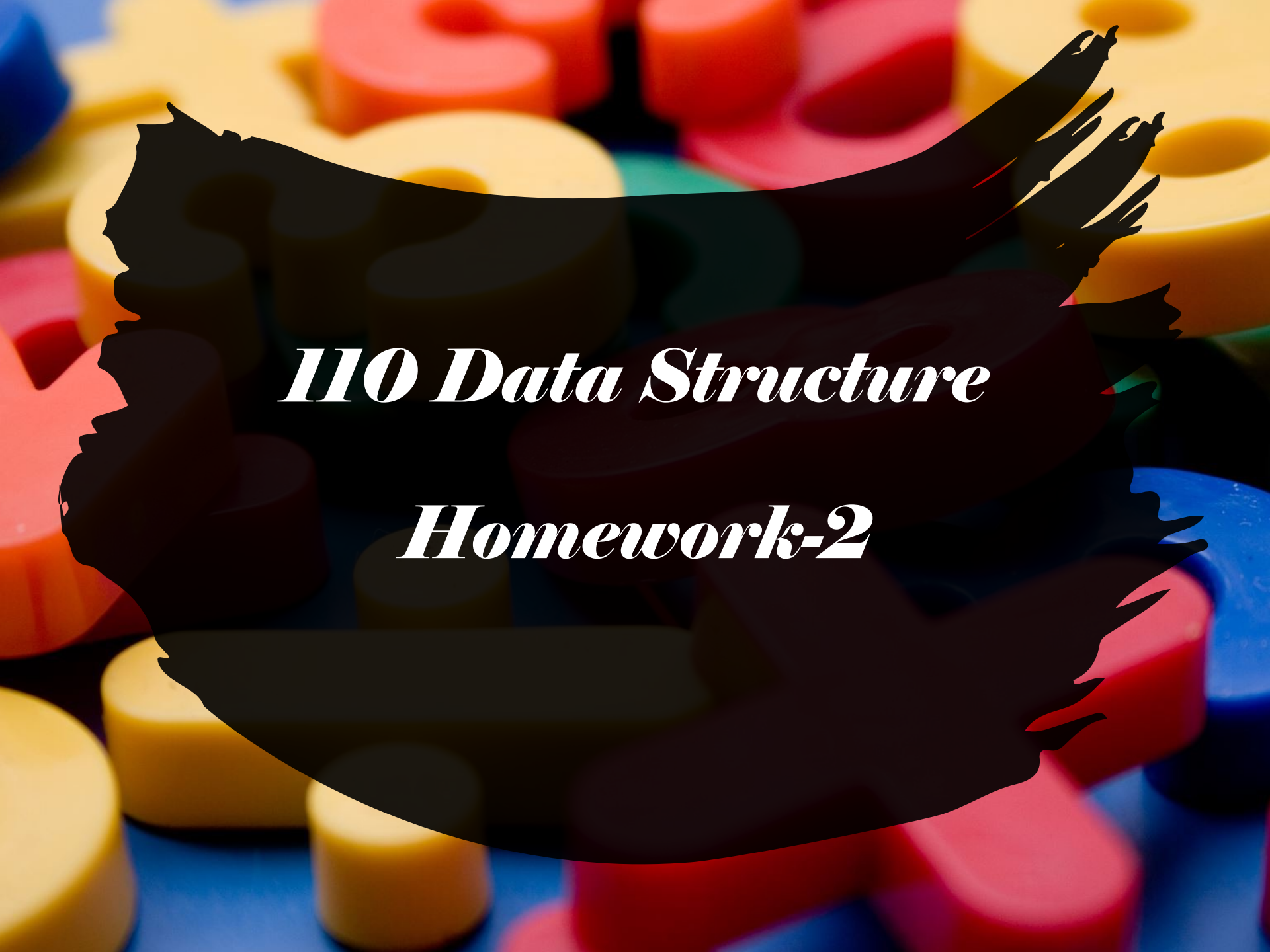

110 Data Structure
Homework-2

Homework rules

- Only accept C
- Filename : [student ID]_[hw2]-[question number]
- E.g. 4110012345_hw2-1.c
- Zip all your files and hand in on the iLearning
- Deadline 2021/11/16 23:59
- Please add comments in your code
- If any question, you can contact TA.
- nchuds110@gmail.com
- Do not copy! 0 points for plagiarism!

Question 1. (35%)

Please use the following structure to implement a singly linked list.

```
typedef struct listNode *listPointer;
typedef struct listNode {
        int data;
        listPointer link;
}
```

The singly linked list must have the following functions：

1. **add i :** add the new node whose data field is i to the end of the list. (5%)
2. **del i :** delete the *i*-th node. (5%)
3. **ins i j :** insert the new node whose data field is j after the *i*-th node. (5%)
4. **mul i j** : the data field of the *i*-th to last node multiplies by j. (5%)
5. **rev k :** treat K data as a group and reverse them. (15%)
6. **show :** print out the data in the singly linked list.

Example

- **Initial list :** empty
- add 5
- add 7
- add 9
- show
- 5 7 9

Example

- **Initial list :** 5 7 9 7 9
- del 2
- show
- 5 9 7 9
- **Initial list :** 5 7 9 7 9
- del 0 / del 6 …
- show
- 5 7 9 7 9
- (No change!)

Example

- **Initial list:** 1 2 3 4 5 7 8 9

  (index: 1 2 3 4 5 6 7 8)

- ins 5 6
- show
- 1 2 3 4 5 6 7 8 9

Example

- **Initial list:** 1 2 3 4 5 6 7 8 9
- mul 3 2
- show
- 1 2 3 4 5 6 14 8 9

Example

- **Initial list:** 1 2 3 4 5 6 7 8
- rev 3
- show
- 3 2 1 6 5 4 8 7
- (最後一組數量雖不足3，但視為一組反轉)

Input & output

- Read the file (input_1.txt)
- The **first line** presents the **initial list**.
- The **second line** contains an integer **n** indicates how many commands there are.
- The next **n** rows are the **commands**.
- Implement the functions in next slice, and write the result of “show” to the file (output_1.txt).

Example

(Input)

- 1 2 3 4 5 6
- 5
- add 9
- add 8
- rev 3
- del 1
- show

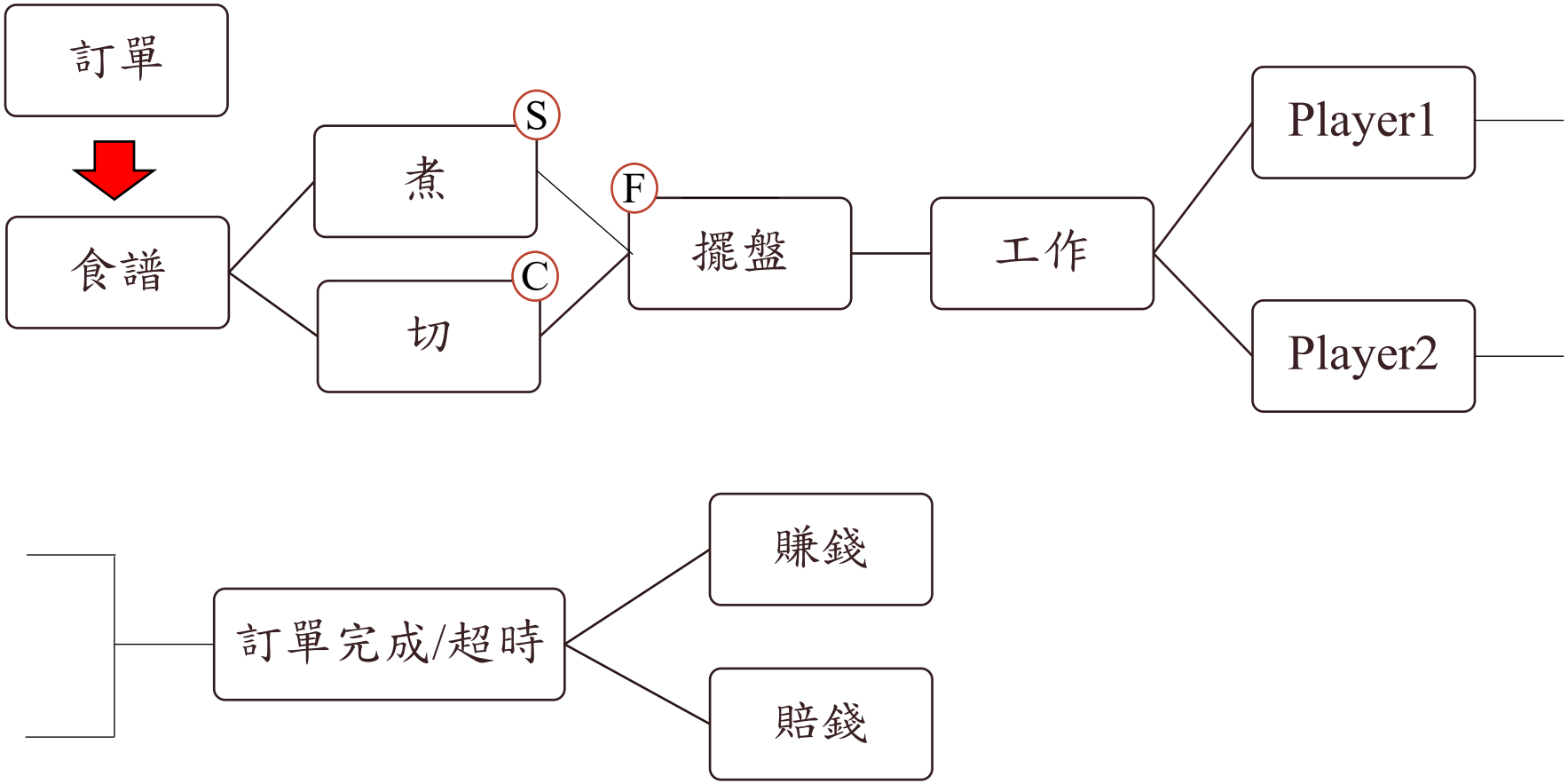
(Output)

- 2 1 6 5 4 8 9

Question 2. (65%)

- **Overcooked** is a well-known multiplayer cooperative game. Now you and your friends decided to play this game.
- **Today, our goal is to make as much money as possible.**
- In this scene, there is no need to wash the dishes.
- There is a stove and a cutting table and **only one** meat/vegetables/fruits available on the stove/cutting table **at the same time**. (*Please note this rule*)
- After all the ingredients are ready, the plate can be arranged. Then deliver meals to the guests and collect money from the guests.

Flow chart



Recipe

- **Read the file** (recipes.txt) to let your program know how to cook.
- For each dish, all the ingredients must be prepared and then arranging/decorating the food on the plate.
- It takes time to cook meat, cut vegetables and fruits, and food presentation(擺盤).
- **Cooking time**：

  (1) 5 minutes for cooking something on the stove (fixed)

  (2) 3 minutes for cutting something (fixed)

  (3) The different recipe has the different time for food presentation : (1+other)

Recipe

- [**recipe_name**] [**s**] [**c**] [**others**]
- **s :** The ingredients that need to be cooked first on the stove.
- **c :** The ingredients that need to be processed first on the cutting board.
- **others :** The ingredients that do not require pre-processing.
- The process of preparing each ingredient does not affect each other, **only one ingredient** can be **processed at a time.**
- **Food presentation** must be done at one time and **cannot be interrupted.**

Recipe

- [recipe_name] [s] [c] [others]

Example:

擺盤時間：5*1 + 3*2 + (1+1)

(1) hamburger beef lettuce,tomato bread

(If there are more than one kind of food ingredient , separate them with ",".)

擺盤時間：5*0 + 3*3 + (1+0)

(2) fruit_platter x peach,orange,grape x

(Use "x" to indicate that there is no such kind of ingredient.)

Input

- Read the file (orders.txt)
- In the **first line**, an integer **n** indicates how many orders there are.
- The next **n** rows are the **order data**.
- [order_ID] [recipe_name] [arrival] [deadline] [money] [punishment]

(1) The "**order_ID**" is used to distinguish the meals of different guests.

(The range of order_ID is 0000~9999)

Input

(2) The "**recipe_name**" field indicates which meals your guests ordered.

(3) The "**arrival**" field indicates the time when the order arrived.

(4) The "**deadline**" field indicates how long the meal can be waited for. ($arrival \leq deadline$)

(5) The "**money**" field indicates how much money you will earn from this order.

(6) The "**punishment**" field indicates how much you should pay for the guest if you didn't serve the meal before the deadline.

Input

- [**order_ID**] [**recipe_name**] [arrival] [deadline] [money] [**punishment**]
- Example:
- **1001** **hamburger** 30 100 50 -100
- **1002** **hamburger** 0 20 40 -80
- **1003** **fruit_platter** 5 85 150 -300

Output

- Output the work schedule of two players to players.txt in the order of time.
- The first line uses an integer to indicate how many commands there are.
- **One line contains only one schedule.**
- Then use the scoring program to calculate the money earned by each classmates.
- The scoring program will detect whether your sequence is legal. (whether the order completion time exceeds the deadline, orders added by yourself, etc.)

Output

- [player_ID] [t] [**order_ID**] [command] [object]
- [player_ID] indicates the player who executes this command.
- [t] indicates the time when the player start to do this command.
- **order_ID** indicates the order you are preparing for.
- Three kind of commands :

(1) s + object: cooking “object” on the stove.

(2) c + object: cutting “object” on cutting board.

(3) f : food presentation.

Output

In players.txt

- 12
- 1 0 1002 s beef
- 2 0 1002 c lettuce
- 2 3 1002 c tomato
- 1 6 1003 c peach
- 2 6 1002 f
- 2 9 1003 c grape
- 1 12 1003 c orange
- 1 15 1003 f
- 1 30 1001 s beef
- 2 30 1001 c lettuce
- 2 33 1001 c tomato
- 1 36 1001 f

Scoring method

- **Write a program to schedule the cooking process of two players.**
- If TA's algorithm earns X dollars.
- You will get **different scores** if the money you earn are Y dollars:

0%: you didn't earn any money. (Y ≤ 0 dollars)

1%~60%: Score = $\boldsymbol{Y/X * 60\%}$

65%: You earn more than TA's program and your schedule is legal. (Y > X)