

# 資料結構-HW1

Temple Run

# 目錄

1. 情境介紹
2. 題目說明
3. 範測分析
4. 解法說明
5. 執行結果



## 情境介紹

### 影片連結

在神廟中做左轉,右轉,跳躍,滑行動作,  
收集金幣獲取高分

# 題目說明

6 種指令 in Temple Run

1. Turn Left (TL)
2. Turn Right (TR)
3. Keep Left (KL)
4. Keep Right (KR)
5. JUMP
6. FLIP

## 題目說明

- 神廟的交叉路口沒有直行的方向,因此遇到交叉路口只能左轉(TL)跟右轉(TR).
- 神廟中會遇到大量的寶藏跟障礙物,需要利用KL、KR、JUMP跟FLIP來收集寶藏跟躲避障礙物.
  - 如果需要FLIP躲避障礙物時,卻使用JUMP,則JUMP後會有MISS TAG.
  - 如果需要JUMP躲避障礙物時,正確使用JUMP,則JUMP後不會出現MISS TAG.
- 當完成神廟的探索,需要計算收集到的寶藏&回顧返回神廟的路徑.

# 題目說明

寶藏有兩種類別-金幣(GOLD)跟銀幣(SILVER),

寶藏的價值事由獲得的順序決定,每獲得三個寶藏會形成一個group,

只有形成group才能計算價值,沒有形成group的寶藏不計算價值.

每個group之中的gold數量決定該group的價值,此gold在group內順序不影響價值.

1. GOLD GOLD GOLD: 500 points. (3個gold價值500分)
2. GOLD GOLD SILVER: 300 points. (2個gold價值300分)
3. GOLD SILVER SILVER: 150 points. (1個gold價值150分)
4. SILVER SILVER SILVER: 50 points. (0個gold價值50分)

## 題目說明

MISS TAG會影響成功搜集到的寶藏價值,  
每當MISS TAG產生,就無法成功收集接下來四個寶藏  
但這四個寶藏仍會另外記錄在總共蒐集到的寶藏

# 題目說明

Input:

1. There are  $n$  inputs in total, consisting of a series of TL, TR, KL, KR, JUMP, FLIP, MISS, GOLD, and SILVER, separated by '\n'.
2. The total number of operations is guaranteed to be  $0 < n \leq 100$ .
3. There must have at least 1 TR or TL

Output:

1. The route back to the temple from the city, separated by  $\rightarrow$ , followed by a line break.
2. The total value of treasures successfully collected, followed by a line break.
3. The total value of treasures had there been no mistake (MISS), followed by a line break.

Restriction: C++ Standard Library like vector, stack, queue are not allowed.



# 範測分析

## Example Input:

TL  
TR  
GOLD  
TL  
KR  
SILVER  
JUMP  
MISS  
GOLD  
GOLD  
TL  
GOLD  
SILVER  
TR  
KL  
SILVER  
TL

## Example Output:

RL->TL->RL->RL->TL->RL  
150  
600

## 範測分析

Example Input:

TL

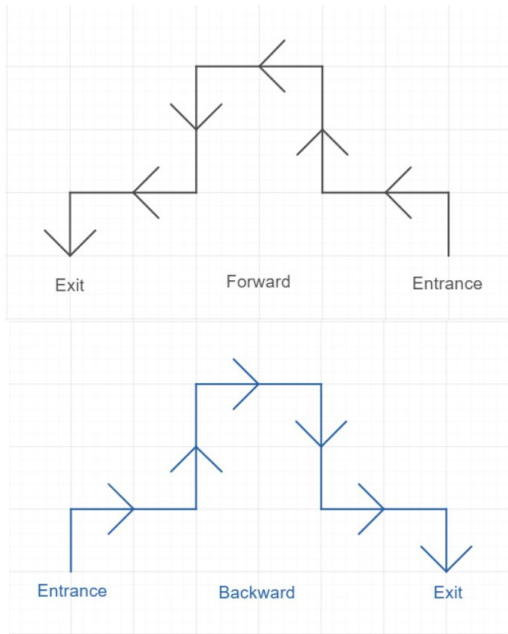
TR

TL

TL

TR

TL



Input給的是逃離的路徑

Output輸出要的是返回的路徑,

路徑只有跟TL & TR 指令與輸出有關.

Input:

TL->TR->TL->TL->TR->TL

## Output:

TR->TL->TR->TR->TL->TR

# 範測分析

Example Input:

GOLD <1>

SILVER <2>

JUMP <3>

MISS <4>

GOLD <5>

GOLD <6>

GOLD <7>

SILVER <8>

SILVER <9>

寶藏成功蒐集價值計算只有跟GOLD, SILVER, MISS指令有關.

<1> Group: GOLD

<2> Group: GOLD->SILVER

<3> Group: GOLD->SILVER

<4> Group: GOLD->SILVER

<5> Group: GOLD->SILVER (不能成功蒐集)

<6> Group: GOLD->SILVER (不能成功蒐集)

<7> Group: GOLD->SILVER (不能成功蒐集)

<8> Group: GOLD->SILVER (不能成功蒐集)

<9> Group: GOLD->SILVER->SILVER

Group: GOLD->SILVER->SILVER

->Group價值為150

(成功收集到的寶藏價值)

# 範測分析

Example Input:

GOLD <1>

SILVER <2>

JUMP <3>

MISS <4>

GOLD <5>

GOLD <6>

GOLD <7>

SILVER <8>

SILVER <9>

寶藏總價值計算只有跟GOLD, SILVER指令有關.

<1> Group1: GOLD

<2> Group1: GOLD->SILVER

<3> Group1: GOLD->SILVER

<4> Group1: GOLD->SILVER

<5> Group1: GOLD->SILVER->GOLD

<6> Group1: GOLD->SILVER->GOLD, Group2: GOLD

<7> Group1: GOLD->SILVER->GOLD, Group2: GOLD->GOLD

<8> Group1: GOLD->SILVER->GOLD, Group2: GOLD->GOLD->SILVER

<9> Group1: GOLD->SILVER->GOLD, Group2: GOLD->GOLD->SILVER, Group3: SILVER

Group1: GOLD->SILVER->SILVER

Group2: GOLD->GOLD->SILVER

Group3: SILVER

Total Group價值為 $150+150=300$

(寶藏總價值)

# 解法說明

- 核心資料結構:
  - stack: 利用FILO特性,將逃離路徑轉成返回路徑.
  - queue: 利用FIFO特性,將依序收集到的寶藏作價值統計.
- 由於題目要求禁止使用C++ Standad Libray,因此需要定義Class實作stack & queue.
  - 利用static array實作stack & queue.
  - 利用template讓stack & queue傳入數值不限於int.
  - stack: 實作DoubleCapacity, Push, Pop, Top, IsEmpty 函數.
  - queue: 實作DoubleCapacity, Push, Pop, IsEmpty, IsFull, getFront, getSize 函數.
  - 當stack或queue容量到達上限時再做Push,則會動態將容量加倍,避免超出array範圍導致segment fault.

# 解法說明

- main()架構:
  - 將Input指令cmd分成路徑指令與計算寶藏價值指令,個別儲存在對應在cmd\_map與cmd\_coin array之中.
    - 路徑相關指令: TL, TR
    - 寶藏價值指令: GOLD, SILVER, MISS
  - 呼叫以下三個函數個別計算並輸出結果:
    - compute\_route(): 輸出返回路徑
    - compute\_coin\_collected(): 輸出成功收集到的寶物總價值
    - compute\_coin\_total(): 輸出所有寶物總價值

# 解法說明

- `compute_route()` 架構:
  - 建立string array `route_result`, 儲存從城市返回路徑結果
  - 建立stack `route_city_to_temple`, 將從神廟到城市逃離路徑指令依序push到stack內.
  - 依序從stack取出路徑指令, 直到stack為清空:
    - 當取出逃離路徑指令為"TL", 代表返回路徑為"TR"
    - 當取出逃離路徑指令為"TR", 代表返回路徑為"TL"
    - 將返回路徑記錄在array `route_result`
  - 依序輸出array `route_result` 返回路徑結果

# 解法說明

- `compute_coin_collected()`架構:
  - 建立`queue coin_collect`,將寶藏價值指令依序push到`queue`內
  - 建立`int gold_count, silver_count, group_count, penalty_count, collect_value`,紀錄金幣數量,銀幣數量,群組(金幣+銀幣)數量,目前碰到障礙物而無法收集寶藏的懲罰計數,成功收集寶藏的價值
  - 依序從`queue`取出路徑指令,直到`queue`為清空:
    - 當取出的寶藏價值指令為"MISS",則設定`penalty_count = 4`
    - 當取出的寶藏價值指令為"GOLD"或"SILVER":
      - 若此時`penalty_count`不為0,代表目前受到懲罰,不能收集寶物,則`penalty_count - 1`.
      - 反之`penalty_count`為0,代表目前沒有受到懲罰,可以收集寶物
        - 如果收到的寶物是"GOLD",則`gold_count++`
        - 如果收到的寶物是"SILVER",則`silver_count++`



# 解法說明

- `compute_coin_collected()`架構:
  - 依序從queue取出路徑指令,直到queue為清空:
    - 計算group\_count數量 ( $\text{group\_count} = \text{gold\_count} + \text{silver\_count}$ )
      - 當group\_count=3,代表收集到的寶物數量形成一個group
        - 若gold\_count = 3,則collect\_value + 500
        - 若gold\_count = 2,則collect\_value + 300
        - 若gold\_count = 1,則collect\_value + 150
        - 若gold\_count = 0,則collect\_value + 50
  - 輸出成功收集到的寶物價值collect\_value

# 解法說明

- `compute_coin_total()`架構:
  - 建立queue `coin_toal`,將寶藏價值指令依序push到queue內
  - 建立int `gold_count`, `silver_count`, `group_count`,`total_value`,紀錄金幣數量,銀幣數量,群組(金幣+銀幣)數量,總共收集到的寶藏價值
  - 依序從queue取出路徑指令,直到queue為清空:
    - 當取出的寶藏價值指令為"GOLD"或"SILVER":
      - 如果收到的寶物是"GOLD",則`gold_count++`
      - 如果收到的寶物是"SILVER",則`silver_count++`

# 解法說明

- `compute_coin_total()`架構:
  - 依序從queue取出路徑指令,直到queue為清空:
    - 計算group\_count數量 ( $\text{group\_count} = \text{gold\_count} + \text{silver\_count}$ )
      - 當group\_count=3,代表收集到的寶物數量形成一個group
        - 若gold\_count = 3,則collect\_value + 500
        - 若gold\_count = 2,則collect\_value + 300
        - 若gold\_count = 1,則collect\_value + 150
        - 若gold\_count = 0,則collect\_value + 50
  - 輸出總共收集到的寶藏價值total\_value

# 執行結果


testdata.txt - 記事本  
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

TL  
TR  
GOLD  
TL  
KR  
SILVER  
JUMP  
MISS  
GOLD  
GOLD  
TL  
GOLD  
SILVER  
TR  
KL  
SILVER  
TL

```
Temple_Run.cpp X
test > Temple_Run.cpp > main()
268 }
269
270
271 int main() {
272
273     #ifndef OJ
274         freopen("testdata.txt", "r", stdin);
275     #endif // OJ
276
277     ios_base::sync_with_stdio(false);
278     cin.tie(0);
279     cout.tie(0);
280
281     string cmd;
282
283     while(cin >> cmd && !cin.eof()){
284         if(cmd == "TL" || cmd == "TR"){
285             cmd_map[cmd_map_idx++] = cmd;
286         }else if(cmd == "GOLD" || cmd == "SILVER" || cmd == "MISS"){
287             cmd_coin[cmd_coin_idx++] = cmd;
288         }
289     }
290
291     終端機 偵錯主控台 問題 輸出
292
293     [Running] cd "c:\Users\user\Desktop\test\" && g++ Temple_Run.cpp -o Temple_Run
294     TR->TL->TR->TR->TL->TR
295     150
296     600
297     |
298     [Done] exited with code=0 in 1.1 seconds
```

# 執行結果

[NTHU Online Judge](#) [Contest](#) [Problem](#) [Submit](#) [Status](#)

 [DS23\\_1102003S](#) [Logout](#)

## Status

Username

Problem ID

Contest ID

Status

---

▼

Search

SID	Submit Time	Username	Problem	Status	Source
3212076	July 31, 2023, 2:48 p.m.	<a href="#">DS23_1102003S</a>	<a href="#">13841 - Temple Run</a>	All Accepted (10/10)	C++17