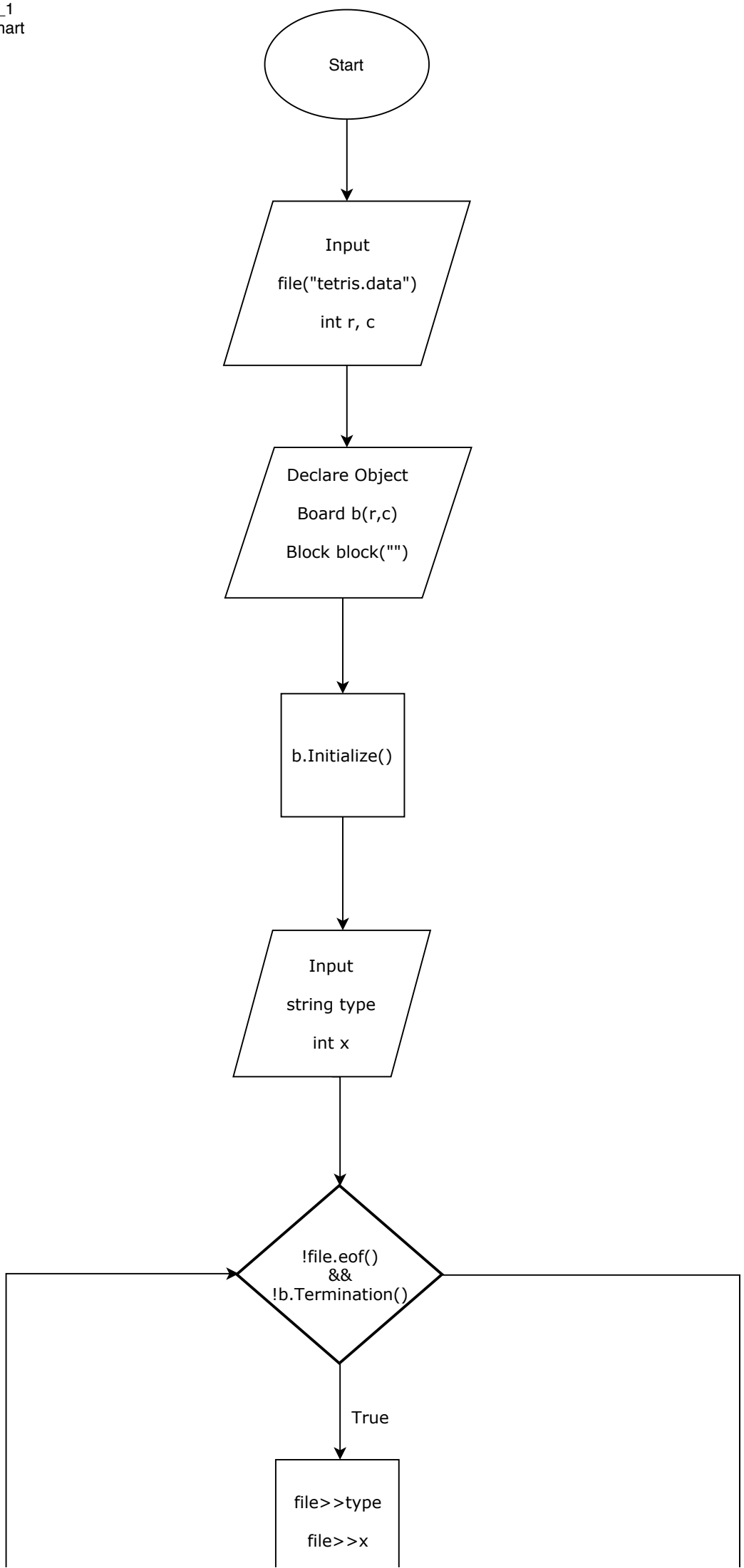
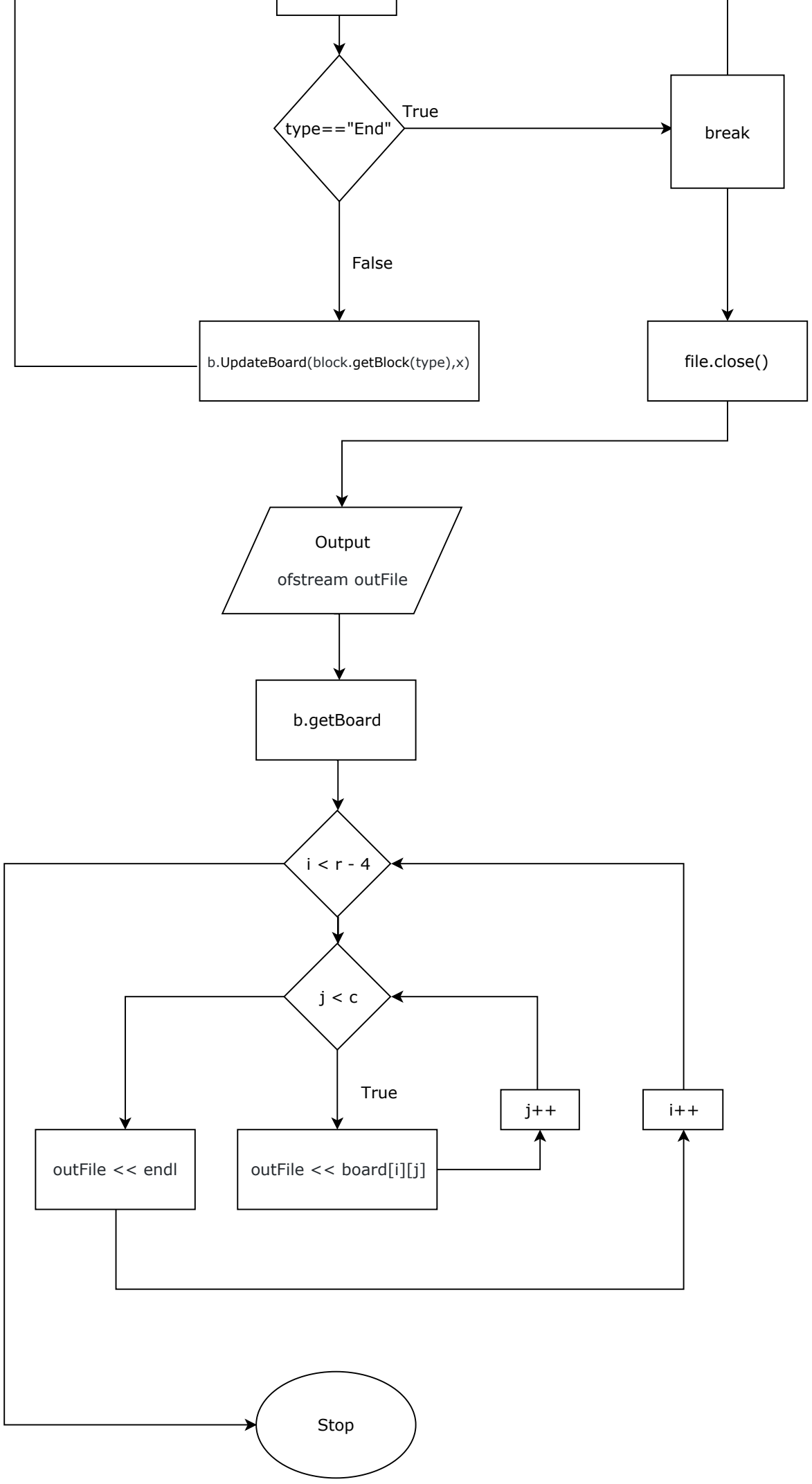
學號 1073007S
姓名 葉致廷




type=="End"
True
break
False
b.UpdateBoard(block.getBlock(type),x)
file.close()
Output
ofstream outFile
b.getBoard
i < r - 4
j < c
True
j++
i++
outFile << endl
outFile << board[i][j]
Stop

Project 1 Tetris Report

學號 1073007S
姓名 葉致廷

Classes Deployment:

class Board:

private data;

- two variables, row & col, to store the numbers of row and column respectively.
- a 2D-array, which is the main board of this game.
- an array of self-defined stacks.

public members:

- constructor:
  - declare
    - an (m+4)*n 2D-array
    - an n-size stack array.

- getRow(), getCol():
  - get private data, row and col.

- Initialization():
  - Initialize every cell of the 2D-array to 0. Initialize every stack of the stack array to the lowest row index of the board.

- getPositionY(int col):
  - return the appropriate row number to start updating the board.
  - algorithm:
    - check from index col to col+3 of the stack array, and find out the smallest number among the top element of these stacks.

- CheckCollision(int** block, int posx, int posy):
  - check if the incoming block would overlap with any current cell that has the value of 1 if put on posy. If it happens, return true, otherwise, return false.

- UpdateBoard(int** block, int posx):
  - update the 2D-array, our mainboard. while the game has not terminated
  - algorithm:
    - First, use getPositionY(posx) to get the supposed starting row.
    - Move posy down (posy++), until it reaches the bottom of the board or a collision happens.
    - If a collision happens, move up until it's not.
    - Start overlapping the 4*4 block with the board at the appropriate position.

- the block would start updating the board from the reference point block[3][0], and the corresponding point on the board would be board[posy][posx-1].
- Also, push the current highest occupied row number plus one to every corresponding stack, with the purpose of updating the top element of each involving stack. This way, we can easily get the appropriate posy next time.

- After finishing drawing the board, start scanning the board and delete every filled line.

- IsFilled():
  /*Thanks to 107062318's test case, I added this function to help*/
  check is the gameboard is filled

- DeleteAnyLine():
  - scan through the board and delete any filled line with DeleteLine_Naive(), yet, the first 4 rows of the board is not included.

- DeleteLine_Naive()
  - move every element of the matrix which is above the deleted line one row down.
  - update each stack after the movement.
  - update every cell at the top row to 0.

- getBoard()
  - get the private data, 2D-array.

class Block

- private data:
  a string variable Type
  a string array with a size of 19 to store all kinds of block
  a 2D-array to represent a 4*4 block

- public members;
  - constructor:
    - Initialize the string array, each element stores a distinct type of Type.

  - InitializeBlock(int** block):
    - initialize every cell of a 4*4 block to zero.

  - getElementIndex(string Type):
    - return the corresponding index of the parameter Type in the array.

- getBlock(string Type):
    - return an int** block that relates to the parameter Type
    - algorithm:
        - new a 4*4 2D-array and initialize it with InitializeBlock(int** block)
        - get the index of the parameter Type in the string array.
        - use switch case to overwrite the block
        - return the 2D-array

struct Node:

- int store: data of the node
- a pointer used to point at its successor node.
- constructor:

```
-       Node(int data = 0):store(data){}

-       Node(int data, Node* next):store(data),next_node(next){}
```

class Stack:

- private data:
    - a pointer to record the address of the top of the stack.
    - int Size used to store the size of the stack.
- public members;
    - constructor:
    - push(int posy):
        - push an element to the stack
        - algorithm:
            - if the stack is empty, new a node with values. assign the new node, which is a pointer, to the top node, stored our private data. Size add one.
            - Otherwise, construct a new node that it's nextnode points to the top address of the stack. Update the top address to this newly created node. Size add one.

    - top():
        - if the stack is empty, cout it's Empty and returns an error.
        - Otherwise, return the private data, store, of the top node.

    - isEmpty():
        - if the size is empty return true, else, return false.

int main()

- As described in the flow chart.

Test case Design

Goal:

1. **Block-Filling Correctness**
    - **Check every type of block**
    - **No Dummy filling**
    - **Ladder**
2. **The correctness of Filled-Lines Deletion**
3. **Termination Correctness**
    - **Top row filled and drop**
    - **Over the top and terminate**
    - **Put one more block to check if it really terminates**

Detailed Description:

Initiate a 20 X 15 grid

First, I try to check the elementary deletion

O 1
O 3
O 5
O 7
O 9
O 11
O 13

Check I1 Z2 S1 L1 T2 T1 J4 L3

I1 15
Z2 14
S1 13
L1 13
T2 14
T1 13
J4 10
L3 9
L3 9

Try to create a filled bottom but not successfully done

O 1
O 3
O 5
O 7

Randomly picking blocks to fill, still, it creates an opportunity to check deletions later

I2 1
I2 3
I2 5
S1 1
T4 1
T2 2
L2 4
J1 7

I2 4
I1 11
I1 12
O 11
I2 4
L1 1

Have fun and create a deletion and check the following blocks are placed correctly

I1 9 (deletion happens)
I2 2
I2 5
T3 2
O 10
L2 12
S2 14 (deletion happens)
L4 6 (deletion happens)
S2 12 (deletion happens)
I1 9
T1 12
O 1
O 4
O 6
L4 9
T2 14
J3 8
T4 10
Z1 10
T1 12
I2 6
I2 3
S1 1
J2 7
T1 4
O 8
I1 1 (Chain deletion happens)
S2 14 (deletion happens)
I1 7(deletion happens)
L3 12(deletion happens)
J4 3
I1 6(deletion happens)
T4 2
O 10(deletion happens)
T2 12
T2 14
I2 8
O 4(deletion happens)
T4 1
T4 3
O 7
O 9
O 11(deletion happens)
T2 13(deletion happens)
Z1 4(deletion happens)

Ladder time

O 1
O 2
O 3
O 4

Try to draw an image of the Statue of Liberty

I2 5
I2 8
I2 2
I1 1
L2 7
J4 4
T3 12
L3 14
O 6

Try to draw an image of a sky tower

O 14
O 14
O 14
O 14
T4 14
T4 13
S1 11
L3 14
O 14
L2 11
I2 11

Check T1 9

T1 9

Draw the with the Statue of Liberty self-flavor

O 6
O 5
I2 2
J4 13
L4 6

Create Termination checking

L2 1
I2 4

Check “Top row filled and drop”

T1 11(deletion happens)
T2 8 (deletion happens)

Check Termination

L1 14 (Termination happens)
O 5 (if Tetris does not terminate, this block will show)

End

git

- **history**

```
commit ae322f314009cc24d8617c960a21f548b928d0f3 (HEAD -> master, origin/master, origin/HEAD)
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Wed Oct 16 12:10:37 2019 +0800
```

-

```
(base) macdeMacBook-Air-5:Tetris_vol.1 mac$ git log
commit 04c4f6d43179041eb216a156c54fa3af6297a36d (HEAD -> master, origin/master, origin/HEAD)
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Tue Oct 15 15:28:25 2019 +0800

    Tetris.cpp modified

commit 60bfa7b153a24b14532f527f626a6e3e373392b5
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Wed Oct 9 10:51:15 2019 +0800

    update: Testcases verified

commit e0d36729e14b8536cc33049e564ebded0b4aa564
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Tue Oct 8 15:46:13 2019 +0800

    update: Data.h not needed

commit eb3177efed1515d4096f141c3fec03bbd98f1685
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Tue Oct 8 15:43:50 2019 +0800

    update: Delete Data darray in main()

commit f2c2e5703ab26f9f4f41c8de9fc3fcfb230f4f47
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Tue Oct 8 12:24:11 2019 +0800

    update: DeleteLine_Navie

commit 0452ee4c28ef662162cb74c803c31eb3beb08d02
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Mon Oct 7 23:11:34 2019 +0800

    update: tetris.data

commit f34d297ba86a2245c7ad2d9ffa1949e61377c017
Author: justinyeh <justinyeh1995@gmail.com>
Date:   Sat Oct 5 19:15:02 2019 +0800
```

repository:
https://github.com/justinyeh1995/108_1_DS_Project_Tetris-