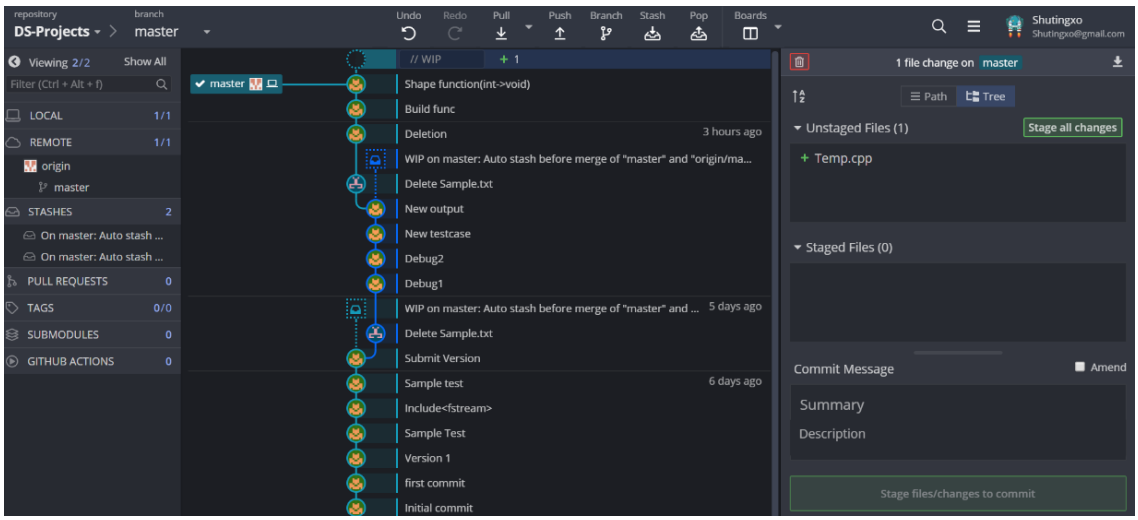
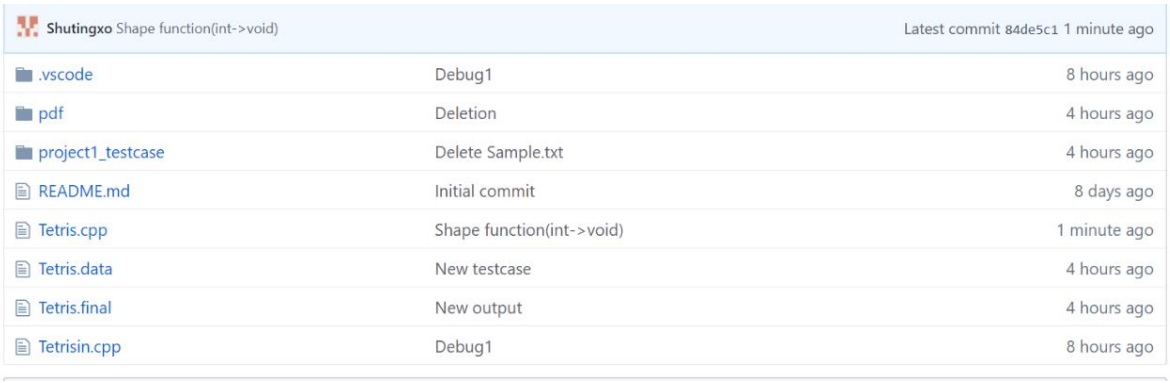
1. Project Description

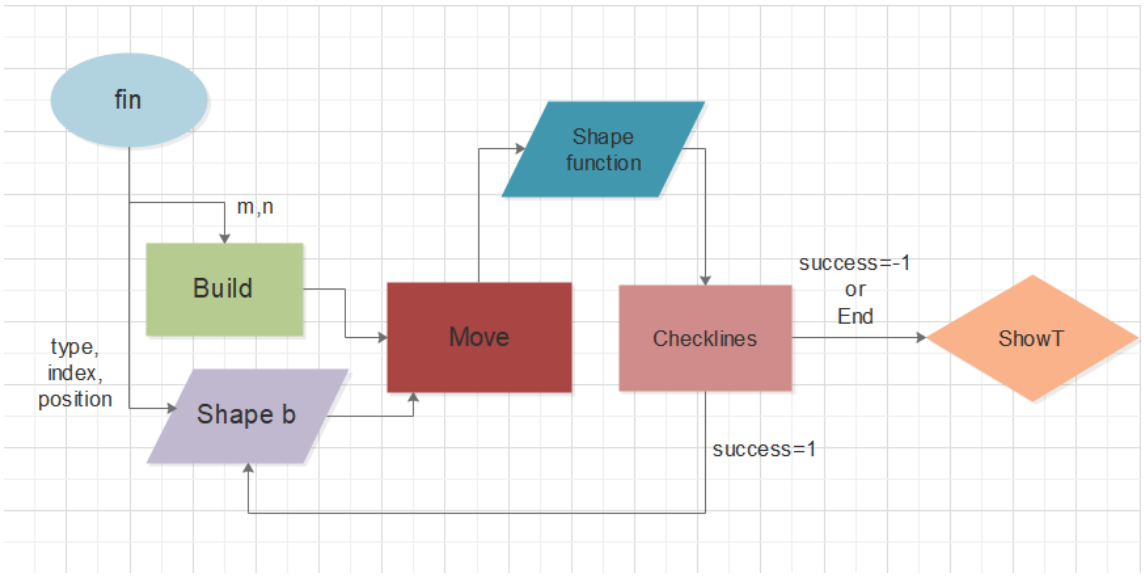
(1) GitKraken



(2) GitHub



(3) Program Flow Chart



(4) Detailed Description

1) class Shape

```
class Shape{
    public:
        Shape(char t,int id):type(t),index(id){};
        char gettype(){
            return type;
        }
        int getindex(){
            return index;
        }
        int getposition(){
            return pos;
        }
        void setposition(int p){
            pos = p;
        }
    private:
        char type;
        int index;
        int pos;
};
```

Class Shape 用來存放 19 種 Shape 的 type(T,L,J,S,Z,I,O),index(1,2,3,4)還有給定的 column。

2) Build function

```
int** Build(int r,int c){
    int** Tetris = new int*[r+4];
    for(int i=0;i<r+4;i++){
        Tetris[i] = new int[c];
        for(int j=0;j<c;j++){
            Tetris[i][j] = 0;   //init with 0
        }
    }return Tetris;
}
```

這個 function 使用雙重指標來建立 Tetris 的平面，並且在 Array 上方保留了 4 行(r+4)，所以是一個(r+4)*c 的 Array。(row*col)

上面 4 行是預留給超出 Tetris 平面的圖形，在下一個 Checklines function 中，檢查完要消掉的 row 並全部消掉之後，會檢查最上面的這 4 行有沒有剩餘的 1。(暫時將這 4 行稱為 Buffer)

如果有剩餘的 1，代表圖形超出平面，就會直接結束。

3) Checklines function

```
36  int Checklines(int **T,int r,int c){
37      int k=(r+4)-1;
38      int i, j,count;
39      int suc=1;
40      for(i=r+4-1;i>=0 ;i--){
41          count=0;
42          for(j=0;j<c;j++){
43              if(T[i][j]==1 && k>=4) count++;
44              T[k][j] = T[i][j];
45          }
46          if(count<c) k--;
47      }
48      for (k=3; k >= 0;k--){
49          for (j = 0; j < c;j++){
50              if(T[k][j]==1){
51                  suc = -1;
52                  break;
53                  }
54              }
55          if(suc==-1)   break;
56          }
57      return suc;
58  }
```

1.第 1 個 for 迴圈檢查 row 會不會消掉，從底部開始(k=(r+4)-1),k 跟 i 同步。如果 i 那一行的 count==c，代表要被消掉，k 會停留在原本的 row，i 繼續往上走。假設 i==8,k==9，那 T[k][j]=T[i][j]，第 9 行就會消掉，第 8 行掉下來。當 count<c，代表不需要消掉，所以 k--，往上檢查。
If(T[i][j]==1&& k>=4)，因為只有平面裡的 row 需要被消掉，所以只有在 k>=4 的時候，才需要 count。(不然會消到上面 Buffer 中的圖形)
2.第 2 個 for 迴圈檢查 Buffer 部分有沒有 1，如果有就 fail，return -1。
如果 Buffer 沒有 1，代表圖形沒有超出平面，return 1。

4) Move function

```
334  int Move(int **T,Shape b,int r,int c){
335      int p,success=-1;
336      fin >> p ; b.setposition(p);
337      switch(b.gettype()){
338          case('T'):
339              ShapeT(T,b,r,c); break;
340          case('L'):
341              ShapeL(T,b,r,c); break;
342          case('J'):
343              ShapeJ(T,b,r,c); break;
344          case('S'):
345              ShapeS(T,b,r,c); break;
346          case('Z'):
347              ShapeZ(T,b,r,c); break;
348          case('I'):
349              ShapeI(T,b,r,c); break;
350          case('O'):
351              ShapeO(T,b,r,c); break;
352          default: break;
353      }
354      success = Checklines(T,r,c);
355      return success;
356  }
```

從 file 中取得 Shape 的初始 column，再依照 type 呼叫圖形的 function。圖形放好後，呼叫 Checklines function 檢查 rows。如果 success==1 代表可以繼續，return 1 ; success==-1，代表結束，return -1。

5) main function

```
358  int main(){
359      if(!fin){ //check file
360          cout << "Filein error!";
361          exit(1);
362      }
363      if(!fout){
364          cout << "Fileout error!";
365          exit(1);
366      }
367      int m,n,success=-1;
368      fin >> m >> n ;      //row,column
369      int **T = Build(m,n);
370      char t; int id;
371      while(fin >> t && t!='E'){
372          if(t!='O')
373             fin >> id;
374          else if(t=='O')
375             id=0;
376          Shape b(t,id);       //set shape
377          success = Move(T,b,m,n);
378          if(success==-1)
379            break;
380      } ShowT(T,m,n);
381  }
```

前面 2 個 if 檢查 file 有沒有成功開啟。如果失敗，印出”error”就結束。讀取 m,n 之後(row&column)，呼叫 Build 建立 Tetris 平面。再使用 while 迴圈一直讀取 fin 中圖形的 type 跟 index，呼叫 Shape b(t,id)初始化。呼叫 Move 開始在平面裡讓方塊掉落。如果成功，Move 會 return 1；失敗的話，return -1。If success==-1，就會直接結束，印出 final output。

6) ShowT function

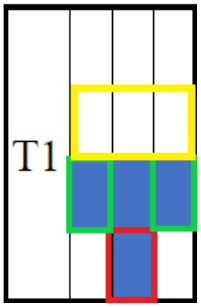
```
326  void ShowT(int **T,int r,int c){
327      for(int i=4;i<r+4;i++){
328          for(int j=0;j<c;j++){
329              fout << T[i][j] ;
330          }
331          fout << endl;
332      }
333      fout.close();
334  }
```

從第 4 行開始印 Tetris 平面(i<4 皆為 Buffer)，印完之後將 file 關閉。

7) Shape T function

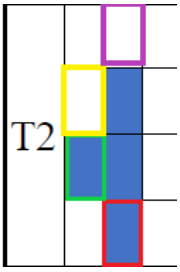
```
void ShapeT(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    int k,j;
    switch(b.getindex()){
        case(1):…
        case(2):…
        case(3):…
        case(4):…
        default:
            break;
    }
}
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p+1]==0 && T[k-1][p]==0 && T[k-1][p+2]==0){
            T[k][p+1]=1; T[k-1][p]=1; T[k-1][p+2]=1; count++;
            if(count>2){
                T[k-2][p]=0; T[k-2][p+1]=0; T[k-2][p+2]=0;
            }
        }else if(count==0 && T[k][p+1]==0){
            T[k][p+1]=1; count++;
        }
        else break;
    }break;
```



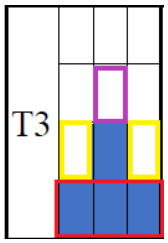
T1:一開始 count==0，紅色那格要進入 Tetris 平面，檢查 T[k][p+1]==0；成功的話，count==1；只要 count>=1，每次往下掉都要檢查紅色&綠色方塊；count>2 的話代表 T1 全部進到平面裡，往下時要把黃色方塊重新設為 0。

```
case(2):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p+1]==0 && T[k-1][p]==0){
            T[k][p+1]=1; T[k-1][p]=1; count++;
            if(count==3){
                T[k-2][p]=0; //clear
            }else if(count>3){
                T[k-2][p]=0; T[k-3][p+1]=0;
            }
        }else if(count==0 && T[k][p+1]==0){
            T[k][p+1]=1; count++;
        }else break;
    }break;
```



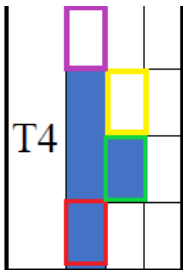
T2:count==0，檢查紅色方塊的位置 T[k][p+1]==0,count++；count>=1，檢查紅色&綠色方塊位置；count==3，T2 全部進入平面，把黃色方塊設為 0；count>3，往下掉的時候要把黃色&紫色方塊設為 0。

```
case(3):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0 && T[k][p+2]==0){
            T[k][p]=1; T[k][p+1]=1; T[k][p+2]=1; count++;
            if(count==2){
                T[k-1][p]=0; T[k-1][p+2]=0;
            }else if(count>2){
                T[k-1][p]=0; T[k-1][p+2]=0; T[k-2][p+1]=0;
            }
        }else break;
    }break;
```



T3:往下掉的時候只需要檢查紅色方塊的位置；count==2 的時候，T3 全部進入平面，黃色方塊設為 0；count>=2，往下時黃色&紫色方塊設為 0。

```
case(4):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p]==0 && T[k-1][p+1]==0){
            T[k][p]=1; T[k-1][p+1]=1; count++;
            if(count==3){
                T[k-2][p+1]=0; //clear
            }else if(count>3){
                T[k-2][p+1]=0; T[k-3][p]=0;
            }
        }else if(count==0 && T[k][p]==0){
            T[k][p]=1; count++;
        }else break;
    }break;
```

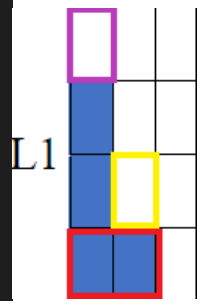


T4:跟 T2 一樣的方法，只是左右相反。

8) Shape L function

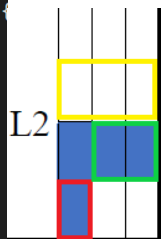
```
void ShapeL(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    int k;
    switch(b.getindex()){
        case(1):…
        case(2):…
        case(3):…
        case(4):…
        default:
            break;
    }
}
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0){
            T[k][p]=1; T[k][p+1]=1; count++;
            if(count==2 ||count ==3)
                T[k-1][p+1]=0;
            else if(count>3){
                T[k-1][p+1]=0; T[k-3][p]=0;
            }
        }else break;
    }break;
```



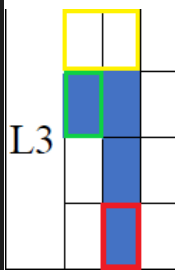
L1:檢查紅色方塊的位置；count==2 或 3，往下將黃色方塊設為 0；count>3，L1 全部進入平面，往下時將黃色&紫色方塊設為 0。

```
case(2):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p]==0 && T[k-1][p+1]==0 && T[k-1][p+2]==0){
            T[k][p]=1; T[k-1][p+1]=1; T[k-1][p+2]=1; count++;
            if(count>2){
                T[k-2][p]=0; T[k-2][p+1]=0; T[k-2][p+2]=0;
            }
        }else if(count==0 && T[k][p]==0){
            T[k][p]=1; count++;
        }else break;
    }break;
```



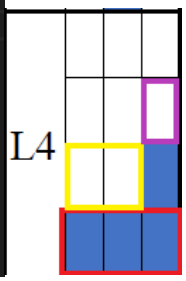
L2:count==0，檢查紅色方塊；count>=1，檢查紅色&綠色方塊；count>2，L2 全部進入平面，每一次往下時將黃色方塊設為 0。

```
case(3):
    for(k=0;k<r+4;k++){
        if(count>=2 && T[k][p+1]==0 && T[k-2][p]==0){
            T[k][p+1]=1; T[k-2][p]=1; count++;
            if(count>3){
                T[k-3][p+1]=0; T[k-3][p]=0;
            }
        }else if(count<2 && T[k][p+1]==0){
            T[k][p+1]=1; count++;
        }else break;
    }break;
```



L3:count<2，檢查紅色方塊；count>=2，檢查紅色&綠色方塊；count>3，L3 全部進入平面，每次往下時將黃色方塊設為 0。

```
case(4):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0 && T[k][p+2]==0){
            T[k][p]=1; T[k][p+1]=1; T[k][p+2]=1; count++;
            if(count>2){
                T[k-1][p]=0; T[k-1][p+1]=0; T[k-2][p+2]=0;
            }else if(count==2){
                T[k-1][p]=0; T[k-1][p+1]=0;
            }
        }else break;
    }break;
```

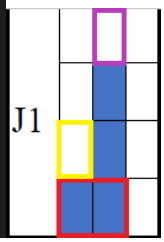


L4:每次往下掉時檢查紅色方塊；count==2，將黃色方塊設為 0；count>2，每次往下掉時將黃色&紫色方塊設為 0。

9) Shape J function

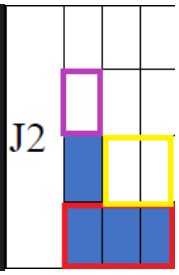
```
void ShapeJ(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    int k;
    switch(b.getindex()){
        case(1):…
        case(2):…
        case(3):…
        case(4):…
        default:
            break;
    }
}
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0){
            T[k][p]=1; T[k][p+1]=1; count++;
            if(count>3){
                T[k-1][p]=0; T[k-3][p+1]=0;
            }else if(count==2 || count==3)
                T[k-1][p]=0;
        }else break;
    }break;
```



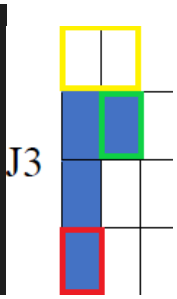
J1:跟 L1 一樣的方法，只是左右相反。

```
case(2):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0 && T[k][p+2]==0){
            T[k][p]=1; T[k][p+1]=1; T[k][p+2]=1; count++;
            if(count>2){
                T[k-1][p+1]=0; T[k-1][p+2]=0; T[k-2][p]=0;
            }else if(count==2){
                T[k-1][p+1]=0; T[k-1][p+2]=0;
            }
        }else break;
    }break;
```



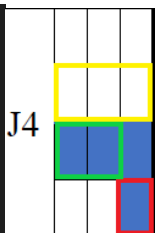
J2:跟 L4 一樣的方法，左右相反。

```
case(3):
    for(k=0;k<r+4;k++){
        if(count>=2 && T[k][p]==0 && T[k-2][p+1]==0){
            T[k][p]=1; T[k-2][p+1]=1; count++;
            if(count>3){
                T[k-3][p]=0; T[k-3][p+1]=0;
            }
        }else if(count<2 && T[k][p]==0){
            T[k][p]=1; count++;
        }else break;
    }break;
```



J3:跟 L3 一樣的方法，左右相反。

```
case(4):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p+2]==0 && T[k-1][p]==0 && T[k-1][p+1]==0){
            T[k][p+2]=1; T[k-1][p]=1; T[k-1][p+1]=1; count++;
            if(count>2){
                T[k-2][p]=0; T[k-2][p+1]=0; T[k-2][p+2]=0;
            }
        }else if(count==0 && T[k][p+2]==0){
            T[k][p+2]=1; count++;
        }else break;
    }break;
```

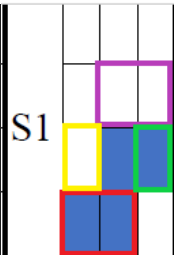


J4:跟 L2 一樣的方法，左右相反。

10) Shape S function

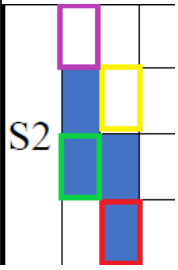
```
void ShapeS(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    int k;
    switch(b.getindex()){
        case(1):…
        case(2):…
        default:
            break;
    }
}
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p]==0 && T[k][p+1]==0 && T[k-1][p+2]==0){
            T[k][p]=1; T[k][p+1]=1; T[k-1][p+2]=1; count++;
            if(count==2) T[k-1][p]=0;
            else if(count>2){
                T[k-1][p]=0; T[k-2][p+1]=0; T[k-2][p+2]=0;
            }
        }else if(count==0 && T[k][p]==0 && T[k][p+1]==0){
            T[k][p]=1; T[k][p+1]=1; count++;
        }else break;
    }break;
```



S1:count==0，檢查紅色方塊；count>=1，檢查紅色&綠色方塊；count==2，將黃色方塊設為 0；count>2，往下時將黃色&紫色設為 0。

```
case(2):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p+1]==0 && T[k-1][p]==0){
            T[k][p+1]=1; T[k-1][p]=1; count++;
            if(count==3) T[k-2][p+1]=0;
            else if(count>3){
                T[k-2][p+1]=0; T[k-3][p]=0;
            }
        }else if(count==0 && T[k][p+1]==0){
            T[k][p+1]=1; count++;
        }else break;
    }break;
```

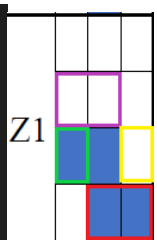


S2:count==0，檢查紅色方塊；count>=1，檢查紅色&綠色方塊；count==3，將黃色設為 0；count>3，往下掉將黃色&紫色設為 0。

11) Shape Z function

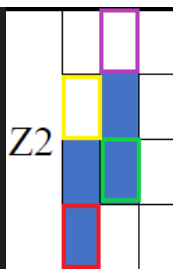
```
void ShapeZ(int**T,Shape b,int r,int c){
    int count = 0, p = b.getposition() - 1;
    int k;
    switch(b.getindex()){
        case(1):…
        case(2):…
        default:…
    }
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p+1]==0 && T[k][p+2]==0 && T[k-1][p]==0){
            T[k][p+1]=1; T[k][p+2]=1; T[k-1][p]=1; count++;
            if(count==2) T[k-1][p+2]=0;
            else{
                T[k-2][p]=0; T[k-2][p+1]=0; T[k-1][p+2]=0;
            }
        }else if(count==0 && T[k][p+1]==0 && T[k][p+2]==0){
            T[k][p+1]=1; T[k][p+2]=1; count++;
        }else break;
    }break;
```



Z1:跟 S1 一樣的方法，左右相反。 Z2:跟 S2 一樣的方法，左右相反。

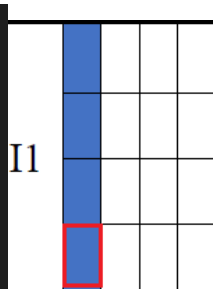
```
case(2):
    for(k=0;k<r+4;k++){
        if(count>=1 && T[k][p]==0 && T[k-1][p+1]==0){
            T[k][p]=1; T[k-1][p+1]=1; count++;
            if(count==3) T[k-2][p]=0;
            else if(count>3){
                T[k-2][p]=0; T[k-3][p+1]=0;
            }
        }else if(count==0 && T[k][p]==0){
            T[k][p]=1; count++;
        }else break;
    }break;
```



12) Shape I function

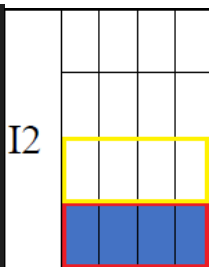
```
void ShapeI(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    int k,j;
    switch(b.getindex()){
        case(1):…
        case(2):…
        default:…
    }
```

```
case(1):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0){
            if(count < 4){
                T[k][p]=1;  count++;
        }else{
            T[k][p] = 1;
            T[k-count][p]= 0; }
        }else break;
    }break;
```



I1:檢查紅色方塊；count>4，I1 全部進入平面，往下時要將最後一個方塊的經過的地方設為 0。

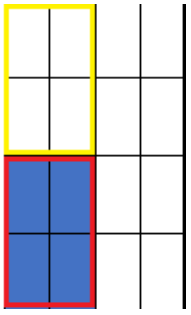
```
case(2):
    for(k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0 && T[k][p+2]==0 && T[k][p+3]==0){
            count =1;
            for(j=p;j<p+4;j++){
                T[k][j]=1;
                if(k>0) T[k-1][j]=0;
            }
         }else break;
    }break;
```



I2:檢查紅色方塊；每次往下掉落將黃色方塊設為 0。

13) Shape O function

```
void ShapeO(int**T,Shape b,int r,int c){
    int count=0,p=b.getposition()-1;
    for(int k=0;k<r+4;k++){
        if(T[k][p]==0 && T[k][p+1]==0 ){
            if(count<2){
                T[k][p]=1; T[k][p+1]=1;
                count++;
            }else{
                T[k][p]=1; T[k][p+1]=1;
                T[k-2][p]=0; T[k-2][p+1]=0;
            }
        }else break;
    }
}
```



O:檢查紅色方塊，每次往下掉落將黃色方塊設為 0。

2. Test case Design : 15*4 Matrix, 29 blocks.

Tetris.data

```
15 4
O 3
T4 2
J3 1
S1 1
S2 3
J4 2
L2 1
Z1 1
Z2 3
L1 1
L3 1
J1 3
I1 3
I2 1
T4 1
T2 3
T1 1
T1 1
I1 4
L4 2
I1 1
T3 1
J3 1
J2 1
O 2
L3 3
S1 1
J1 2
I1 1
End
```

Test case includes all types of blocks , and tests whether the program can place the blocks and eliminate the rows correctly.
The output should end at the 29th line,"J1 2",when the block is out of Tetris Matrix.

Tetris.final

```
0010
0110
0110
0111
1110
1100
1100
1011
1011
1100
1100
1100
1110
0111
0011
```