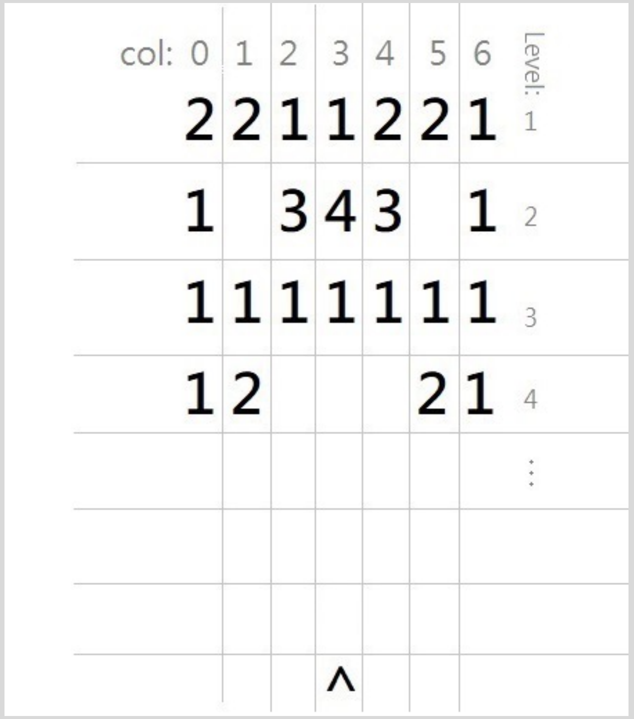
HW1 Stack Invader

Data Structures EECS2040

Brief

First impression

- Inspired by the classic arcade game **Space Invader**
- Simplified rules and the mechanics
- Can be solved with C/C++ basic, queues, stacks.



col: 0	1	2	3	4	5	6	Level:
2	2	1	1	2	2	1	1
1		3	4	3		1	2
1	1	1	1	1	1	1	3
1	2				2	1	4
							⋮
			^				

Goal

- Building wheels
  - Queue
  - Stack
- Implement the game core
  - Monitoring enemy positions
  - Player commands (firing bullets & querying enemy positions)

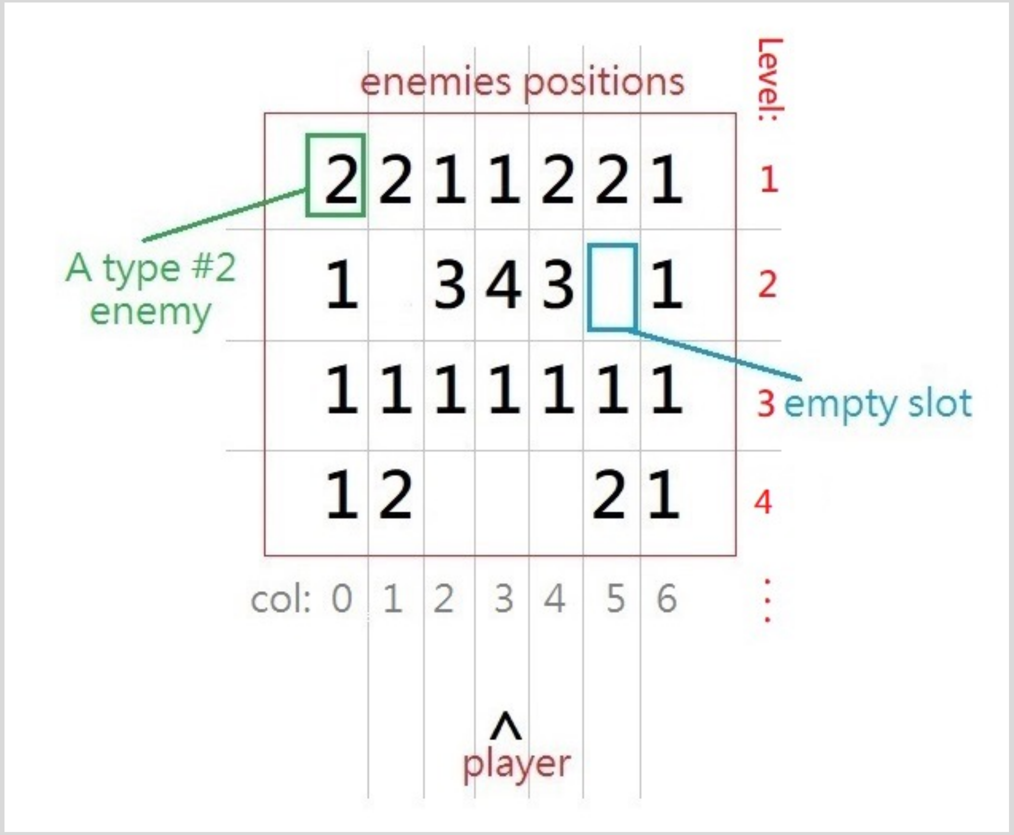
Details

Stage

- Enemies are placed in a grid.
- The numbers represent the enemy types. (1 to 5)

Enemy positions and types will be specified in the input

- Player fires from the bottom
- Columns index starts from 0
- Levels index starts from 1



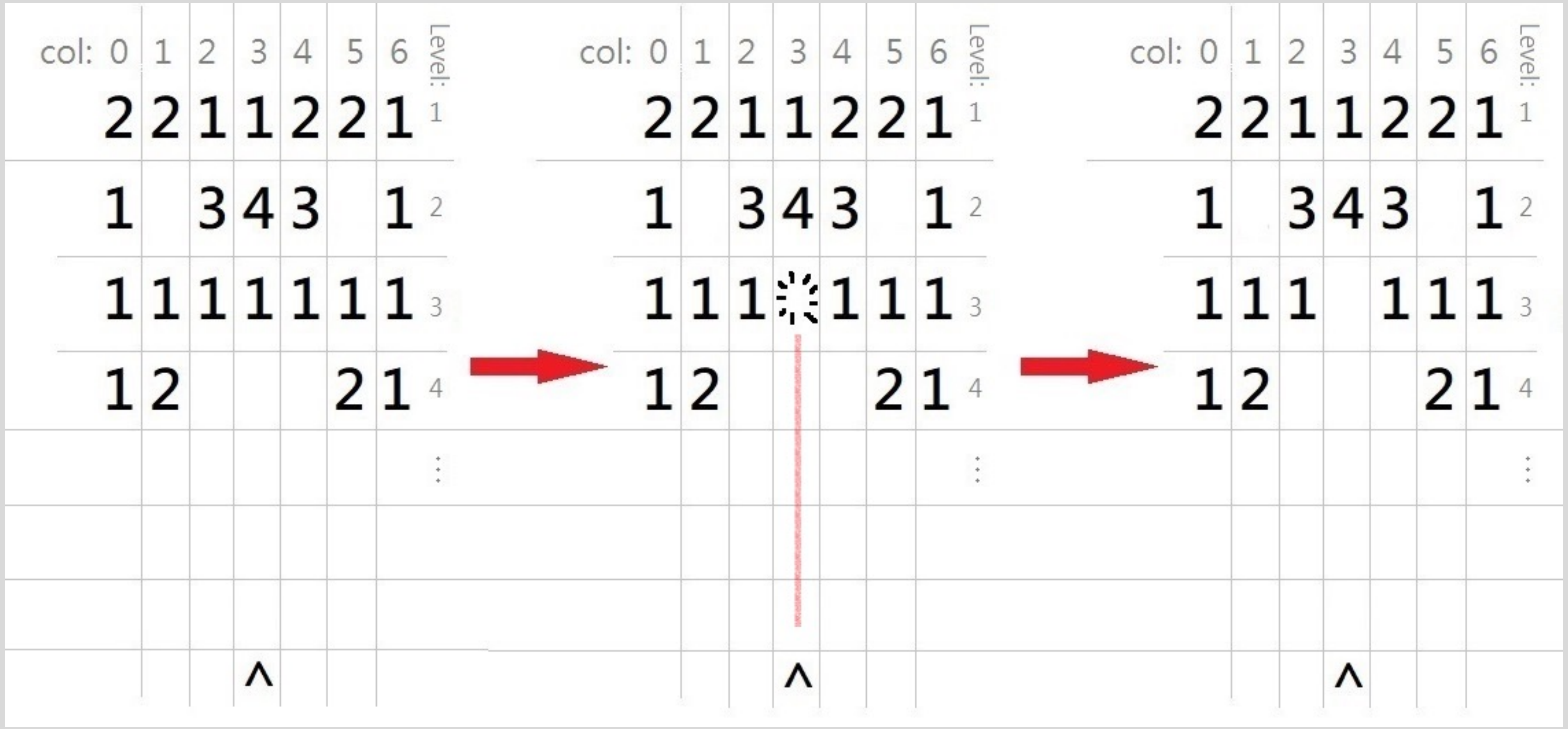
Bullet types

1. Normal Bullets
2. Shotgun Shells, noted as **<SG>**
3. Penetration bullets, noted as **<P>**
4. Super Bullets, noted as **<SB>**

These are referred as special bullets

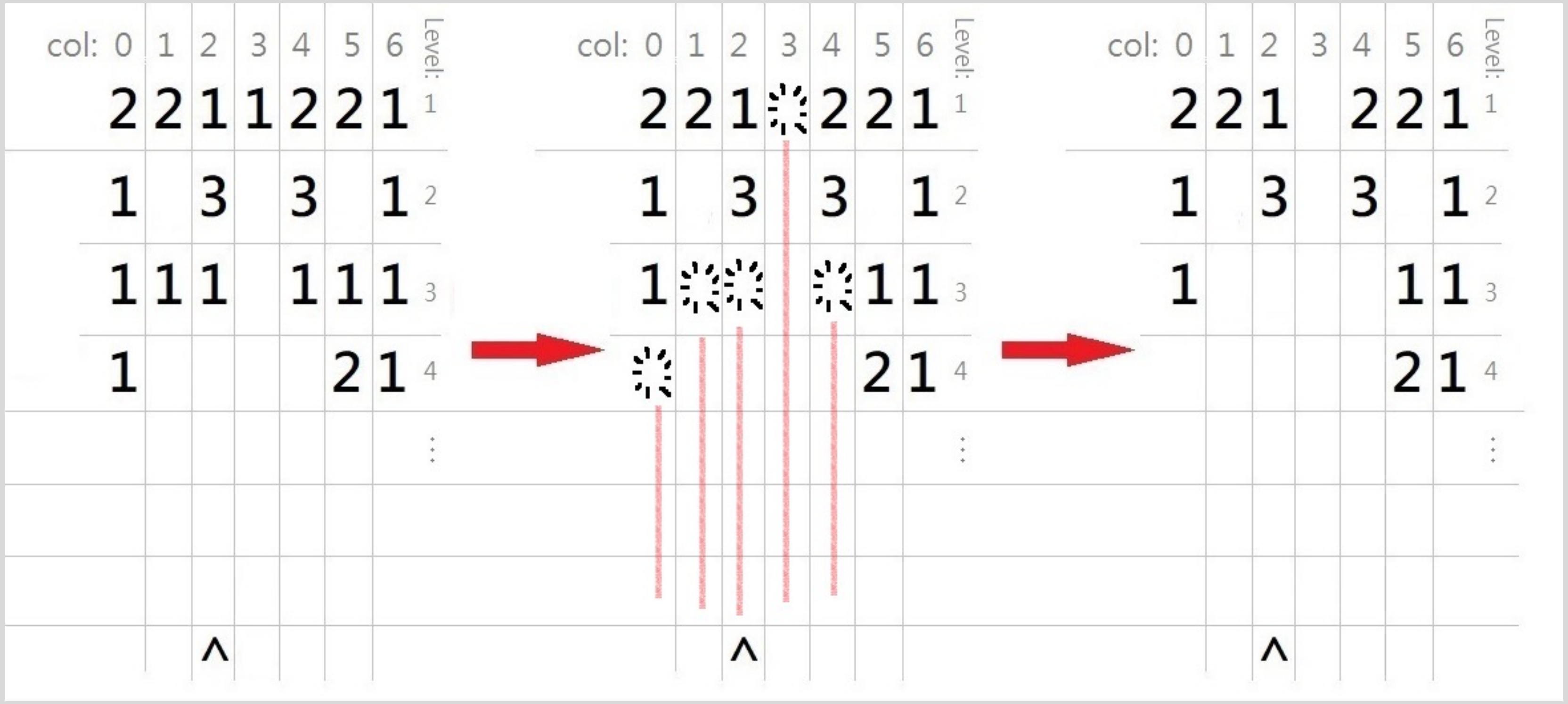
The special bullets need to earned and will be fired in the order as they are collected.

Bullet types - Normal bullets



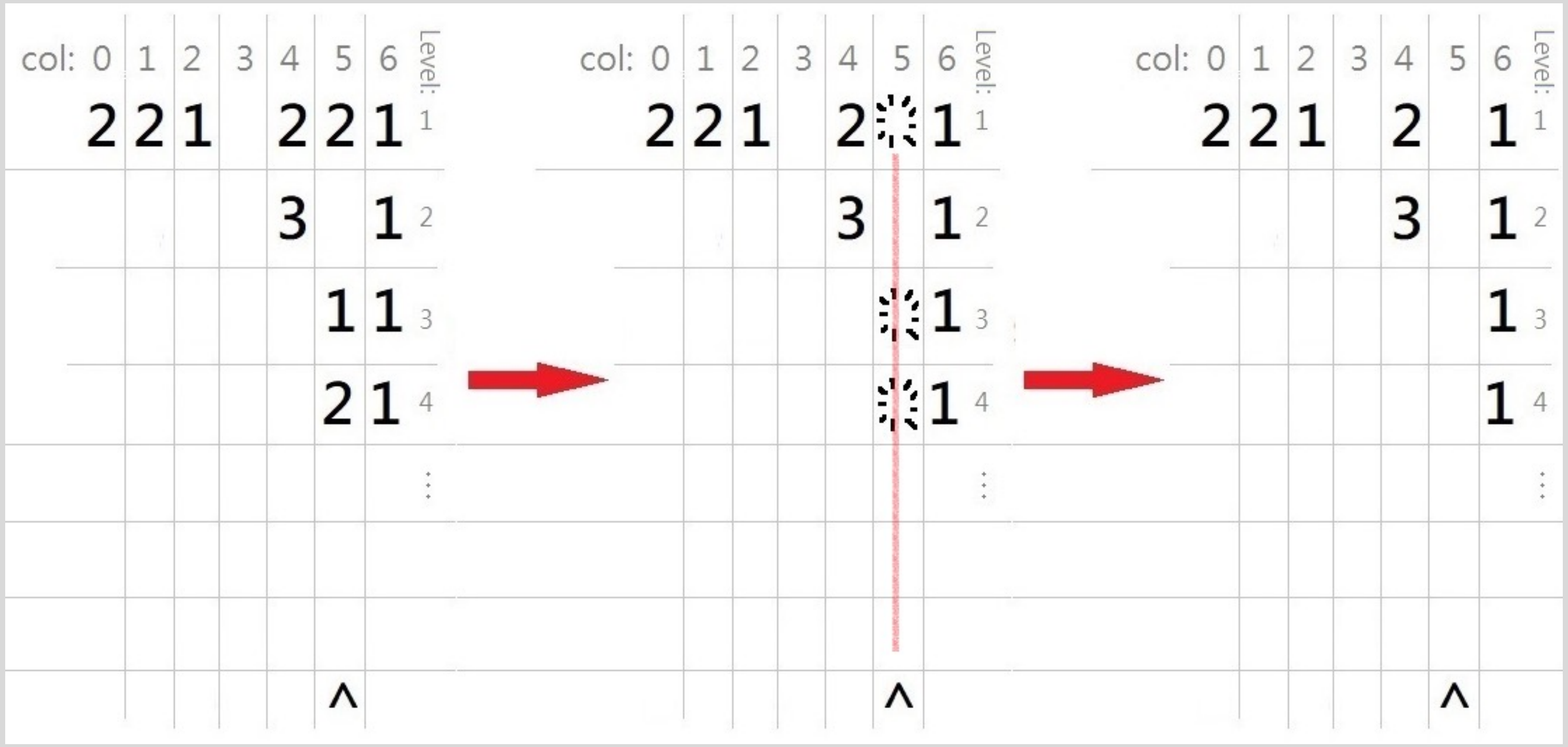
- Kills one enemy at a time
- No special effects.
- Infinite ammo

Bullet types - Shotgun Shells **<SG>**



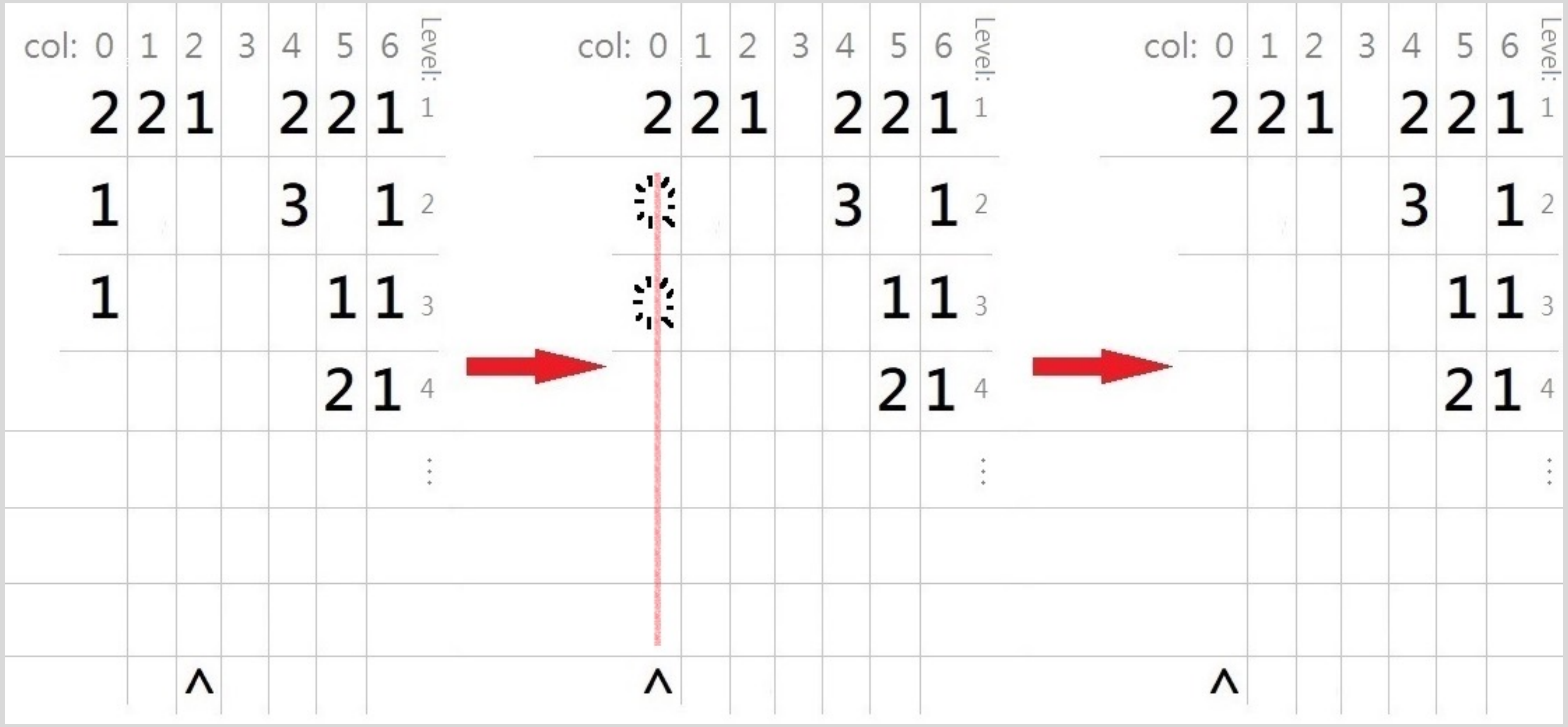
- Kills 5 enemies at a time
- Equals shooting 5 normal bullets to 5 different columns.
- (Aiming at col. ***X***, will end up hitting the col. ***X***-2 ~ ***X***+2)

Bullet types - Penetration bullets **<P>**



- Kills 3 enemies in the same column at a time.
- Equals shooting 3 normal bullets to 1 column.

Bullet types - Super bullets **<SB>**



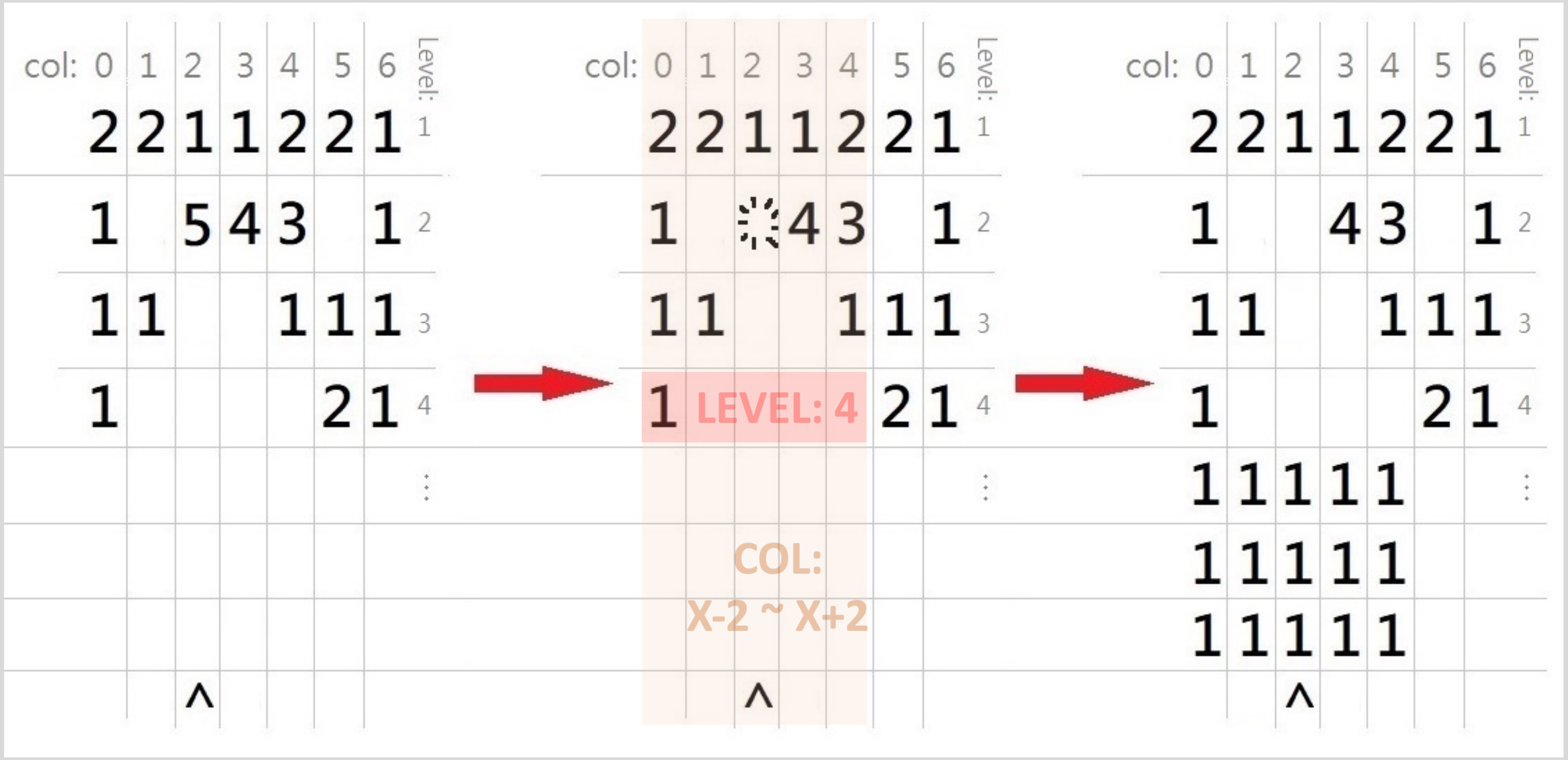
- Kills the same type of enemy in a column as many as possible…
- until it meets a different kind of enemy.

Enemies

- There are 5 types of enemies:
  - Enemy #1 : A Normal enemy. Nothing special.
  - Enemy #2 : If a player kills it, the player gets a shotgun shell **<SG>.**
  - Enemy #3 : If a player kills it, the player gets a penetration bullet **<P>.**
  - Enemy #4 : If a player kills it, the player gets a super bullet **<SB>.**

  - Enemy #5 : If a player kills it in col. ***X***, for each col. ***X***-2 ~ ***X***+2 will generate 3 new enemies If the max level that contains any enemy of those columns is ***L***, the enemies will be placed at level ***L***+1, ***L***+2, ***L***+3.

Example of Enemy #5

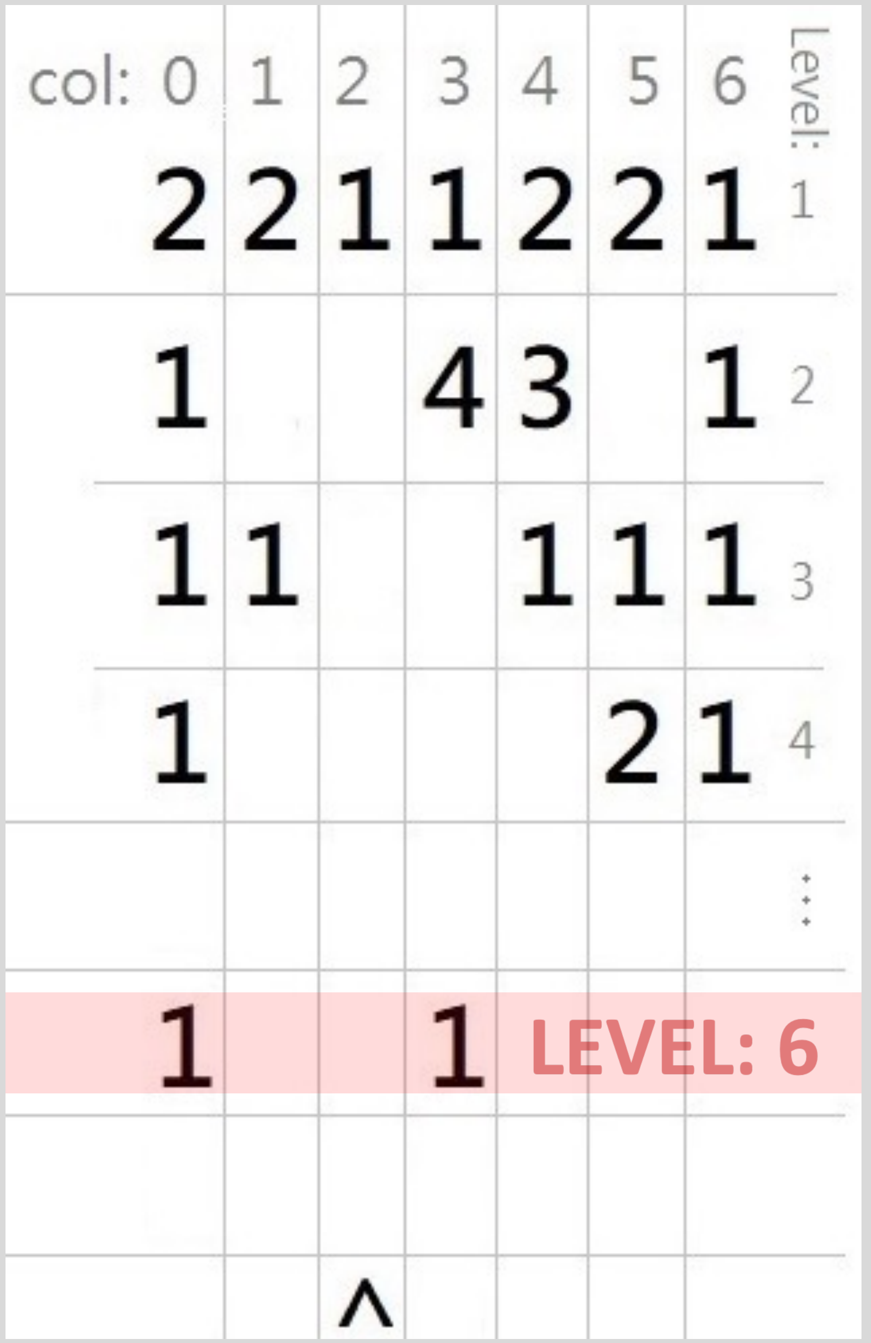


Player commands

- SHOOT <col> : Fire a normal bullet at a column.
  - E.g., "SHOOT 3" means firing a normal bullet at column 3.
- SPECIAL <col> : Fire a special bullet at a column.
  - E.g., "SPECIAL 3" means firing a special bullet at column 3.
- FRONT_ROW : Query the max level that contains any enemy and show the enemies on that level.

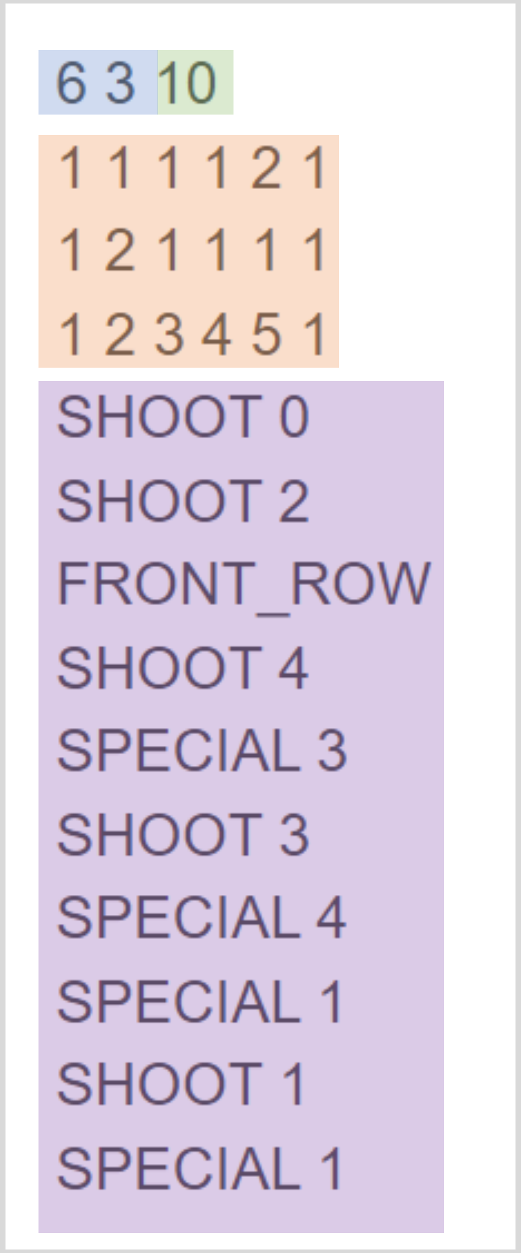
  - Output will be like:

```
FRONT_ROW, LEVEL: 6
1 _ _ 1 _ _ _
```



INPUT

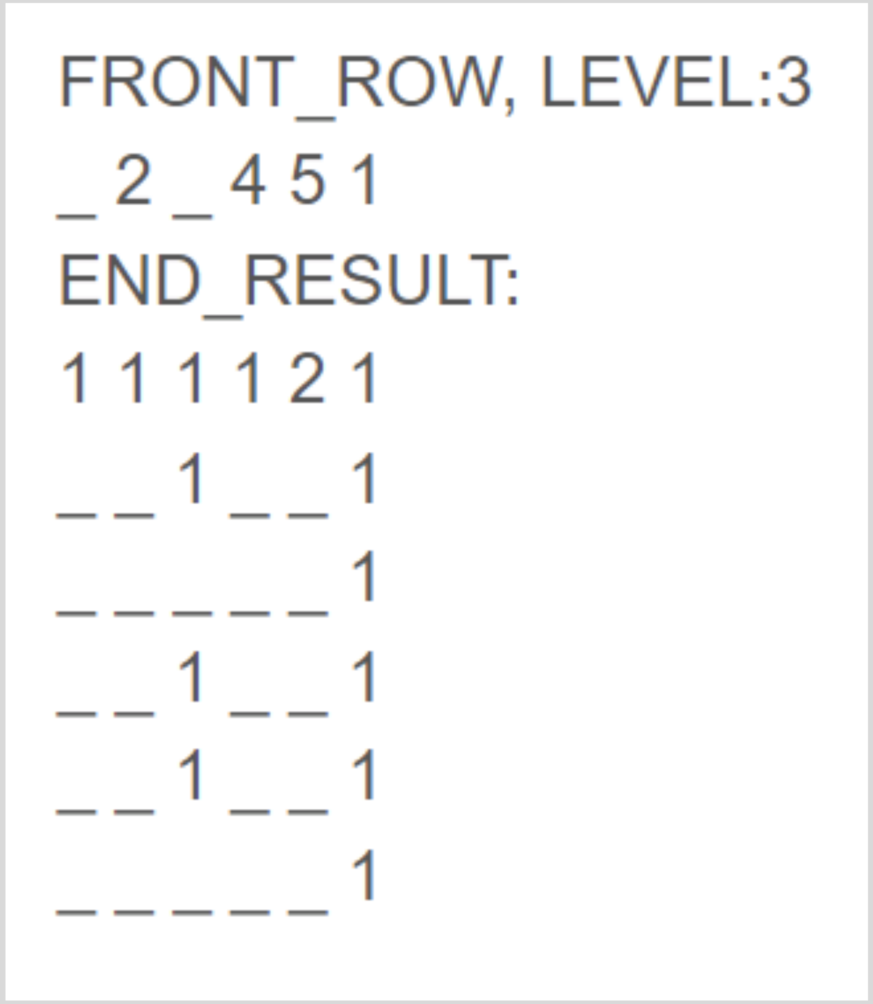
- Including…
  - The initial state parameter (# of columns and rows)
  - the number of commands.
  - Enemy types and positions.
  - Player commands.



```
6 3 10
1 1 1 1 2 1
1 2 1 1 1 1
1 2 3 4 5 1
SHOOT 0
SHOOT 2
FRONT_ROW
SHOOT 4
SPECIAL 3
SHOOT 3
SPECIAL 4
SPECIAL 1
SHOOT 1
SPECIAL 1
```

OUTPUT

- Print the front row information if player use the command “FRONT_ROW”
- After all the commands are finished, print the remaining enemies’ info.



```
FRONT_ROW, LEVEL:3
_ 2 _ 4 5 1
END_RESULT:
1 1 1 1 2 1
_ _ 1 _ _ 1
_ _ _ _ _ 1
_ _ 1 _ _ 1
_ _ 1 _ _ 1
_ _ _ _ _ 1
```

How to Start

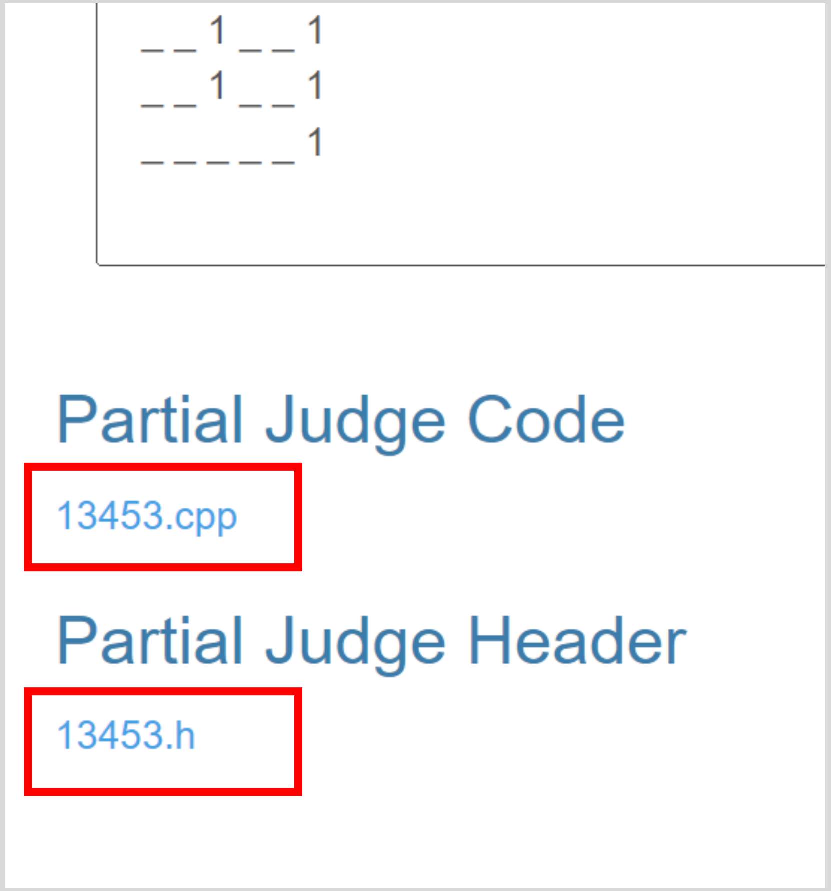
How to start (Using IDE Code::blocks)

- This is a partial judge problem. You need to compile two .cpp files and a header file together.

- Download the two files at the bottom of the problem page.

  https://acm.cs.nthu.edu.tw/contest/2491/#problem13453

- Rename the .h file to "function.h"



```
__1__1
__1__1
_____1
```

Partial Judge Code

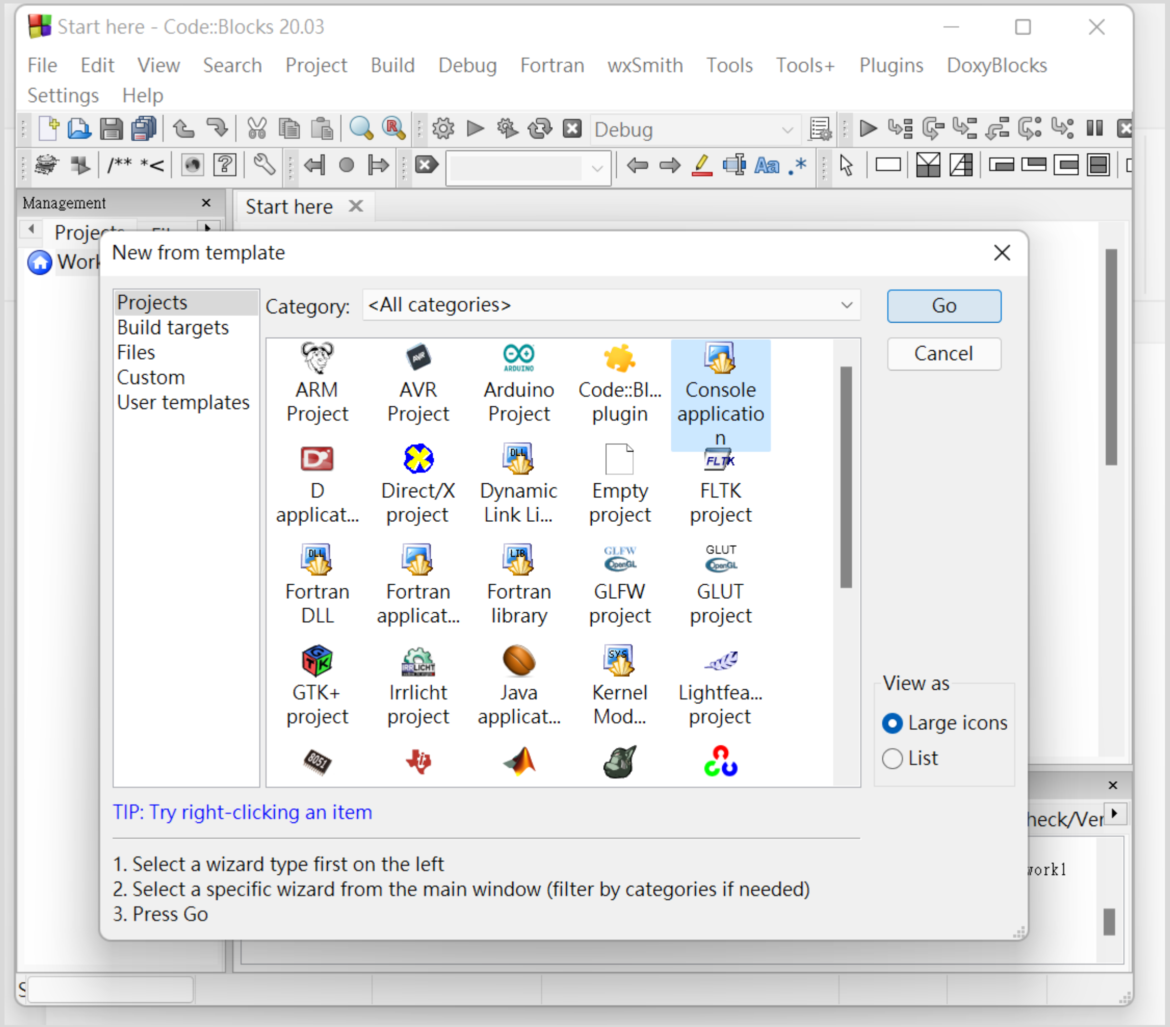
13453.cpp

Partial Judge Header

13453.h

How to start (Using IDE Code::blocks)

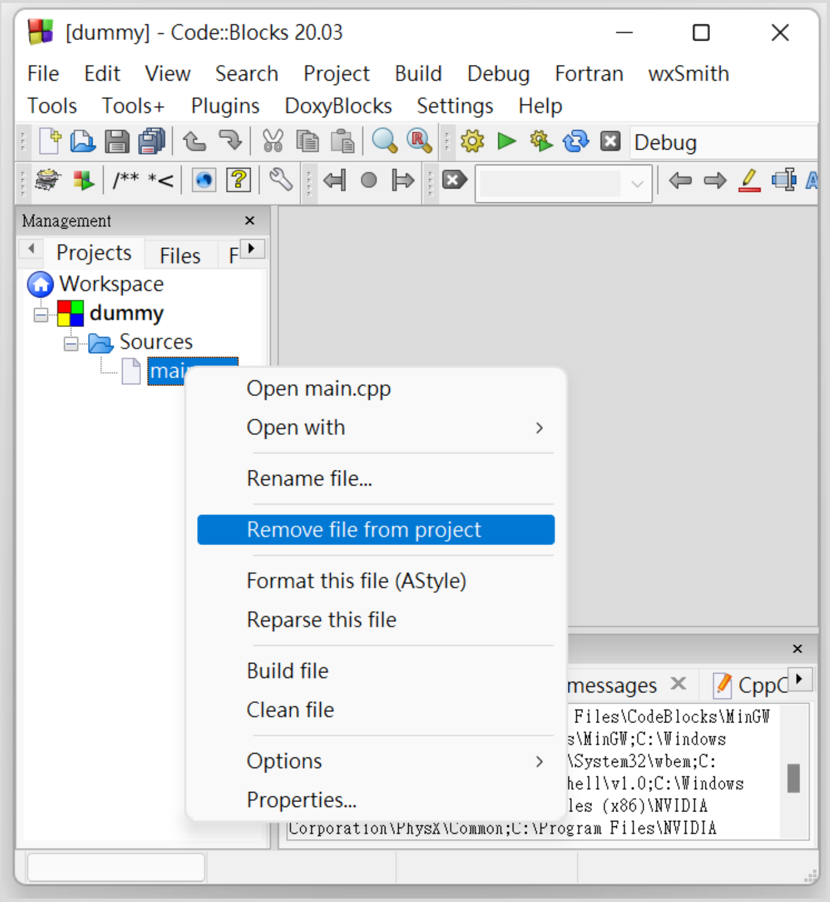
- Create a new console application project in code::blocks.



1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

How to start (Using IDE Code::blocks)

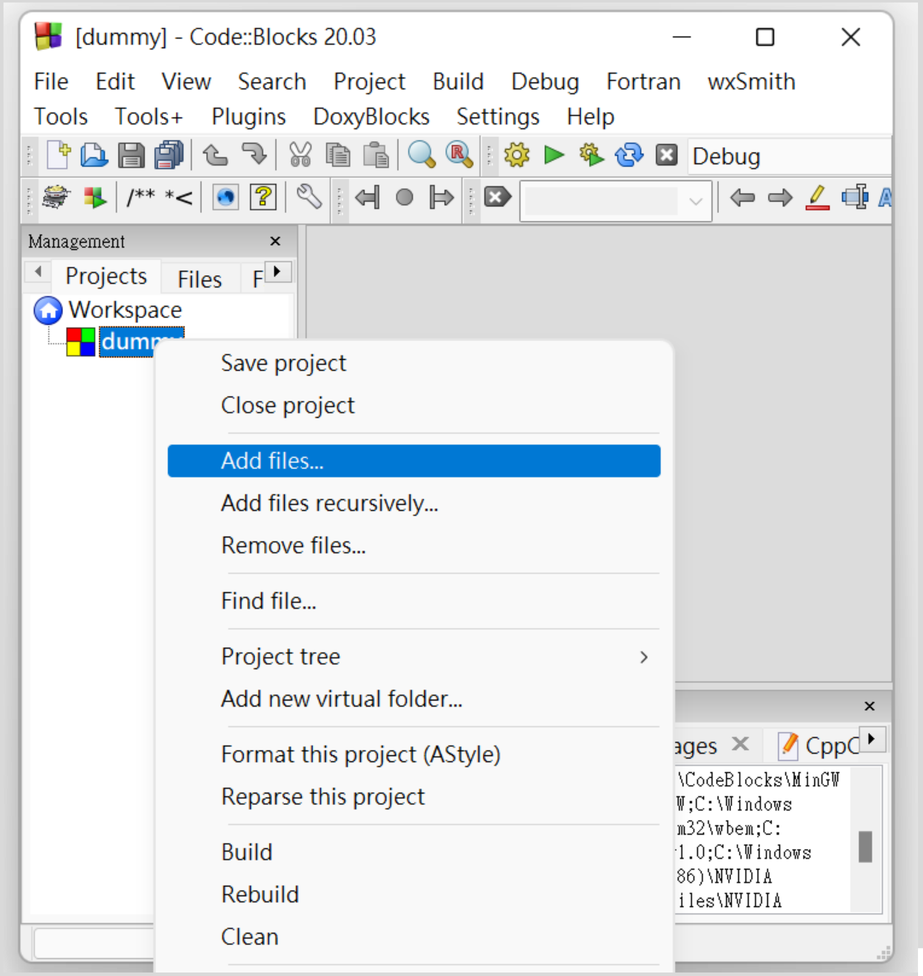
- Remove the original main.cpp file.



```
 Files\CodeBlocks\MinGW
s\MinGW;C:\Windows
\System32\wbem;C:
hell\v1.0;C:\Windows
les (x86)\NVIDIA
Corporation\PhysX\Common;C:\Program Files\NVIDIA
```

How to start (Using IDE Code::blocks)

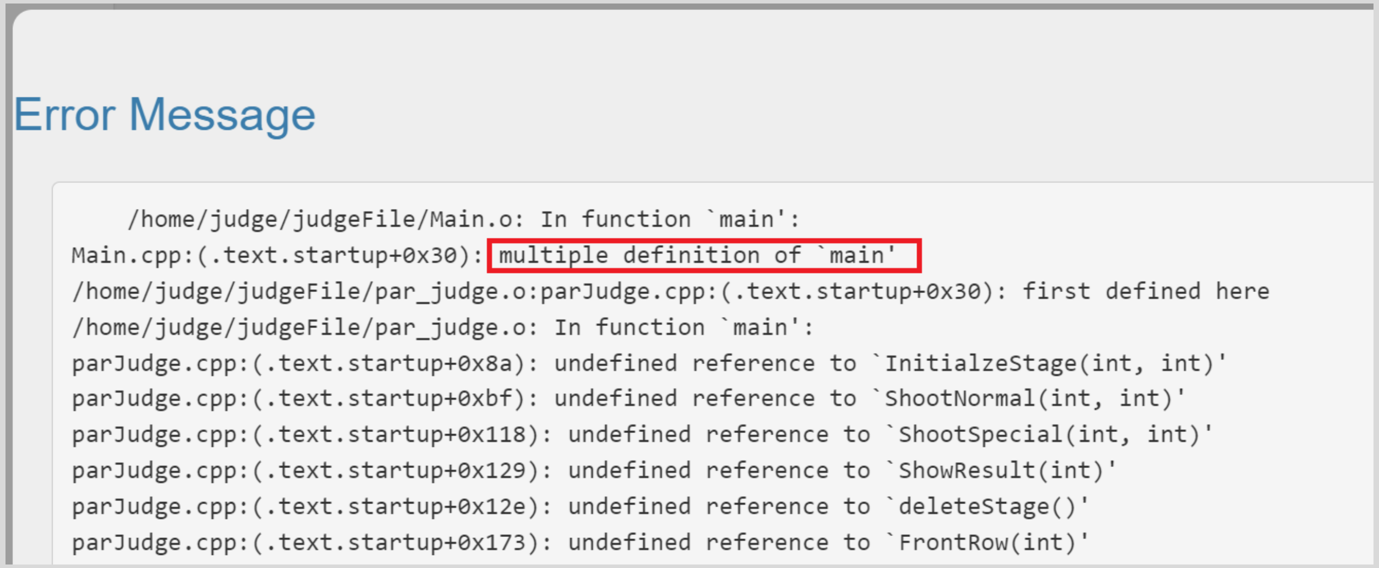
- Create a file “function.cpp”
- Add the 2 files you downloaded and the “function.cpp” file you just created into the project.
- Start writing your code but remember to write only in the “function.cpp” file.



```
\CodeBlocks\MinGW
W;C:\Windows
m32\wbem;C:
1.0;C:\Windows
86)\NVIDIA
iles\NVIDIA
```

How to start (Using IDE Code::blocks)

- You only need to submit your function.cpp code to OJ. (No need to upload the two files we provided)
- Otherwise, you might get a compile error since the two files we provided are already uploaded in the OJ.

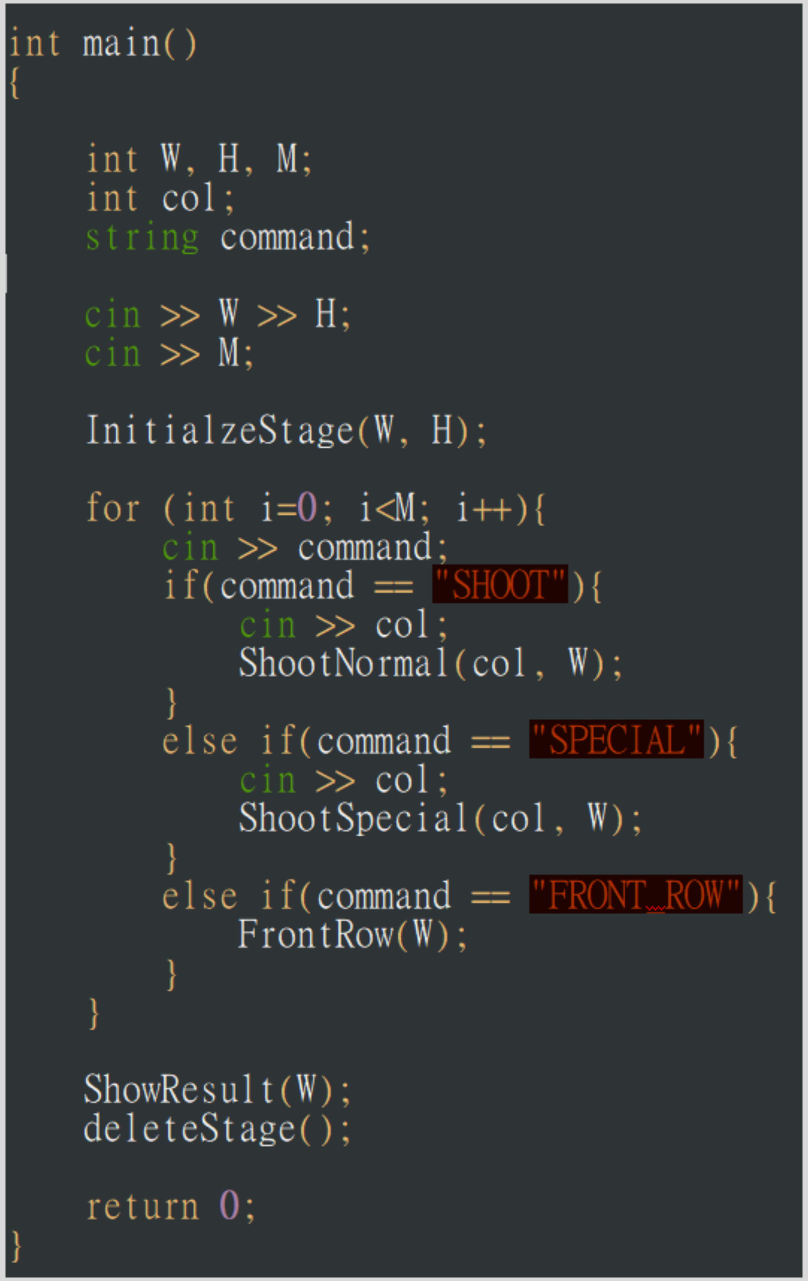


Error Message

```
    /home/judge/judgeFile/Main.o: In function `main':
Main.cpp:(.text.startup+0x30): multiple definition of `main'
/home/judge/judgeFile/par_judge.o:parJudge.cpp:(.text.startup+0x30): first defined here
/home/judge/judgeFile/par_judge.o: In function `main':
parJudge.cpp:(.text.startup+0x8a): undefined reference to `InitialzeStage(int, int)'
parJudge.cpp:(.text.startup+0xbf): undefined reference to `ShootNormal(int, int)'
parJudge.cpp:(.text.startup+0x118): undefined reference to `ShootSpecial(int, int)'
parJudge.cpp:(.text.startup+0x129): undefined reference to `ShowResult(int)'
parJudge.cpp:(.text.startup+0x12e): undefined reference to `deleteStage()'
parJudge.cpp:(.text.startup+0x173): undefined reference to `FrontRow(int)'
```

Understand the code

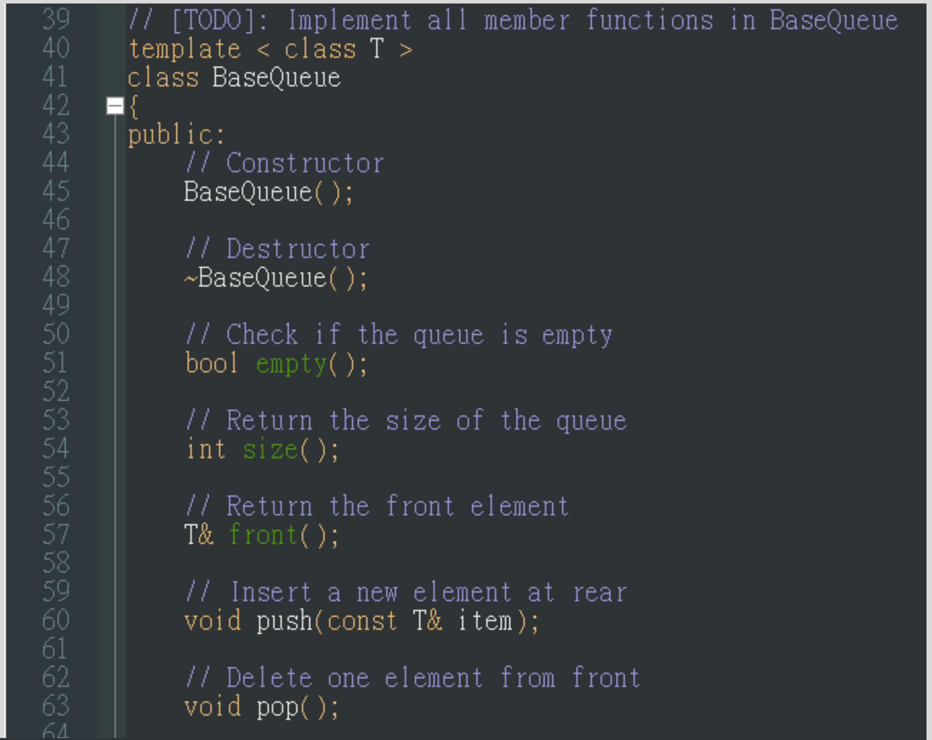
- 13453.cpp : Like an interaction interface
- It reads **some of** the inputs.
- It calls some functions specified in the header file with inputs as parameters.



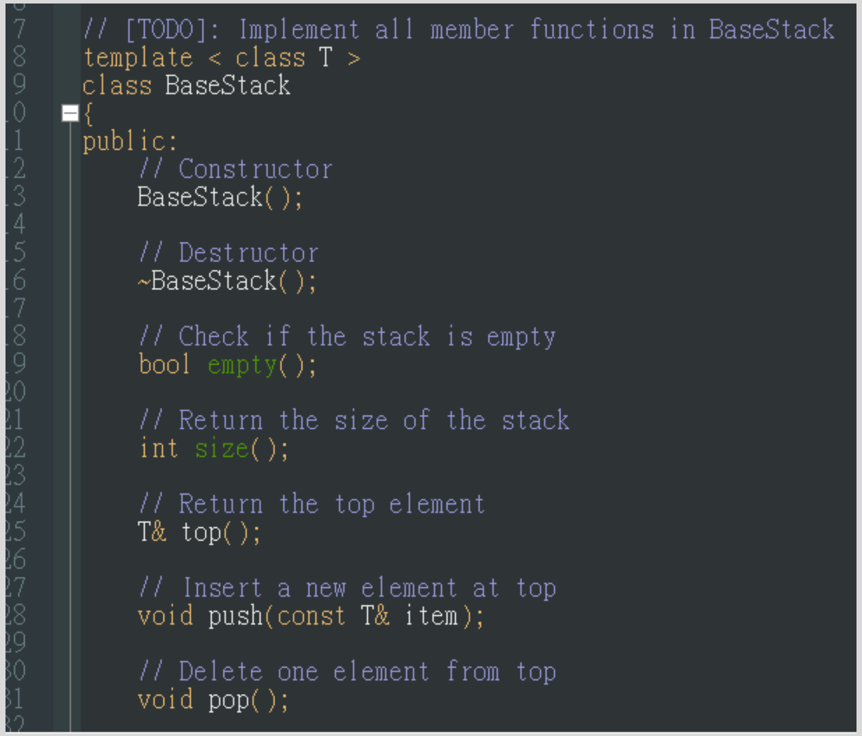
```
int main()
{

    int W, H, M;
    int col;
    string command;

    cin >> W >> H;
    cin >> M;

    InitialzeStage(W, H);

    for (int i=0; i<M; i++){
        cin >> command;
        if(command == "SHOOT"){
            cin >> col;
            ShootNormal(col, W);
        }
        else if(command == "SPECIAL"){
            cin >> col;
            ShootSpecial(col, W);
        }
        else if(command == "FRONT_ROW"){
            FrontRow(W);
        }
    }

    ShowResult(W);
    deleteStage();

    return 0;
}
```

Understand the code

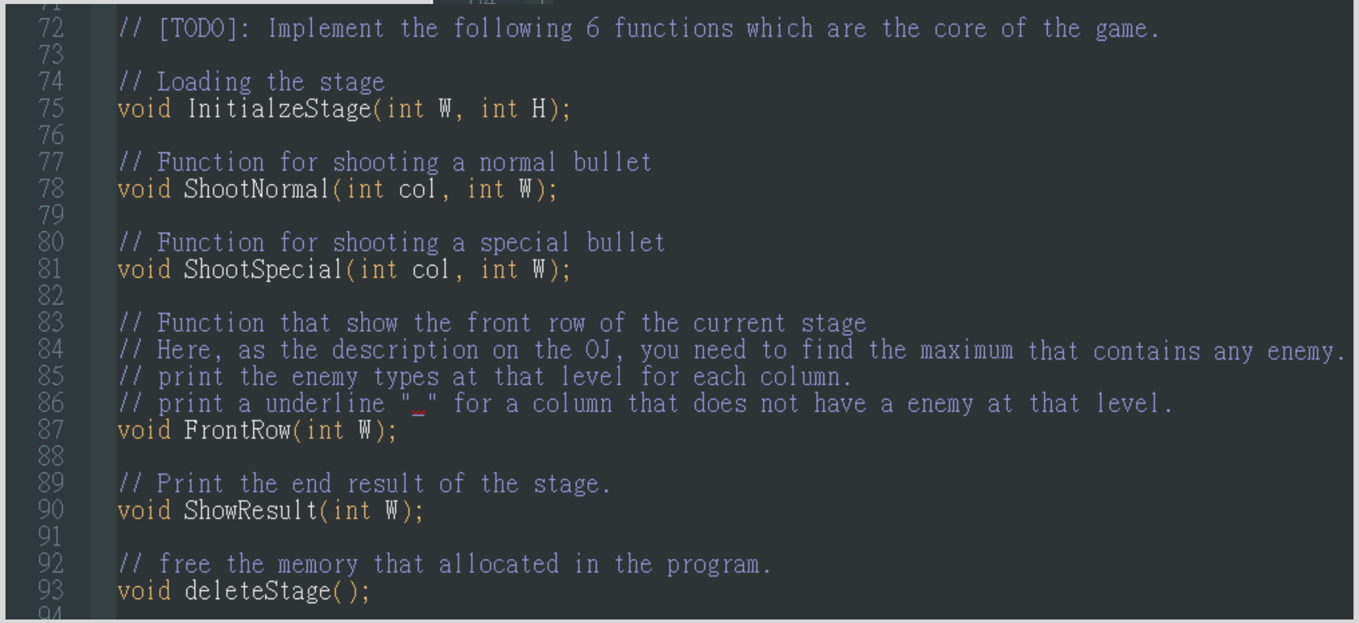
- 13453.h : **[TODO]** Tags tell you the underlying functions to implement in your own function.cpp file.



```
// [TODO]: Implement all member functions in BaseQueue
template < class T >
class BaseQueue
{
public:
    // Constructor
    BaseQueue();

    // Destructor
    ~BaseQueue();

    // Check if the queue is empty
    bool empty();

    // Return the size of the queue
    int size();

    // Return the front element
    T& front();

    // Insert a new element at rear
    void push(const T& item);

    // Delete one element from front
    void pop();
```



```
// [TODO]: Implement all member functions in BaseStack
template < class T >
class BaseStack
{
public:
    // Constructor
    BaseStack();

    // Destructor
    ~BaseStack();

    // Check if the stack is empty
    bool empty();

    // Return the size of the stack
    int size();

    // Return the top element
    T& top();

    // Insert a new element at top
    void push(const T& item);

    // Delete one element from top
    void pop();
```



```
// [TODO]: Implement the following 6 functions which are the core of the game.

// Loading the stage
void InitialzeStage(int W, int H);

// Function for shooting a normal bullet
void ShootNormal(int col, int W);

// Function for shooting a special bullet
void ShootSpecial(int col, int W);

// Function that show the front row of the current stage
// Here, as the description on the OJ, you need to find the maximum that contains any enemy.
// print the enemy types at that level for each column.
// print a underline "_" for a column that does not have a enemy at that level.
void FrontRow(int W);

// Print the end result of the stage.
void ShowResult(int W);

// free the memory that allocated in the program.
void deleteStage();
```

Hint

- Think about the **characteristics** of the **queues** and **stacks.**
- Choose a more suitable data structure for implementing **enemy position** and **special bullets shooting orders** depends on their characteristics.

Hint

- Conquer the problems one by one: (sample workflow)
  1. Make sure that your Stack and Queue work properly.
  2. Load the enemies’ position & types.
  3. Implement the function ShowResult(int W).
  4. Handling shooting normal bullets.
  5. Handling enemy effects.
  6. Handling special bullets effects.
  7. Handling front row queries.

Be careful about those empty slots. You might need to check them many times in an operation, depends on your design.

Hint

- You are allowed to write addition functions and declare global variables if that helps you implementing the system or debugging.

Hint

- You could refer the related resource provided on the course website if you got stuck. http://www.cs.nthu.edu.tw/~yishin/courses/EECS2040/EECS2040-2022.html
- Resource for C++ template syntax : https://vinesmsuic.github.io/2020/01/12/c++-template/#class-templates (Chinese) https://www.codeproject.com/Articles/257589/An-Idiots-Guide-to-Cplusplus-Templates-Part-1 (English)

Summary

- Deadline: 03/27 23:59
- Quiz related to this homework will be held on 03/28 18:30.
- Feel free to ask questions to our TAs if you don’t understand some part of the homework or have some trouble when doing the homework. But since we have limited time, we won’t help you debugging.
- No plagiarism is allowed!

Good Luck!