_num> will be given in the beginning.
- Then, there are <student_num> following lines to provide the participant information (<name>, <whose gift I want to receive the most>, <my gift name>, <good gift or not>).
- For example, "Rose Ted Apple_Magsafe 1", means the participant's name is "Rose", she would like to receive Ted's gift the most, the gift provided by Rose is "Apple_Magsafe" which is a good gift.
- For <good gift or not>, 1 stands for good gift while 0 stands for bad gift.
- The last line is the host and his/her friends, provided by <host_name>, <friend1_name>, <friend2_name>, …. (Noted: friend number is less than student number)

✔ Exchange Rules

- Exchange Priority: host > friend 1 > friend 2 > ……
- Part 1: according to the priority of the host and his/her friend, give the gifts to them.
- In part 1, if the designated gift is already given, the participant will not receive gifts in this part.
- Part 2: according to the input order, give the gift to those who haven't received gifts, and the gifts are also distributed in input file order (the given gifts will be skipped).
- Please note that the participant must not receive the gift provided by him/herself.

✔ Mood Points Rules

- Initially, every participant has a mood point of 0.
- If the participant receive the gift from whom he/she would like to receive,
  - His/Her mood point is changed from 0 to 2, if the gift is a good gift.
  - His/Her mood point is changed from 0 to -2, if the gift is a bad gift.
- If the participant "does not" receive the gift from whom he/she would like to receive,
  - His/Her mood point is changed from 0 to 1, if the gift is a good gift.
  - His/Her mood point is changed from 0 to -1, if the gift is a bad gift.

✔ Correlation of the emojis and the mood points

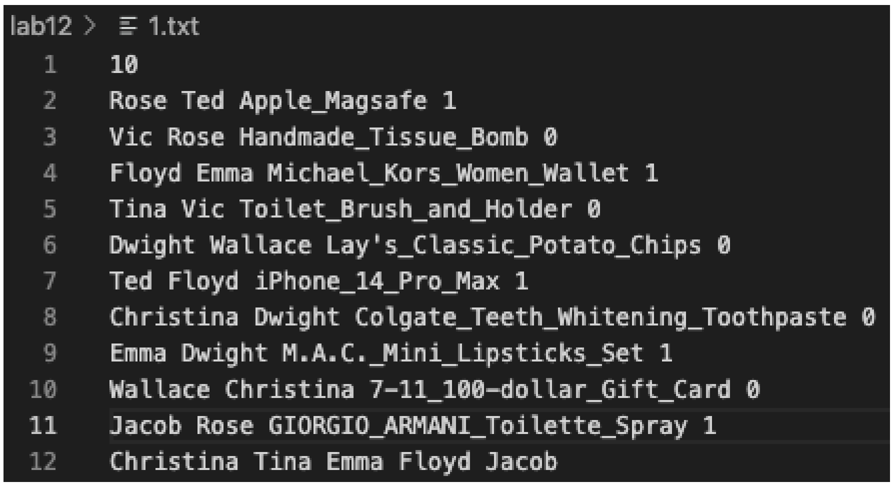
- When the mood point is 2, the corresponding emoji is ":)))".
- When the mood point is 1, the corresponding emoji is ":)".
- When the mood point is -1, the corresponding emoji is ":(".
- When the mood point is -2, the corresponding emoji is ":(((".

✔ The following images are an example of input file, structure of the 2D linked list and the example output corresponding to the 1.txt input file.

✔ You can get the test case from /home/share/lab12/ex12-1/.

✔ Please note that

- you can only get the score by implementing a 2D linked list according to the structure in Fig. 2.
- applying STL containers is not allowed.



lab12 > 1.txt

```
10
Rose Ted Apple_Magsafe 1
Vic Rose Handmade_Tissue_Bomb 0
Floyd Emma Michael_Kors_Women_Wallet 1
Tina Vic Toilet_Brush_and_Holder 0
Dwight Wallace Lay's_Classic_Potato_Chips 0
Ted Floyd iPhone_14_Pro_Max 1
Christina Dwight Colgate_Teeth_Whitening_Toothpaste 0
Emma Dwight M.A.C._Mini_Lipsticks_Set 1
Wallace Christina 7-11_100-dollar_Gift_Card 0
Jacob Rose GIORGIO_ARMANI_Toilette_Spray 1
Christina Tina Emma Floyd Jacob
```

Figure 1: example input file - 1.txt

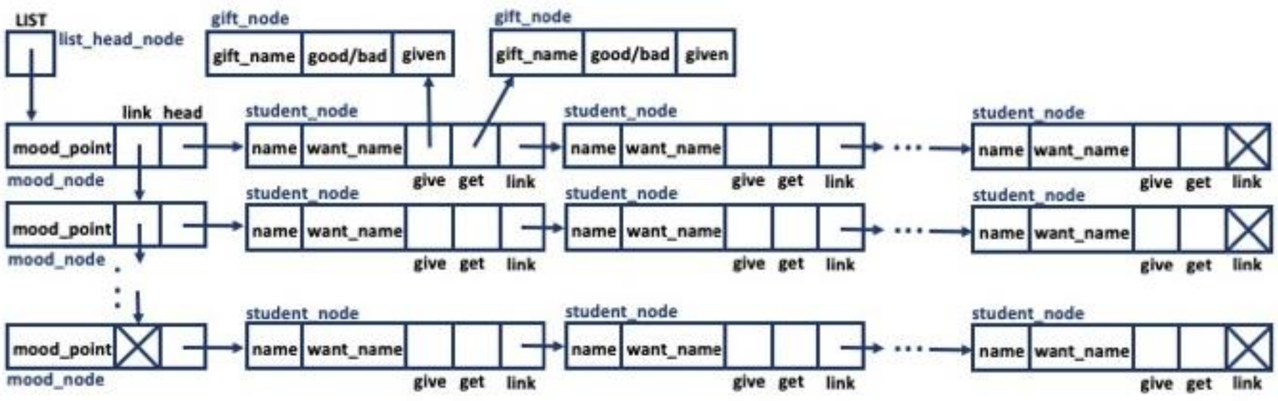
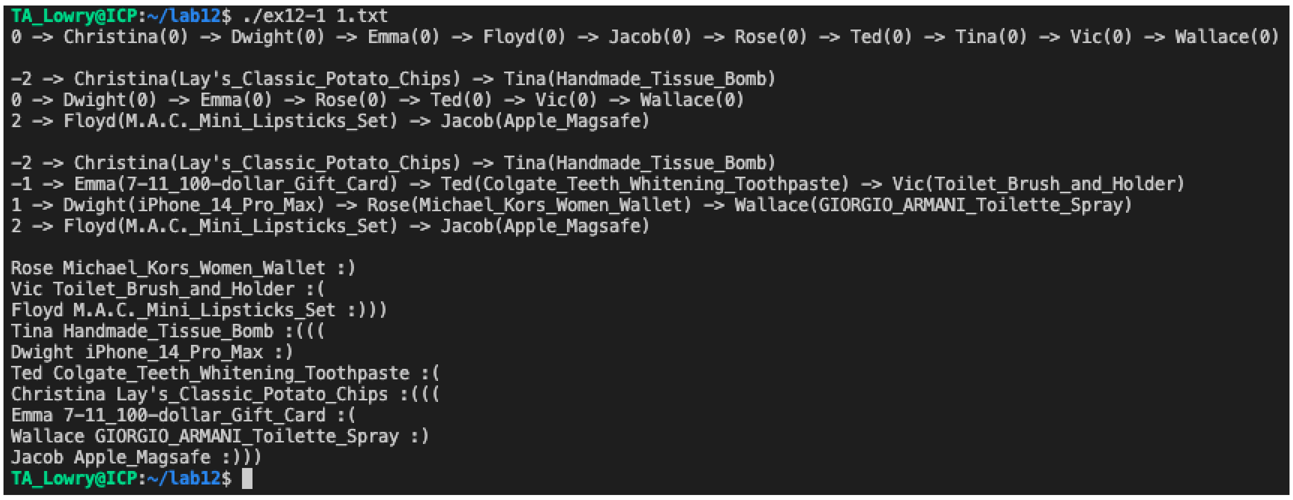


Figure 2: structure of the 2D linked list



```
TA_Lowry@ICP:~/lab12$ ./ex12-1 1.txt
0 -> Christina(0) -> Dwight(0) -> Emma(0) -> Floyd(0) -> Jacob(0) -> Rose(0) -> Ted(0) -> Tina(0) -> Vic(0) -> Wallace(0)

-2 -> Christina(Lay's_Classic_Potato_Chips) -> Tina(Handmade_Tissue_Bomb)
0 -> Dwight(0) -> Emma(0) -> Rose(0) -> Ted(0) -> Vic(0) -> Wallace(0)
2 -> Floyd(M.A.C._Mini_Lipsticks_Set) -> Jacob(Apple_Magsafe)

-2 -> Christina(Lay's_Classic_Potato_Chips) -> Tina(Handmade_Tissue_Bomb)
-1 -> Emma(7-11_100-dollar_Gift_Card) -> Ted(Colgate_Teeth_Whitening_Toothpaste) -> Vic(Toilet_Brush_and_Holder)
1 -> Dwight(iPhone_14_Pro_Max) -> Rose(Michael_Kors_Women_Wallet) -> Wallace(GIORGIO_ARMANI_Toilette_Spray)
2 -> Floyd(M.A.C._Mini_Lipsticks_Set) -> Jacob(Apple_Magsafe)

Rose Michael_Kors_Women_Wallet :)
Vic Toilet_Brush_and_Holder :(
Floyd M.A.C._Mini_Lipsticks_Set :)))
Tina Handmade_Tissue_Bomb :(((
Dwight iPhone_14_Pro_Max :)
Ted Colgate_Teeth_Whitening_Toothpaste :(
Christina Lay's_Classic_Potato_Chips :(((
Emma 7-11_100-dollar_Gift_Card :(
Wallace GIORGIO_ARMANI_Toilette_Spray :)
Jacob Apple_Magsafe :)))
TA_Lowry@ICP:~/lab12$
```

Figure 3: corresponding example output

✔ File name rule:

- ➢ cpp file name: `p5.cpp`
- ➢ executable file name: `p5`