Hints for Task C of Game 1024

Task C (40 points)
Define code to press UP, DOWN, LEFT, RIGHT. That is, define pressUp, pressDown, pressLeft and pressRight methods of Board.cpp.

Note that selectRandomCell method is called in these above methods. In case your version of selectRandomCell in Task B has errors, you would fail Task C. To avoid that scenario, we provide a corrected version of selectRandomCell in gradescope scripts. As a result, do not include your definition of selectRandomCell and constructors/destructor in submission of Task C.

1. From the submission of Task B, which includes print and selectRandomCell methods, add definition of pressLeft, pressRight, pressDown, and pressUp methods.
2. When you define pressLeft, pressRight, pressDown, and pressUp methods, remember to call selectRandomCell method in the end. This is because, after an arrow key is pressed, a random empty cell is selected and number 1 is put in that cell, these steps are defined in selectRandomCell.
3. Before submission, comment the definition of selectRandomCell since a correct version of it will be provided by gradescope scripts. If you include the definition of selectRandomCell method in submission of Task C, compiler would complain that that method is redefined.
   In short, only include codes for **pressLeft**, **pressRight**, **pressUp**, **pressDown**, and **print** methods in Task C.

   Submit only Board.cpp.

Key ideas of press up

For **each** column do the following
begin
   (1) Save the non-zeros values in a column **from top to bottom** to a vector of integers.
   (2) Merge these non-zero values in the above vector. Suppose the current item has index 0, then its down-stair neighbor in the vector has index 1. Suppose the current item has index 1, then its down-stair neighbor in the vector has index 2, and so on.
   Start from index zero,
      (a) **If** the current item has a down-stair neighbor **and** they share the same value, double the value of the current item and set the second item to be zero. Increase the index of vector by … (you fill the number by yourself) to skip the current item and its down-stair neighbor.
      (b) Otherwise (either the current item does not have down-stair neighbor, or it does not share the same value as its down-stair neighbor), the value of current item is not changed. Increase the index of vector by … (you fill the number yourself) to skip the current item.
      (c) Continue (a) and (b) until every item in the vector is processed.
   (3) Start from the item of vector, copy all the non-zeros to the original column from top to bottom.
   Do not forget to pad the remaining elements in the column to be zeros.
end

For example, suppose we have following 5 x 4 panel.

row\column	0	1	2	3
0	2			
1	2			
2	0			
3	1			
4	2			

- In the first column, after Step (1), we get the following vector of integers.

index	
0	2
1	2
2	1
3	2

- Current index (represented by variable curr) starts from 0. The item indexed at curr has a down-stair neighbor since (curr + 1) is smaller than the size of the vector and these two items share the same value. **update max if necessary.**

curr	
0	2
	2
	1
	2

- Merge the current item and its down-stair neighbor, double the current item while set its neighbor to be zero. Increase curr by 2. Now curr is 2.

	4
	0
curr 2	1
	2

- Even though the current item indexed at 2 has a down-stair neighbor, they do not share values, so there is no way to merge. Move to the next item by increasing curr by 1. Now curr is changed to 3.

	4
	0
	1
curr 3	2

- The current item indexed at 3 has no down-stair neighbor, no need to merge. Move to the next item by increasing curr by 1. Now curr is 4, which is larger than or equal to the size of vector, so every item in the above vector is processed.

- Copy all the non-zeros in the vector to the first column, from top to bottom (do not forget to pad the remaining elements to be zero). Now the panel looks as follows.

4			
1			
2			
0			
0			

- Continue to work the rest columns.
- Afterwards, call selectRandomCell(int& row, int& col) to place number 1 in a randomly chosen cell with value zero.

Pseudocode of pressUp

```
begin
    for (int col = ?; col < ??; …)
    begin
        vector<int> nonZeros;

        //get all the non-zeros of (col)th column to nonZeros, from top to bottom.
        for (int row = …; row < …; …)
        begin
            if (panel[row][col] is not zero)
                push panel[row][col] back to nonZeros;
        end

        //merge adjacent same value elements of nonZeros
        int curr = 0; //index of nonZeros vector
        while (curr < …)
        begin
            if (curr+1 < … && nonZeros[…] == nonZeros[…])
            begin
                set nonZeros[curr] to be …
                set nonZeros[curr+1] to be …
                update max if necessary
                update curr by …
            end
            else update curr by …
        end

        //copy non-zero element of non-zeros to the column, from top to bottom
        int row = …;
        for (curr = …; curr < …; update curr)
        begin
            if (nonZeros[curr] is not zero)
            begin
                set panel[row][col] by nonZeros[curr];
```

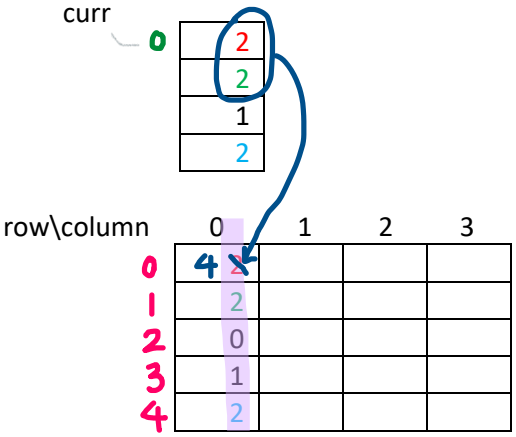
```
                update row;
            end
        end

        pad the remaining elements in the column to be zeros
    end

    call selectRandomCell method
end
```

Simplified version of pressUp

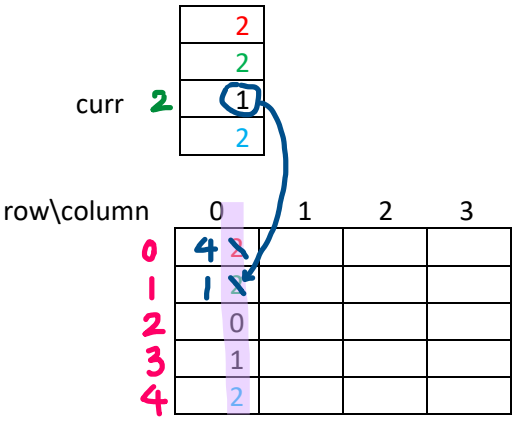
Modification: When merging adjacent same value elements of *nonZeros* vector, instead of updating the elements of nonZeros, copy the merged result to the column, from top to bottom.

For example, suppose vector nonZeros is as follows.

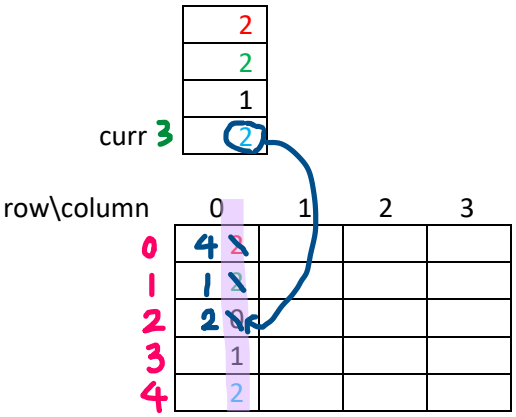
- Current index (represented by variable curr) starts from 0. The item indexed at curr has a down-stair neighbor since (curr + 1) is smaller than the size of the vector and these two items share the same value. Copy this value to the first element in the column. Increase curr by 2. **Update max if necessary.**



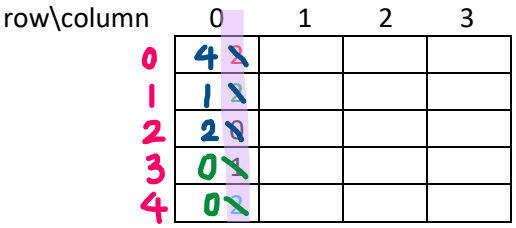
- Now curr is 2. Since this element does not share the same value with its down-stair neighbor, only copy this value to the next element in the column. Then curr is updated to be 3.



- Now curr is 3. Since it has no down-stair neighbor, there is no way to merge, copy this value to the corresponding element in the column.



- Pad the remaining elements in the column to be zero.



Pseudocode of pressUp

```
begin
        for (int col = ?; col < ??; …)
        begin
            vector<int> nonZeros;

            //get all the non-zeros of (col)th column to nonZeros, from top to bottom.
            for (int row = …; row < …; …)
            begin
                if (panel[row][col] is not zero)
                    push panel[row][col] back to nonZeros;
            end

            //merge adjacent same value elements of nonZeros and
            //copy non-zero element of non-zeros to the column, from top to bottom.
            int curr = 0; //index of vector nonZeros
            int row = …; //ADD
            while (curr < …)
            begin
                if (curr+1 < … && nonZeros[…] == nonZeros[…])
```

```
                begin
                    ~~set nonZeros[i] to be … //REMOVE~~
                    ~~set nonZeros[i+1] to be … //REMOVE~~
                    set panel[row][col] by … //ADD
                    update max if necessary
                    update curr by …
                end
                else
                begin
                    set panel[row][col] by … //ADD
                    update curr by …
                end
                update row by …
            end

            ~~int row = 0; //REMOVE~~
            ~~for (curr = …; curr < …; update curr) //REMOVE~~
            ~~begin //REMOVE~~
                ~~if (nonZeros[curr] is not zero) //REMOVE~~
                ~~begin //REMOVE~~
                    ~~set panel[row][col] by nonZeros[curr]; //REMOVE~~
                    ~~update row; //REMOVE~~
                ~~end //REMOVE~~
            ~~end //REMOVE~~

            pad the remaining elements in the column to be zeros
        end

        call selectRandomCell method
end
```

Pseudocode of pressDown

The only difference between pressDown and pressUp are as follows.

(1) When copy non-zero elements to nonZeros vector, starting **from bottom to top** in that column.
(2) When copy merged result of nonZeros back to column, start **from bottom to top** in that column.

In the above code, we need to travel a column once to put non-zeros to a vector, then merge adjacent same-value elements in the vector and copy the result back to the original column, so we need to use additional spaces – a vector – and also traverse each column for at least twice, such approaches are not efficient in both time and space.

The following is an approach to implement pressDown method, traversing each column exactly once without using additional vector.

Key ideas: use two headers, read and write. Use read to filter all non-zeros, when merged, write to the elements at write header.

Pseudocode of pressDown

```
int read = …; //what is the start point of read in pressDown method
int write = …; //what is the next to write position of pressDown method
while (read ….)
{
      //TODO: skip all zeros to get non-zero element

}
```

To merge two adjacent non-zero elements, we'd better to use prev and curr to record the indices of two adjacent non-zero elements. So, instead of using variable, we use prev and curr. So the above code is changed as prev and curr.

```
int prev = numRows;
int curr = numRows – 1;
int write = numRows -1;
while (curr >= 0)
{
     //TODO: skip all zeros to get non-zero element
    While (curr >= 0 && panel[curr][col] == 0)
          Curr--;

    //curr < 0 or panel[curr][col] != 0
    If (curr >= 0)
    {
       //panel[curr][col] != 0

       If (prev == numRows)
          Curr = prev;
      Else {
      }
     }


}
```

Remember panel as a data member to represent board. Let col be the current column index. The relationship between prev and curr are as follows,

(1) prev is not valid
(2) prev is valid
    (a) panel[prev][col] equals panel[curr][col]
    (b) panel[prev][col] does not equal panel[curr][col]