Connect 4 Project

Tong Yi

Implement connect 4 game in `https://www.cbc.ca/kids/games/all/connect-4`. We use colored shapes, not just balls. To indidate a win, we replace the shapes in a trace to similar ones.

Warning:

1. This is copyrighted materials; you are not allowed to upload to the Internet.

2. Our project is different from similar products in Internet. We use shapes instead of balls.

    (a) Ask help only from teaching staff of this course.

    (b) Use solutions from ChatGPT or online tutoring websites like, but not limited to, chegg.com, violates academic integrity and is not allowed.

1 Rules

1. A Board object has several bins.

2. All bins have the same capacity, that is, the maximum number of elements a bin can hold.

3. Rules for moving are listed as follows.

    (a) At any time, number of elements in a bin cannot exceed its capacity.

    (b) The shapes in a bin are filled from bottom to top.

    (c) After adding a shape to a bin, check wether there are at least 4 consecutive shapes in one of the following directions: horizontal, vertical, diagonal, or anti-diagonal.

        i. If so, the game is finished and a winner is declared.
        ii. Otherwise, the game continues until there is no more spot to add a shape.
        iii. If all empty spots are filled, and there is no winner, then the game is tied.

2 Files of the Project

We use Object-oriented Programming approach.

1. Create directory `connect4` to hold codes of the project **<u>if</u>** you have not done so. Said differently, you only need to run the following command once.

    `mkdir connect4`

2. Move to the above directory.

   ```
   cd connect4
   ```

3. Create `Board.hpp` with the following contents. **Warning:** do not write `Board.hpp` as `board.hpp`. C++ is a case-sensitive language.

   `Board.hpp` is the header file of Board class that **declares** data members and operations (aka methods) on those data members.

```cpp
#ifndef BOARD_H
#define BOARD_H
#include <vector>
class Board {
public:
    Board(); //6 bins, each bin holds at most 4 balls
    Board(int numBins, int capacity); //numBins, each bin holds at most capacity
    many shapes
    void display() const;
    int add(int player);
        //Given a player,
        //return which bin the player is added to.

    int winInHorizontal(int bin);
    int winInVertical(int bin);
    int winInDiagonal(int bin);

    int win(int bin); //column must be the most recent ball in that bin

    void play();

//private: //TODO: comment private: for gradescope test purpose only
    int numBins; //number of bins
    int capacity; //maximum number of shapes held in each bin
    //need to compile using -std=c++11
    //if using std::vector<std::vector<int>>, otherwise
    //need to use std::vector<std::vector<int> >
    //with a space after the last > >.

    std::vector<std::vector<int> > grid;
    //If not using c++11, need to have space between the last two > >
    //std::vector<std::vector<int> > grid;
    //The above statement cannot write as
    //std::vector<int> grid(numBins, capacity);
    //which results in a one-dimensional array
    //of numBins elements, each element equals capacity.
};
#endif
```

4. Your task is to implement `Board.cpp`, which **defines** constructors and methods declared in Board.hpp.

   (a) Note that, in `Board.hpp`, data members are declared but not yet initialized. The data members are initialized in constructors.

   (b) Similarly, constructors and methods are declared (have function header) in `Board.hpp` but not defined (no function body).

   (c) **Warning:** do NOT put main function in `Board.cpp`.

3 Data Members in Board.hpp

The details of data members, constructors and methods in Board class of the game are discussed as follows.

```
1   +--+--+--+--+--+--+
2   |  |  |  |  |  |  |
3   +--+--+--+--+--+--+
4   |  |  |  |  |  |  |
5   +--+--+--+--+--+--+
6   |  |  |  |  |  |  |
7   +--+--+--+--+--+--+
8   |  |  |  |  |  |  |
9   +--+--+--+--+--+--+
10    0  1  2  3  4  5
```

1. Data member `numBins` is an integer representing the number of bins. In the above example, `numBins` is 6. One column is a bin.

2. Data member `capacity` is an integer representing the maximum number of element each bin can hold. In the previous example, `capacity` is 4.

3. Data member `grid` of type `std::vector<std::vector<int>>` is a two dimensional array of integers.

   (a) A vector is a one-dimensional array that can grow or shrink. It is a template class, documentation can be found at `https://cplusplus.com/reference/vector/vector/`.

   (b) To use vector, need to include the library.
   `include <vector>`

      i. If you do not use standard namespace std, then need to add `std::` before vector.

      ii. Example: declare a vector as an array of integers with 4 elements. Each element is initialized to be 1.

```cpp
1   #include <iostream>
2   #include <vector>
3   #include <string>
4   //using namespace std;
5
6   int main() {
7       std::vector<int> oneBin = {1, 1, 1, 1};
```

```
8          //oneBin is a vector of integers with elements 1, 1, 1, 1.
9
10      for (int i = 0; i < oneBin.size(); i++)
11          std::cout << oneBin[i] << std::endl;
12      return 0;
13  }
```

(c) Each bin is represented by a vector of integers, similar to a one-dimensional array of integers. A bin may be empty.

(d) In the beginning, data member `grid` has six bins, each bin is empty.

(e) When displaying, map integer 0 to red circle and integer 1 to blue pentagon. More mapping details are shown in `display` method.

You may think `grid` in the previous example has six bins, each bin can hold at most four (same value as capacity) elements.

Note that data member `capacity` does not suppose a limit on the size of the vector. When this capacity is exhausted and more is needed, it is automatically expanded by the container (reallocating it storage space).

However, for this game, we need to make sure that no bin can have more `capacity` elements at any time. When there is `capacity` elments in a bin, we stop adding elements to that bin.

4 Task A: Define constructors in Board.cpp

The purpose of constructor is to initialize data members. A class may have multiple constructors. Different constructors have different parameter lists. Each constructor has exactly the same name as class, no return type, not even void.

4.1 The default constructor `Board()`

The default constructor does not take any parameter. It does the following:

1. Set data members `numBins` to be 6.

2. Set data member `capacity` to be 4.

   Warning: the following code is wrong. `int` before `numBins` means to the varialbe is a local variable for constructor `Board`, but not data member `numBins`.

```
1  Board::Board() {
2      int numBins = 6;
3      ... //omit other code
4  }
```

Correct way:

```
1  Board::Board() {
2      numBins = 6;
3      ... //omit other code
4  }
```

3. You may use the hints from the following code to initialize data member `grid`.

```
//for each shape, do the following:
for (int i = 0; i < ?; i++) { //TODO: fill in ?
    std::vector<int> oneBin;
        //that is, oneBin is an empty vector
        //You may think oneBin as a bin in our application.


    //add the one-dimensional array oneBin to grid
    grid.push_back(??); //TODO: fill in ??
}
```

4.2   A nondefault constructor `Board(int numBins, int capacity)`

1. If given parameter `numBins` is smaller than 5, reset it to be 5.

2. If given parameter `capacity` is smaller than 4, reset it to be 4.

3. Now given parameters are correct, use them to set the corresponding data members. Note that if a formal parameter has exactly the same name as a data member, we need to put `this->` before the data member, where `this` is a pointer to the current object.

You may notice that there are a lot of common codes among those constructors. A better way is to define `Board(int numBins, int capacity)`. Then use constructor delegate to define `Board()`.

No need to define destructor in this project since we did not dynamically allocate memories for data members.

4.3   Finish Task A

1. Define constructors in `Board.cpp`.

```
#include "Board.hpp"
#include <iostream> //cout
#include <iomanip> //setw
#include <algorithm> //swap

//TODO: fill in ? and ?? in the parentheses.
//Hint: what are the values of numBins and capacity for a default Board object?
//Question: after calling Board(?, ??) to create a Board object with
//? means number of bins,
//each bin holds at most ?? elements,
Board::Board() : Board(?, ??) {
    //No more code is needed
}

Board::Board(int numBins, int capacity) {
    //TODO: If given parameter numBins is smaller than 5,
```

```
17        //reset it to be 5.
18
19
20        //TODO: If given parameter capacity is smaller than 4,
21        //reset it to be 4.
22
23
24        //Now given parameters are correct,
25        //use them to set the corresponding data members.
26        //Note that if a formal parameter has exactly
27        //the same name as a data member,
28        //we need to put this-> before the data member,
29        //where this is a pointer to the current object.
30
31        //TODO: use formal parameter numBins to set data member numBins
32
33
34        //TODO: use formal parameter capacity to set data member capacity
35
36
37
38        //TODO: initialize data member grid
39
40        //for i in [0, numBins):
41        //begin
42        //    instantiate an empty bin, call it oneBin
43        //    push oneBin back to data members grid
44        //end
45
46
47
48
49
50 }
```

2. Implement method `display`.

   See the following hints.

   We provide a non-member function `print`.

```
1 //not a method from Board class,
2 //hence cannot access data member capacity directly,
3 //need to pass capacity as a parameter
4 void print(int numBins) {
5     //No need to print spaces before the first +
6     //std::cout << "   ";
7     std::cout << "+";
```

```
8        for (int i = 0; i < numBins; i++)
9            std::cout << "--+";
10
11       std::cout << std::endl;
12   }
```

Here is a skeleton of method `display`.

```
1        //map 0 to a red \033[31m ball \u2b24
2        //map 1 to a blue \033[34m pentagon \u2b1f
3        //map 2 to a red double circle \033[31m\u25c9
4        //map 3 to a blue empty pentagon \033[31m\u2b54
5        //For more shapes, see https://jrgraphix.net/r/Unicode/25A0-25FF
6        std::string mapping[] = {"\033[31m\u2b24\033[0m", "\033[34m\u2b1f\033[0m", "
         \033[31m\u25c9\033[0m", "\033[34m\u2b54\033[0m"}; //\033[32m is green color
7
8        print(numBins);
9        for (int j = capacity-1; j >= 0; j--) {
10           //Your codes goes here.
11           //Instead of printing grid[i][j],
12           //where i is bin index,
13           //you may consider using mapping[grid[i][j]].
14
15
16
17       }
18
19       //TODO: print labels
20
21   }
```

3. Implement method `add`.

```
1    int Board::add(int player) {
2        //TODO:
3        //(1) Given player,
4        //    choose a bin index that is valid,
5        //    ie, in [0, numBins),
6        //    and is not full.
7        //(2) push back this player id to that bin.
8        //(3) return the bin index chosen.
9
10
11
12
13   }
```

7

4. Test codes locally.

   (a) Comment private: line in `Board.hpp` as `//private:`. This is for debug purpose.

   (b) Edit main.cpp as follows. This file can be downloaded from `https://onlinegdb.com/aedvljUjk`.

```cpp
#include <iostream>
#include <vector>
#include "Board.hpp"
//g++ -std=c++11 Board.cpp main.cpp -o test
//test default constructor using
//./test A or ./test 'A'
//./test B or ./test 'B'
//...
//./test H or ./test 'H'


int main(int argc, const char *argv[]) {
    if (argc != 2) {
        std::cout << "Need 'A'-'C' in parameters" << std::endl;
        return -1;
    }

    //unit-testing for constructors and the destructor
    char type = *argv[1];
    std::string prompt;
    Board *game;
    int** arr;

    if (type == 'A') {
        prompt = "default constructor,";
        game = new Board;

//Sample output:
//After default constructor, data member numBins is 6
//After default constructor, data member capacity is 4
//number of elements of bin 0 is: 0
//number of elements of bin 1 is: 0
//number of elements of bin 2 is: 0
//number of elements of bin 3 is: 0
//number of elements of bin 4 is: 0
//number of elements of bin 5 is: 0
    }
    else if (type == 'B') {
            prompt = "Board game(7, 5);";
            game = new Board(7, 5);

//Sample output:
```

```
//After Board game(7, 5); data member numBins is 7
//After Board game(7, 5); data member capacity is 5
//number of elements of bin 0 is: 0
//number of elements of bin 1 is: 0
//number of elements of bin 2 is: 0
//number of elements of bin 3 is: 0
//number of elements of bin 4 is: 0
//number of elements of bin 5 is: 0
//number of elements of bin 6 is: 0
    }
    else if (type == 'C') {
            prompt = "Board game(5, 1);";
            game = new Board(5, 1);

//sample output:
//After Board game(5, 1); data member numBins is 5
//After Board game(5, 1); data member capacity is 4
//number of elements of bin 0 is: 0
//number of elements of bin 1 is: 0
//number of elements of bin 2 is: 0
//number of elements of bin 3 is: 0
//number of elements of bin 4 is: 0
    }
    else if (type == 'D') {
            game = new Board;

            game->grid[0].push_back(0);
            game->grid[0].push_back(1);
            game->grid[1].push_back(0);
            game->grid[1].push_back(1);
            game->grid[2].push_back(0);
            game->grid[3].push_back(1);
            game->display();

//sample output:
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|     |  |  |  |  |
//+--+--+--+--+--+--+
//|     |  |     |  |  |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
    }
```

```cpp
    else if (type == 'E') {
            game = new Board;

            game->grid[0].push_back(2);
            game->grid[0].push_back(1);
            game->grid[1].push_back(2);
            game->grid[1].push_back(1);
            game->grid[2].push_back(2);
            game->grid[2].push_back(1);
            game->grid[3].push_back(2);
            game->display();
//sample output:
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |   |  |  |  |
//+--+--+--+--+--+--+
//|    |    |   |   |  |  |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
    }
    else if (type == 'F') {
            game = new Board;

            game->grid[0].push_back(0);
            game->grid[1].push_back(3);
            game->grid[0].push_back(0);
            game->grid[1].push_back(3);
            game->grid[0].push_back(0);
            game->grid[1].push_back(3);
            game->grid[3].push_back(0);
            game->grid[1].push_back(3);
            game->display();

//sample output:
//+--+--+--+--+--+--+
//|  |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |    |  |  |
//+--+--+--+--+--+--+
```

```
135   //  0  1  2  3  4  5
136       }
137   else if (type == 'G') {
138           game = new Board;
139           int bin = game->add(0); //choose a bin for player with id 0
140           int bin2 = game->add(1); //choose a bin for player with id 1
141
142           std::cout << "The layout of the bins are as follows." << std::endl
      ;
143           for (int i = 0; i < game->numBins; i++) {
144               if (game->grid[i].size() == 0)
145                   std::cout << "empty" << std::endl;
146               else {
147                       for (int j = 0; j < game->grid[i].size(); j++)
148                           std::cout << game->grid[i][j] << " ";
149                       std::cout << std::endl;
150               }
151           }
152
153   //sample output:
154   //Enter a bin index in [0, 6) that is not full: -1
155   //invalid bin index, needs to be in [0, 6)
156   //Re-enter a bin index in [0, 6) that is not full: 7
157   //invalid bin index, needs to be in [0, 6)
158   //Re-enter a bin index in [0, 6) that is not full: 0
159   //Enter a bin index in [0, 6) that is not full: -2
160   //invalid bin index, needs to be in [0, 6)
161   //Re-enter a bin index in [0, 6) that is not full: 10
162   //invalid bin index, needs to be in [0, 6)
163   //Re-enter a bin index in [0, 6) that is not full: 1
164   //The layout of the bins are as follows.
165   //0
166   //1
167   //empty
168   //empty
169   //empty
170   //empty
171       }
172   else if (type == 'H') {
173           //When a bin is full, cannot add more element to it.
174           game = new Board;
175           game->grid[0].push_back(0);
176           game->grid[0].push_back(1);
177           game->grid[0].push_back(0);
178           game->grid[0].push_back(1);
179           int bin = game->add(0); //choose a bin for player with id 0
```

```cpp
                    std::cout << "The layout of the bins are as follows." << std::endl
    ;
                for (int i = 0; i < game->numBins; i++) {
                    if (game->grid[i].size() == 0)
                        std::cout << "empty" << std::endl;
                    else {
                            for (int j = 0; j < game->grid[i].size(); j++)
                                std::cout << game->grid[i][j] << " ";
                            std::cout << std::endl;
                    }
                }
//sample output:
//Enter a bin index in [0, 6) that is not full: 0
//the bin is full
//Re-enter a bin index in [0, 6) that is not full: 1
//The layout of the bins are as follows.
//0 1 0 1
//0
//empty
//empty
//empty
//empty
        }

    //When type is 'A' - 'C', work on constructors
    if (type == 'A' || type == 'B' || type == 'C') {
        std::cout << "After " << prompt
                << " data member numBins is " << game->numBins << std::endl;
        std::cout << "After " << prompt
                << " data member capacity is " << game->capacity << std::endl;

        for (int i = 0; i < game->numBins; i++) {
            std::cout << "number of elements of bin "
                << i << " is: "
                << game->grid[i].size() << std::endl;

        }
    }

    delete game;
    game = nullptr;

    return 0;
}
```

(c) Run the following command to compile main.cpp and Board.cpp.

```
g++ -std=c++11 main.cpp Board.cpp -o test
```

(d) If there is no compilation errors, run the following command.

```
./test A
```

(e) You should be able see something like the following.

```
1  After default constructor, data member numBins is 6
2  After default constructor, data member capacity is 4
3  number of elements of bin 0 is: 0
4  number of elements of bin 1 is: 0
5  number of elements of bin 2 is: 0
6  number of elements of bin 3 is: 0
7  number of elements of bin 4 is: 0
8  number of elements of bin 5 is: 0
```

(f) Test non-default construtor `Board(int numBins, int capacity)` by using

```
./test B
```

You should see the following output.

```
1  After Board game(7, 5); data member numBins is 7
2  After Board game(7, 5); data member capacity is 5
3  number of elements of bin 0 is: 0
4  number of elements of bin 1 is: 0
5  number of elements of bin 2 is: 0
6  number of elements of bin 3 is: 0
7  number of elements of bin 4 is: 0
8  number of elements of bin 5 is: 0
9  number of elements of bin 6 is: 0
```

5. Run ./test C, we get the following output.

```
1  After Board game(5, 1); data member numBins is 5
2  After Board game(5, 1); data member capacity is 4
3  number of elements of bin 0 is: 0
4  number of elements of bin 1 is: 0
5  number of elements of bin 2 is: 0
6  number of elements of bin 3 is: 0
7  number of elements of bin 4 is: 0
```

6. Run ./test D. Get the following output.

7. Run `./test` E. Get the following output.

8. Run `./test` F. Get the following output.

9. Or you can test the code in `https://www.onlinegdb.com/online_c++_compiler`.

   Upload main.cpp, Board.hpp (comment private: line) and Board.cpp to onlinegdb. In the textbox right to **Command line arguments:**, enter A to H.

10. If the code runs correctly in a local computer, upload `Board.cpp` to gradescope.

11. Again, do not add `main` function in `Board.cpp`.

5 Task B: define winHorizontal, winVertical, winDiagonal, and win methods

After a player puts a shape to a bin using `add` method in Task A, an id (either 0 or 1) representing that player is pushed back to the bin. There is a chance the player could win.

5.1 A win or not

To find out whether there is an actual win or not, define method `winInHorizontal`, given a bin, check whether its top element (including itself) has 4 or more same-value neighbors in horizontal direction. Simiarly, methods `winInVertical` and `winInDiagonal` (including diagonal and anti-diagonal) are defined.

5.2 With a win, save the locations of four or more consecutive same-value elements in the corresponding direction

Furthermore, once a win occurs, we would like to mark out those 4 or more consecutive elements in the correspondin direction. To do so, we need to record the location of an element.

Use type `Coord` to save the location of an element, where `bin` is the bin index and `idx` is the index of the element in `bin`.

```
1  struct Coord {
2      int bin; //bin index of data member grid
3      int idx; //index inside the bin
4  };
```

Some students ask the differences between `struct` and `class` in C++.

1. Both `struct` and `class` define a type.

2. `struct` also works in C. Unlike an array, which is a collection of same-type data residing in consecutive memory, a `struct` consists of members of different types, but without methods or constructor (this part changes in C++), while `class` is defined in C++.

3. By default, members in `struct` are public.

4. By default, members in `class` are private.

5. According to `https://stackoverflow.com/questions/54585/when-should-you-use-a-class-vs-a-str` we can do the following.

    (a) Use struct for plain-old-data structures without any class-like features.
    (b) Use class when you make use of features such as private or protected members, non-default constructors and operators.

Put the locations of those elements in a vector. If the size of that vector is larger than or equal to 4, increase the value of those elements by 2. In this way, those four or more consecutive elements in the path can be marked out, so we can see clearly why a player wins.

We illustrate a run of method `winInHorizontal`.

1. In the beginning, player red (with id 0) puts 0, which maps to a red circle, into bin 5.
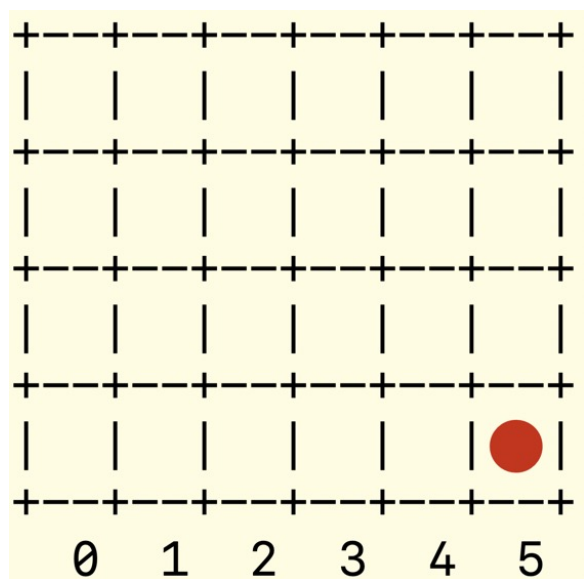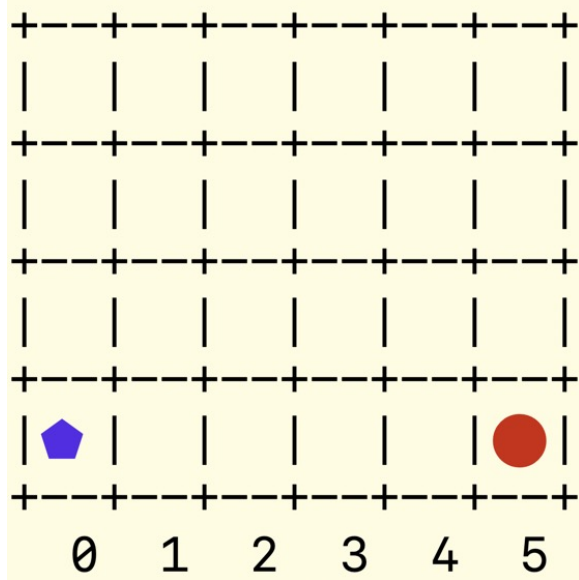

    The above process is achieved by the following statements, taken from `main.cpp` to test methods in Task B.
```

Note that we do not call methods to put elements to bins in case other methods are not properly defined. We just concentrate on testing on `winInHorizontal` method.

```
1  game = new Board; //default constructor
2  game->grid[5].push_back(0);
3  //game->display(); //optional,
```

2. Next, player blue (with id 1) puts a 1, which maps to a blue pentagon, into bin 0.

This is achieved by adding the following statements to the above codes.

```
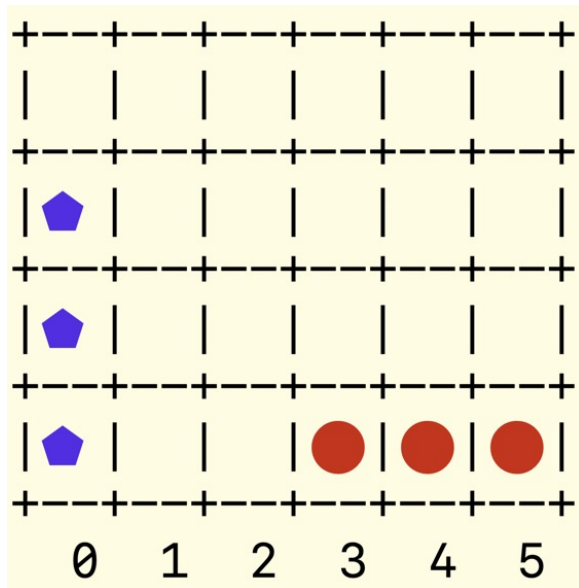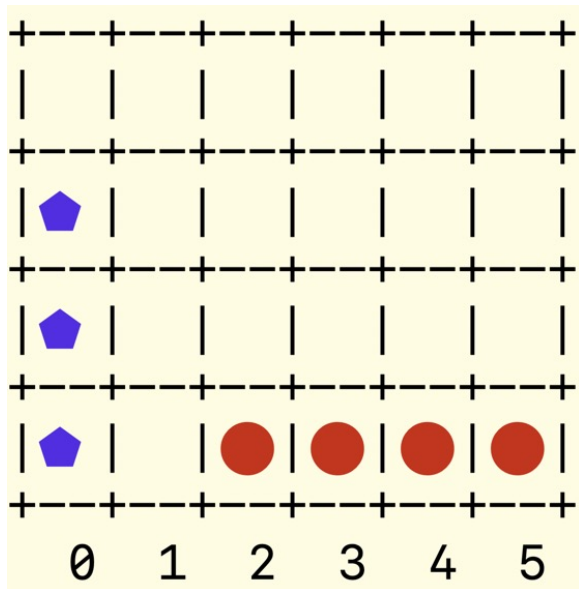1  game->grid[0].push_back(1);
```

3. Fastforward, the layout of bins are as follows after player 0 chooses bin 4, player 1 choose bin 0, then player 0 chooses bin 3, player 1 chooses bin 0.

```
1  game->grid[4].push_back(0); //player 0 chooses bin 4
2  game->grid[0].push_back(1); //player 1 chooses bin 0
3  game->grid[3].push_back(0); //player 0 chooses bin 3
4  game->grid[0].push_back(1); //player 1 chooses bin 0
```

4. Now is the turn of player 0. If bin 2 is chosen, then four 0s (each 0 is mapped to a red circle) are aligned consecutively in horizontal direction.


```
1  game->grid[2].push_back(0);
```

5. Test `winInHorizontal` method in bin 2, the most recent bin with a new element added.

```
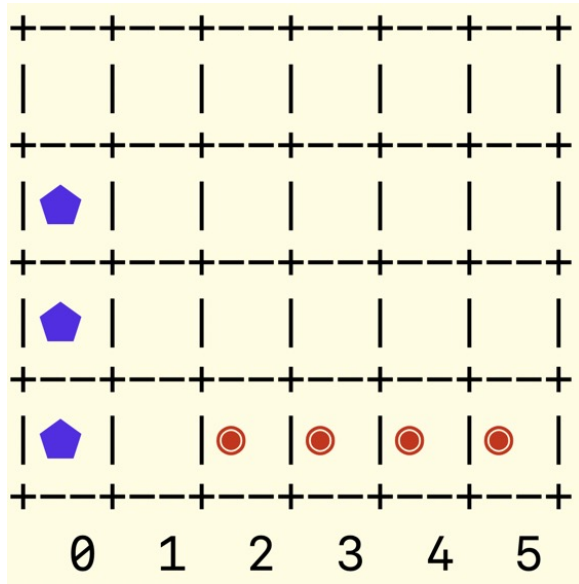1  player = game->winInHorizontal(2);
```

The return should be 0, which means player 0 wins in horizontal direction.

6. Now call `winInHorizontal` method on bin 2. If there is a winner, return the corresponding id, otherwise, return -1.

```
1  player = game->winInHorizontal(2);
```

7. To show users why player 0 wins, we display the corresponding elements in slightly different shapes, from the original red circles to double red circles. To do so, we add 2 to the original value 0. By the mapping in display method, label 2 is mapped to a double red circle.

Here is expected result after test option 1 in the above main function, where each column represents a bin. Display the elements for each bin from top to bottom, if there is no element, use a space.

The first bin has three blue pentagons, represented by id 1. In bins indexed at 2, 3, 4, 5, each has one element 2, which implies that player 0 wins.

```
1  player 0 wins in horizontal
2   , , , , , ,
3  1, , , , , ,
4  1, , , , , ,
5  1, ,2,2,2,2,
```

5.3   Method `winInHorizontal`

In short, method `int winInHorizontal(int bin)` does the following.

1. Find out the player id in the top of `bin`. (Hint: what is the last index of `bin`, a vector of integers?

2. Count the number of consecutive same-value as the above player id elements in horizontal direction.

    (a) Record the locations, that is, bin index and the index in the bin, to a vector variable.

3. If the number of consecutive same-value elements in horizontal direction is at least four, set the player id as the winner.

4. Increase the corresponding elements contributed to the win by 2. That is, if the original value is 0, the new value is 2. If the original value is 1, the new value would be 3.

Here is a skeleton of method `winInHorizontal`.

19

```cpp
int Board::winInHorizontal(int bin) {
    //Find out the last index of bin.

    //Find out the player residing at that last index of bin.
    //Save in an int variable called player.

    //Declare candidates as a vector of Coord.

    //candidates saves the locations of
    //all horizontal neighbors of the top elements of the given parameter bin
    //that share the same value of that top element,
    //which is saved in player.


    //Who can be the first element of candidates?


    //count consecutive elements equaling player to the left of grid[bin][idx]. Save
    their locations to candidates.

    //count consecutive elements equaling player to the right of grid[bin][idx]. Save
    their locations to candidates.

    if (count >= 4) {
        //change all elements in candidates to e player +2

        return player;
    }

    //Return -1 when no winner in horizontal direction
}
```

Define `winInVertical` method.

Define `winInDiagonal` method, need to consider both diagonal and anti-diagonal directions.

Define `win` method, which calls `winInHorizontal`, `winInVertical`, and `winInDiagonal` methods, if one of them return a value other than -1, then a winner is found, return the corresponding player id, otherwise, return -1.

5.4 Test code locally

1. Create a subdirectory called `taskB` under `connect4`.

2. In `taskB`, copy `Board.cpp` from Task A.

    (a) Define `winInHorizontal`, `winInVertical`, `winInDiagonal`, and `win` methods in `Board.cpp`.

3. Download the following `main.cpp` and `Board.hpp` from `https://onlinegdb.com/7buWVilKs` to subdirectory `taskB`.

   (a) Run command

   ```
   g++ -std=c++11 Board.cpp main.cpp -o test
   ```

   Test methods in Task B using the following commands. Compare the output with sample output in each test case of `main.cpp`.

   ```
   ./test 1
   ```

   There are 9 cases to test. Change 1 to 2, 3, ..., 9 if necessary.

   If codes run fine, upload `Board.cpp` to gradescope.

   Warning: some unicode symbols are not shown in the following codes, download from the previous onlinegdb link or brightspace.

   ```cpp
   #include <iostream>
   #include <vector>
   #include "Board.hpp"
   //g++ -std=c++11 Board.cpp main.cpp -o test
   //test different methods using
   //./test 1 or ./test '1'
   //./test 2 or ./test '2'
   //...
   //./test 9 or ./test '9'


   int main(int argc, const char *argv[]) {
       if (argc != 2) {
           std::cout << "Need 'A'-'C' in parameters" << std::endl;
           return -1;
       }

       //unit-testing for constructors and the destructor
       char type = *argv[1];
       std::string prompt;
       Board *game;
       int player = -2; //a not-exist value

       if (type == '1') {
           //test player 0 wins in winInHorizontal
           game = new Board; //default constructor
           game->grid[5].push_back(0);
           //game->display();
           game->grid[0].push_back(1);
           //game->display();
           game->grid[4].push_back(0);
   ```

```
32          //game->display();
33          game->grid[0].push_back(1);
34          //game->display();
35          game->grid[3].push_back(0);
36          //game->display();
37          game->grid[0].push_back(1);
38          //game->display();
39          game->grid[2].push_back(0);
40          //game->display();
41
42          player = game->winInHorizontal(2);
43          prompt = " in horizontal";
44
45          //game->display();
46  //Sample output:
47  //player 0 wins in horizontal
48  //  , , , , , ,
49  //1, , , , , ,
50  //1, , , , , ,
51  //1, ,2,2,2,2,
52  //+--+--+--+--+--+--+
53  //|  |  |  |  |  |  |
54  //+--+--+--+--+--+--+
55  //|     |  |  |  |  |  |
56  //+--+--+--+--+--+--+
57  //|     |  |  |  |  |  |
58  //+--+--+--+--+--+--+
59  //|     |  |  |  |     |     |
60  //+--+--+--+--+--+--+
61  //  0  1  2  3  4  5
62      }
63      else if (type == '2') {
64          //test no one wins in winInHorizontal
65          game = new Board; //default constructor
66          game->grid[5].push_back(0);
67          game->grid[0].push_back(1);
68          game->grid[4].push_back(0);
69          game->grid[0].push_back(1);
70          game->grid[3].push_back(0);
71          game->grid[0].push_back(1);
72
73          player = game->winInHorizontal(0);
74          prompt = " in horizontal";
75          //game->display();
76  //Sample output:
77  //no player wins in horizontal
```

```
//  , , , , , ,
//1, , , , , ,
//1, , , , , ,
//1, , ,0,0,0,
//Visual result
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|    |  |  |  |  |  |
//+--+--+--+--+--+--+
//|    |  |  |  |  |  |
//+--+--+--+--+--+--+
//|    |  |  |  |    |    |
//+--+--+--+--+--+--+
//   0  1  2  3  4  5
    }
    else if (type == '3') {
        //win in vertical
        game = new Board;

        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[3].push_back(0);
        game->grid[1].push_back(1);
        //game->display();

        player = game->winInVertical(1);
        prompt = " in vertical";

//sample output:
//player 1 wins in vertical
//  ,3, , , , ,
//0,3, , , , ,
//0,3, , , , ,
//0,3, ,0, , ,
//+--+--+--+--+--+--+
//|  |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
```

```
//|    |    |  |    |  |  |
//+--+--+--+--+--+--+
//   0  1  2  3  4  5
    }
    else if (type == '4') {
        //no win in vertical
        game = new Board;

        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        //game->display();

        player = game->winInVertical(1);
        prompt = " in vertical";
//Sample output:
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//|    |    |  |  |  |  |
//+--+--+--+--+--+--+
//   0  1  2  3  4  5
//no player wins in vertical
//  , , , , , ,
//0,1, , , , ,
//0,1, , , , ,
//0,1, , , , ,
    }
    else if (type == '5') {
        //win in diagonal
        game = new Board;
        game->grid[0].push_back(0);
        game->grid[1].push_back(1);
        game->grid[1].push_back(0);
        game->grid[2].push_back(1);
        game->grid[3].push_back(0);
        game->grid[2].push_back(1);
        game->grid[2].push_back(0);
        game->grid[3].push_back(1);
```

```cpp
        game->grid[3].push_back(0);
        game->grid[4].push_back(1);
        game->grid[3].push_back(0);

        player = game->winInDiagonal(0);
        prompt = " in diagonal";

        //game->display();

//Sample output:
//player 0 wins in diagonal
// , , ,2, , ,
// , ,2,0, , ,
// ,2,1,1, , ,
//2,1,1,0,1, ,
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|  |  |  |   |  |  |
//+--+--+--+--+--+--+
//|  |  |   |   |  |  |
//+--+--+--+--+--+--+
//|   |  |  |  |   |  |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
        }
        else if (type == '6') {
                //win in anti-diagonal
            game = new Board;
            game->grid[0].push_back(0);
            game->grid[1].push_back(1);
            game->grid[2].push_back(0);
            game->grid[3].push_back(1);
            game->grid[0].push_back(0);
            game->grid[2].push_back(1);
            game->grid[0].push_back(0);
            game->grid[0].push_back(1);
            game->grid[1].push_back(0);
            game->grid[1].push_back(1);

            player = game->winInDiagonal(1);
            prompt = " in anti-diagonal";

            //game->display();

//sample output:
```

```cpp
//+--+--+--+--+--+--+
//|   |  |  |  |  |  |
//+--+--+--+--+--+--+
//|   |  |  |  |  |  |
//+--+--+--+--+--+--+
//|   |  |  |  |  |  |
//+--+--+--+--+--+--+
//|   |  |  |  |  |  |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
//player 1 wins in anti-diagonal
//3, , , , , ,
//0,3, , , , ,
//0,0,3, , , ,
//0,1,0,3, , ,
        }
    else if (type == '7') {
            //no win, a tie
            //msg = '0 1  0 2 0  0 3 1 2 1 1 2 3  3 5  3 5 5 4 4 4 5  4  2\n'
            game = new Board;
            game->grid[0].push_back(0);
            game->grid[1].push_back(1);
            game->grid[0].push_back(0);
            game->grid[2].push_back(1);
            game->grid[0].push_back(0);
            game->grid[0].push_back(1);
            game->grid[3].push_back(0);
            game->grid[1].push_back(1);
            game->grid[2].push_back(0);
            game->grid[1].push_back(1);
            game->grid[1].push_back(0);
            game->grid[2].push_back(1);
            game->grid[3].push_back(0);
            game->grid[3].push_back(1);
            game->grid[5].push_back(0);
            game->grid[3].push_back(1);
            game->grid[5].push_back(0);
            game->grid[5].push_back(1);
            game->grid[4].push_back(0);
            game->grid[4].push_back(1);
            game->grid[4].push_back(0);
            game->grid[5].push_back(1);
            game->grid[4].push_back(0);
            game->grid[2].push_back(1);

            player = game->win(1);
```

```cpp
            //game->display();
//sample output:
//+--+--+--+--+--+--+
//|   |   |   |   |   |   |
//+--+--+--+--+--+--+
//|   |   |   |   |   |   |
//+--+--+--+--+--+--+
//|   |   |   |   |   |   |
//+--+--+--+--+--+--+
//|   |   |   |   |   |   |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
//no player wins
//1,0,1,1,0,1,
//0,1,1,1,0,1,
//0,1,0,0,1,0,
//0,1,1,0,0,0,
        }
        else if (type == '8') {
            //msg = '0 1 0 2 1 3 0 4\n'
            game = new Board;
            game->grid[0].push_back(0);
            game->grid[1].push_back(1);
            game->grid[0].push_back(0);
            game->grid[2].push_back(1);
            game->grid[1].push_back(0);
            game->grid[3].push_back(1);
            game->grid[0].push_back(0);
            game->grid[4].push_back(1);

            player = game->win(4); //4 is the most recent bin with balls added

            //game->display();
//sample output:
//+--+--+--+--+--+--+
//|  |  |  |  |  |  |
//+--+--+--+--+--+--+
//|   |  |  |  |  |  |
//+--+--+--+--+--+--+
//|   |   |  |  |  |  |
//+--+--+--+--+--+--+
//|   |   |   |   |   |  |
//+--+--+--+--+--+--+
//  0  1  2  3  4  5
//player 1 wins
```

```
308  // , , , , , ,
309  //0, , , , , ,
310  //0,0, , , , ,
311  //0,3,3,3,3, ,
312      }
313      else if (type == '9') {
314          //msg = '1 0 1 2 1 3 1\n'
315          game = new Board;
316          game->grid[1].push_back(0);
317          game->grid[0].push_back(1);
318          game->grid[1].push_back(0);
319          game->grid[2].push_back(1);
320          game->grid[1].push_back(0);
321          game->grid[3].push_back(1);
322          game->grid[1].push_back(0);

324          player = game->win(1); //1 is the most recent bin with balls added

326  //        game->display();
327  //sample output:
328  //+--+--+--+--+--+--+
329  //|  |   |  |  |  |  |
330  //+--+--+--+--+--+--+
331  //|  |   |  |  |  |  |
332  //+--+--+--+--+--+--+
333  //|  |   |  |  |  |  |
334  //+--+--+--+--+--+--+
335  //|    |   |  |  |  |  |
336  //+--+--+--+--+--+--+
337  //  0  1  2  3  4  5
338  //player 0 wins
339  // ,2, , , , ,
340  // ,2, , , , ,
341  // ,2, , , , ,
342  //1,2,1,1, , ,
343      }

345      if (player != -1)
346          std::cout << "player " << player << " wins";
347      else std::cout << "no player wins";

349      std::cout << prompt << std::endl;

351      for (int j = game->capacity-1; j>=0; j--) {
352              for (int i = 0; i < game->numBins; i++) {
353                      if (j < game->grid[i].size())
```

28

```cpp
                    std::cout << game->grid[i][j] << ",";
                else std::cout << " ,";
            }
            std::cout << std::endl;
        }

    delete game;
    game = nullptr;

    return 0;
}
```