

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 indicate 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 whether 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 **if** 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.

```
1 #ifndef BOARD_H
2 #define BOARD_H
3 #include <vector>
4 class Board {
5 public:
6     Board(); //6 bins, each bin holds at most 4 balls
7     Board(int numBins, int capacity); //numBins, each bin holds at most capacity
8     many shapes
9     void display() const;
10    int add(int player);
11        //Given a player,
12        //return which bin the player is added to.
13
14    int winInHorizontal(int bin);
15    int winInVertical(int bin);
16    int winInDiagonal(int bin);
17
18    int win(int bin); //column must be the most recent ball in that bin
19
20    void play();
21
22    //private: //TODO: comment private: for gradescope test purpose only
23    int numBins; //number of bins
24    int capacity; //maximum number of shapes held in each bin
25    //need to compile using -std=c++11
26    //if using std::vector<std::vector<int>>, otherwise
27    //need to use std::vector<std::vector<int> >
28    //with a space after the last > >.
29
30    std::vector<std::vector<int> > grid;
31    //If not using c++11, need to have space between the last two > >
32    //std::vector<std::vector<int> > grid;
33    //The above statement cannot write as
34    //std::vector<int> grid(numBins, capacity);
35    //which results in a one-dimensional array
36    //of numBins elements, each element equals capacity.
37 };
38 #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.

```
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 its 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` elements 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 variable 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`.

```
1 //for each shape, do the following:
2 for (int i = 0; i < ?; i++) { //TODO: fill in ?
3     std::vector<int> oneBin;
4         //that is, oneBin is an empty vector
5         //You may think oneBin as a bin in our application.
6
7
8     //add the one-dimensional array oneBin to grid
9     grid.push_back(??); //TODO: fill in ??
10 }
```

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`.

```
1 #include "Board.hpp"
2 #include <iostream> //cout
3 #include <iomanip> //setw
4 #include <algorithm> //swap
5
6 //TODO: fill in ? and ?? in the parentheses.
7 //Hint: what are the values of numBins and capacity for a default Board object?
8 //Question: after calling Board(?, ??) to create a Board object with
9 //? means number of bins,
10 //each bin holds at most ?? elements,
11 Board::Board() : Board(?, ??) {
12     //No more code is needed
13 }
14
15 Board::Board(int numBins, int capacity) {
16     //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 ball \033[31m\u2b24,
2     //where \033[31m is red and \u2b24 is a ball
3     //map 1 to a blue pentagon \033[34m\u2b1f
4     //where \033[34m is blue and \u2b1f is a pentagon
5     //map 2 to a red double circle \033[31m\u25c9
6     //where \u25c9 is a double circle
7     //map 3 to a blue empty pentagon \033[31m\u2b54
8     //where \u2b54 is a pentagon with edges only
9     //\033[0m is black color, the default color
10    //For more shapes, see https://jrgraphix.net/r/Unicode/25A0-25FF
11    std::string mapping[] = {"\033[31m\u2b24\033[0m", "\033[34m\u2b1f\033[0m", "
12    \033[31m\u25c9\033[0m", "\033[34m\u2b54\033[0m"}; //\033[32m is green color
13
14    print(numBins);
15    for (int j = capacity-1; j >= 0; j--) {
16        //Your codes goes here.
17        //Instead of printing grid[i][j],
18        //where i is bin index,
19        //you may consider using mapping[grid[i][j]].
20
21    }
22
23    //TODO: print labels
24
25 }
26

```

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 }

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>.

```
1  #include <iostream>
2  #include <vector>
3  #include "Board.hpp"
4  //g++ -std=c++11 Board.cpp main.cpp -o test
5  //test default constructor using
6  ///./test A or ./test 'A'
7  ///./test B or ./test 'B'
8  ///...
9  ///./test H or ./test 'H'
10
11
12 int main(int argc, const char *argv[]) {
13     if (argc != 2) {
14         std::cout << "Need 'A'-'C' in parameters" << std::endl;
15         return -1;
16     }
17
18     //unit-testing for constructors and the destructor
19     char type = *argv[1];
20     std::string prompt;
21     Board *game;
22     int** arr;
23
24     if (type == 'A') {
25         prompt = "default constructor,";
26         game = new Board;
27
28         //Sample output:
29         //After default constructor, data member numBins is 6
30         //After default constructor, data member capacity is 4
31         //number of elements of bin 0 is: 0
32         //number of elements of bin 1 is: 0
33         //number of elements of bin 2 is: 0
34         //number of elements of bin 3 is: 0
35         //number of elements of bin 4 is: 0
36         //number of elements of bin 5 is: 0
37     }
```



```

38     else if (type == 'B') {
39         prompt = "Board game(7, 5);";
40         game = new Board(7, 5);
41
42         //Sample output:
43         //After Board game(7, 5); data member numBins is 7
44         //After Board game(7, 5); data member capacity is 5
45         //number of elements of bin 0 is: 0
46         //number of elements of bin 1 is: 0
47         //number of elements of bin 2 is: 0
48         //number of elements of bin 3 is: 0
49         //number of elements of bin 4 is: 0
50         //number of elements of bin 5 is: 0
51         //number of elements of bin 6 is: 0
52     }
53     else if (type == 'C') {
54         prompt = "Board game(5, 1);";
55         game = new Board(5, 1);
56
57         //sample output:
58         //After Board game(5, 1); data member numBins is 5
59         //After Board game(5, 1); data member capacity is 4
60         //number of elements of bin 0 is: 0
61         //number of elements of bin 1 is: 0
62         //number of elements of bin 2 is: 0
63         //number of elements of bin 3 is: 0
64         //number of elements of bin 4 is: 0
65     }
66     else if (type == 'D') {
67         game = new Board;
68
69         game->grid[0].push_back(0);
70         game->grid[0].push_back(1);
71         game->grid[1].push_back(0);
72         game->grid[1].push_back(1);
73         game->grid[2].push_back(0);
74         game->grid[3].push_back(1);
75         game->display();
76
77         //sample output:
78         //+---+---+---+---+---+
79         //|  |  |  |  |  |  |
80         //+---+---+---+---+---+
81         //|  |  |  |  |  |  |
82         //+---+---+---+---+---+
83         //|  |  |  |  |  |  |

```

```

84 //+---+---+---+---+
85 //|   |   |   |   |   |
86 //+---+---+---+---+
87 // 0  1  2  3  4  5
88 }
89     else if (type == 'E') {
90         game = new Board;
91
92         game->grid[0].push_back(2);
93         game->grid[0].push_back(1);
94         game->grid[1].push_back(2);
95         game->grid[1].push_back(1);
96         game->grid[2].push_back(2);
97         game->grid[2].push_back(1);
98         game->grid[3].push_back(2);
99         game->display();
100 //sample output:
101 //+---+---+---+---+
102 //|   |   |   |   |   |
103 //+---+---+---+---+
104 //|   |   |   |   |   |
105 //+---+---+---+---+
106 //|   |   |   |   |   |
107 //+---+---+---+---+
108 //|   |   |   |   |   |
109 //+---+---+---+---+
110 // 0  1  2  3  4  5
111 }
112     else if (type == 'F') {
113         game = new Board;
114
115         game->grid[0].push_back(0);
116         game->grid[1].push_back(3);
117         game->grid[0].push_back(0);
118         game->grid[1].push_back(3);
119         game->grid[0].push_back(0);
120         game->grid[1].push_back(3);
121         game->grid[3].push_back(0);
122         game->grid[1].push_back(3);
123         game->display();
124
125 //sample output:
126 //+---+---+---+---+
127 //|   |   |   |   |   |
128 //+---+---+---+---+
129 //|   |   |   |   |   |

```

```

130 //+---+---+---+---+---+
131 //|   |   |   |   |   |
132 //+---+---+---+---+---+
133 //|   |   |   |   |   |
134 //+---+---+---+---+---+
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         ;
144         for (int i = 0; i < game->numBins; i++) {
145             if (game->grid[i].size() == 0)
146                 std::cout << "empty" << std::endl;
147             else {
148                 for (int j = 0; j < game->grid[i].size(); j++)
149                     std::cout << game->grid[i][j] << " ";
150                 std::cout << std::endl;
151             }
152         }
153
154         //sample output:
155         //Enter a bin index in [0, 6) that is not full: -1
156         //invalid bin index, needs to be in [0, 6)
157         //Re-enter a bin index in [0, 6) that is not full: 7
158         //invalid bin index, needs to be in [0, 6)
159         //Re-enter a bin index in [0, 6) that is not full: 0
160         //Enter a bin index in [0, 6) that is not full: -2
161         //invalid bin index, needs to be in [0, 6)
162         //Re-enter a bin index in [0, 6) that is not full: 10
163         //invalid bin index, needs to be in [0, 6)
164         //Re-enter a bin index in [0, 6) that is not full: 1
165         //The layout of the bins are as follows.
166         //0
167         //1
168         //empty
169         //empty
170         //empty
171         //empty
172     }
173     else if (type == 'H') {
174         //When a bin is full, cannot add more element to it.
175         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
180
181         std::cout << "The layout of the bins are as follows." << std::endl
182         ;
183         for (int i = 0; i < game->numBins; i++) {
184             if (game->grid[i].size() == 0)
185                 std::cout << "empty" << std::endl;
186             else {
187                 for (int j = 0; j < game->grid[i].size(); j++)
188                     std::cout << game->grid[i][j] << " ";
189                 std::cout << std::endl;
190             }
191         }
192         //sample output:
193         //Enter a bin index in [0, 6) that is not full: 0
194         //the bin is full
195         //Re-enter a bin index in [0, 6) that is not full: 1
196         //The layout of the bins are as follows.
197         //0 1 0 1
198         //0
199         //empty
200         //empty
201         //empty
202         //empty
203         }
204         //When type is 'A' - 'C', work on constructors
205         if (type == 'A' || type == 'B' || type == 'C') {
206             std::cout << "After " << prompt
207                 << " data member numBins is " << game->numBins << std::endl;
208             std::cout << "After " << prompt
209                 << " data member capacity is " << game->capacity << std::endl;
210
211             for (int i = 0; i < game->numBins; i++) {
212                 std::cout << "number of elements of bin "
213                     << i << " is: "
214                     << game->grid[i].size() << std::endl;
215             }
216         }
217     }
218
219     delete game;

```

```

220     game = nullptr;
221
222     return 0;
223 }

```

(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

```

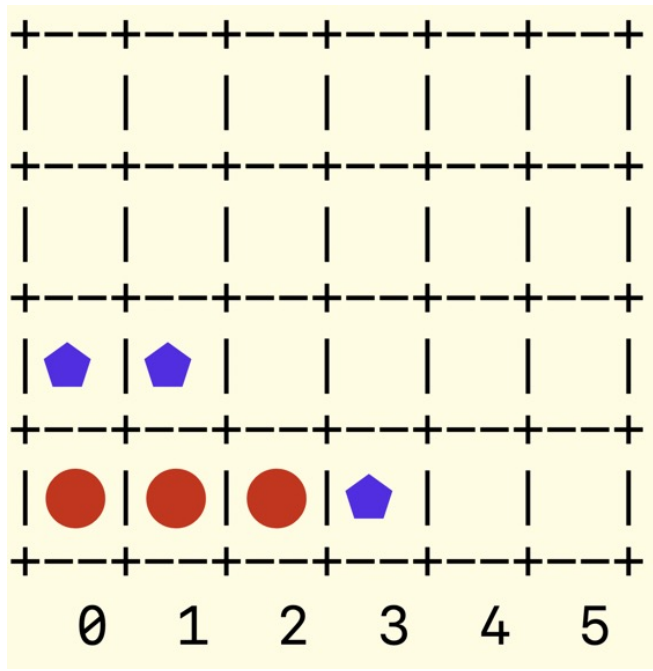
5. Run ./test C, we get the following output.

```

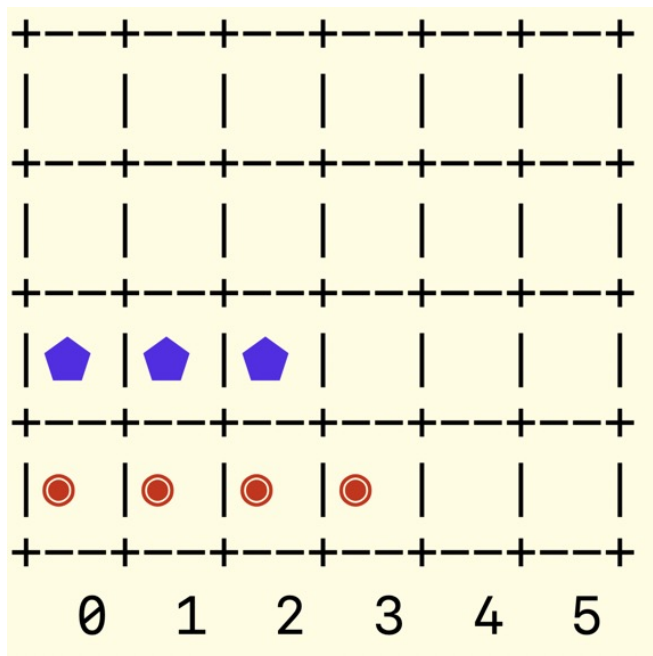
1 After Board game(5, 1); data member numBins is 5
2 After Board game(5, 1); data member capacity is 4
3 number of elements of bin 0 is: 0
4 number of elements of bin 1 is: 0
5 number of elements of bin 2 is: 0
6 number of elements of bin 3 is: 0
7 number of elements of bin 4 is: 0

```

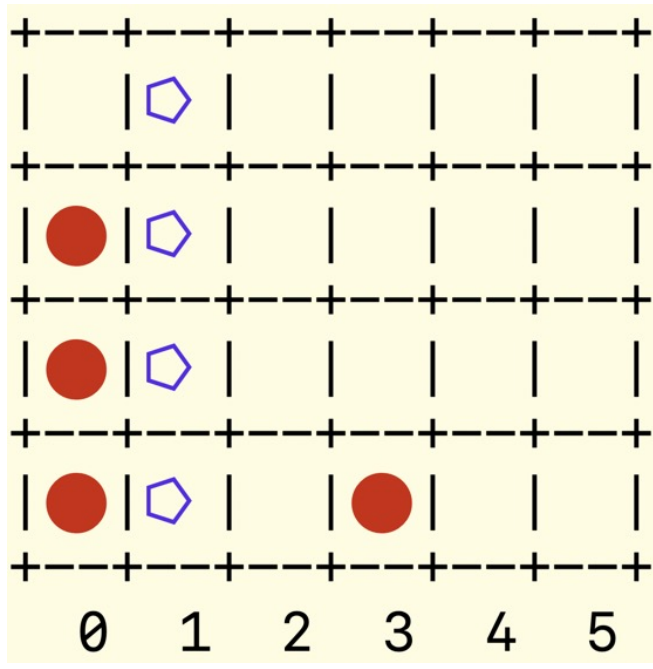
6. Run ./test D. Get the following output.



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. Similarly, 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 };

```

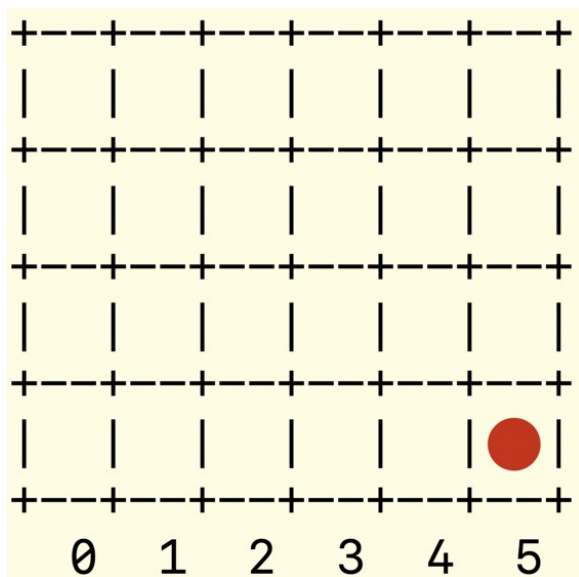
Some students ask the differences between **struct** and **class** in C++.

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-struct> 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.

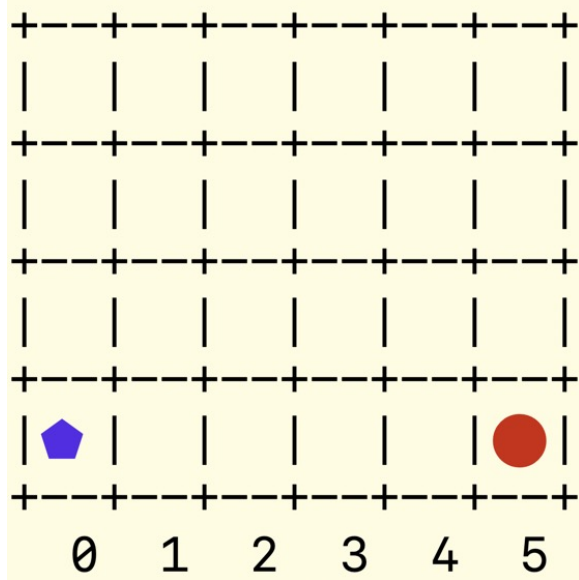


The above process is achieved by the following statements, taken from **main.cpp** to test methods in Task B.

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.

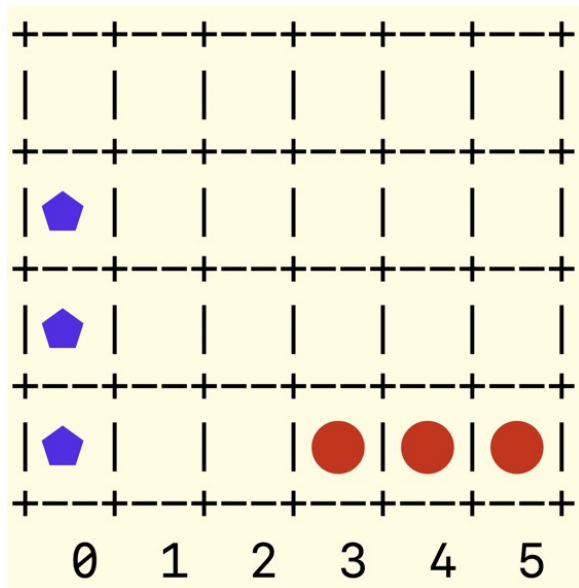


This is achieved by adding the following statements to the above codes.

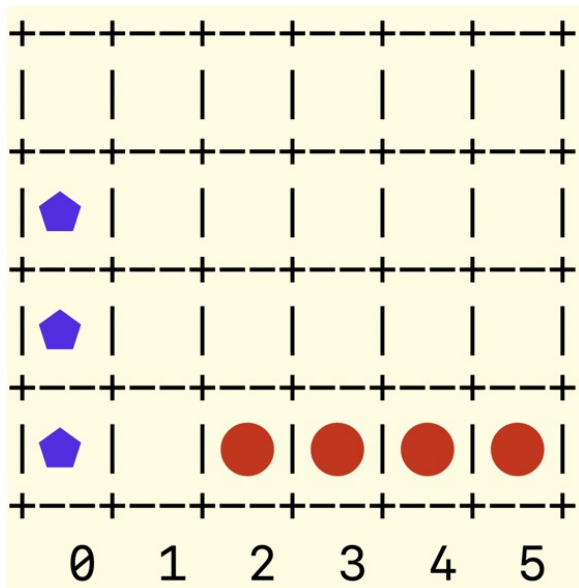
```
1 game->grid[0].push_back(1);
```

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 on bin 2, the most recent bin with a new element added. In this example, the return should be 0, the id of the player who wins the game. That is, player 0 wins in horizontal direction.

If there is no winner, return -1.

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

6. 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.

+	-	+	-	+	-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-	+	-	+	-	+
	⬡											
+	-	+	-	+	-	+	-	+	-	+	-	+
	⬡											
+	-	+	-	+	-	+	-	+	-	+	-	+
	⬡			⬤		⬤		⬤		⬤		
+	-	+	-	+	-	+	-	+	-	+	-	+
	0		1		2		3		4		5	

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`.

```

1 int Board::winInHorizontal(int bin) {
2     //Find out the last index of bin.
3

```

```

4 //Find out the player residing at that last index of bin.
5 //Save in an int variable called player.
6
7 //Declare candidates as a vector of Coord.
8
9 //candidates saves the locations of
10 //all horizontal neighbors of the top elements of the given parameter bin
11 //that share the same value of that top element,
12 //which is saved in player.
13
14
15 //Who can be the first element of candidates?
16
17
18 //count consecutive elements equaling player to the left of grid[bin][idx]. Save
19 //their locations to candidates.
20
21 //count consecutive elements equaling player to the right of grid[bin][idx]. Save
22 //their locations to candidates.
23
24 if (count >= 4) {
25     //change all elements in candidates to e player +2
26
27     return player;
28 }
29
30 //Return -1 when no winner in horizontal direction
31 }

```

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

```
1 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`.

```
1 ./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.

```
1 #include <iostream>
2 #include <vector>
3 #include "Board.hpp"
4 //g++ -std=c++11 Board.cpp main.cpp -o test
5 //test different methods using
6 //./test 1 or ./test '1'
7 //./test 2 or ./test '2'
8 //...
9 //./test 9 or ./test '9'
10
11
12 int main(int argc, const char *argv[]) {
13     if (argc != 2) {
14         std::cout << "Need 'A'-'C' in parameters" << std::endl;
15         return -1;
16     }
17
18     //unit-testing for constructors and the destructor
19     char type = *argv[1];
20     std::string prompt;
21     Board *game;
22     int player = -2; //a not-exist value
23
24     if (type == '1') {
25         //test player 0 wins in winInHorizontal
26         game = new Board; //default constructor
27         game->grid[5].push_back(0);
28         //game->display();
29         game->grid[0].push_back(1);
30         //game->display();
31         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
78 // , , , , ,
79 //1, , , , ,
80 //1, , , , ,
81 //1, , ,0,0,0,

```

```

82 //Visual result
83 //+---+---+---+---+---+
84 //|   |   |   |   |   |
85 //+---+---+---+---+---+
86 //|   |   |   |   |   |
87 //+---+---+---+---+---+
88 //|   |   |   |   |   |
89 //+---+---+---+---+---+
90 //|   |   |   |   |   |
91 //+---+---+---+---+---+
92 // 0  1  2  3  4  5
93 }
94     else if (type == '3') {
95         //win in vertical
96         game = new Board;
97
98         game->grid[0].push_back(0);
99         game->grid[1].push_back(1);
100        game->grid[0].push_back(0);
101        game->grid[1].push_back(1);
102        game->grid[0].push_back(0);
103        game->grid[1].push_back(1);
104        game->grid[3].push_back(0);
105        game->grid[1].push_back(1);
106        //game->display();
107
108        player = game->winInVertical(1);
109        prompt = " in vertical";
110
111 //sample output:
112 //player 1 wins in vertical
113 // ,3, , , , ,
114 //0,3, , , , ,
115 //0,3, , , , ,
116 //0,3, ,0, , ,
117 //+---+---+---+---+---+
118 //|   |   |   |   |   |
119 //+---+---+---+---+---+
120 //|   |   |   |   |   |
121 //+---+---+---+---+---+
122 //|   |   |   |   |   |
123 //+---+---+---+---+---+
124 //|   |   |   |   |   |
125 //+---+---+---+---+---+
126 // 0  1  2  3  4  5
127 }

```

```

128     else if (type == '4') {
129         //no win in vertical
130         game = new Board;
131
132         game->grid[0].push_back(0);
133         game->grid[1].push_back(1);
134         game->grid[0].push_back(0);
135         game->grid[1].push_back(1);
136         game->grid[0].push_back(0);
137         game->grid[1].push_back(1);
138         //game->display();
139
140         player = game->winInVertical(1);
141         prompt = " in vertical";
142         //Sample output:
143         //+---+---+---+---+
144         //|   |   |   |   |
145         //+---+---+---+---+
146         //|   |   |   |   |
147         //+---+---+---+---+
148         //|   |   |   |   |
149         //+---+---+---+---+
150         //|   |   |   |   |
151         //+---+---+---+---+
152         // 0  1  2  3  4  5
153         //no player wins in vertical
154         // , , , , ,
155         //0,1, , , ,
156         //0,1, , , ,
157         //0,1, , , ,
158     }
159     else if (type == '5') {
160         //win in diagonal
161         game = new Board;
162         game->grid[0].push_back(0);
163         game->grid[1].push_back(1);
164         game->grid[1].push_back(0);
165         game->grid[2].push_back(1);
166         game->grid[3].push_back(0);
167         game->grid[2].push_back(1);
168         game->grid[2].push_back(0);
169         game->grid[3].push_back(1);
170         game->grid[3].push_back(0);
171         game->grid[4].push_back(1);
172         game->grid[3].push_back(0);
173

```



```

174     player = game->winInDiagonal(0);
175     prompt = " in diagonal";
176
177     //game->display();
178
179     //Sample output:
180     //player 0 wins in diagonal
181     // , , ,2, , ,
182     // , ,2,0, , ,
183     // ,2,1,1, , ,
184     //2,1,1,0,1, ,
185     //+---+---+---+---+---+
186     //|   |   |   |   |   |
187     //+---+---+---+---+---+
188     //|   |   |   |   |   |
189     //+---+---+---+---+---+
190     //|   |   |   |   |   |
191     //+---+---+---+---+---+
192     //|   |   |   |   |   |
193     //+---+---+---+---+---+
194     // 0  1  2  3  4  5
195     }
196     else if (type == '6') {
197         //win in anti-diagonal
198         game = new Board;
199         game->grid[0].push_back(0);
200         game->grid[1].push_back(1);
201         game->grid[2].push_back(0);
202         game->grid[3].push_back(1);
203         game->grid[0].push_back(0);
204         game->grid[2].push_back(1);
205         game->grid[0].push_back(0);
206         game->grid[0].push_back(1);
207         game->grid[1].push_back(0);
208         game->grid[1].push_back(1);
209
210         player = game->winInDiagonal(1);
211         prompt = " in anti-diagonal";
212
213         //game->display();
214
215         //sample output:
216         //+---+---+---+---+---+
217         //|   |   |   |   |   |
218         //+---+---+---+---+---+
219         //|   |   |   |   |   |

```

```

220 //+---+---+---+---+
221 //|   |   |   |   |   |
222 //+---+---+---+---+
223 //|   |   |   |   |   |
224 //+---+---+---+---+
225 // 0 1 2 3 4 5
226 //player 1 wins in anti-diagonal
227 //3, , , , ,
228 //0,3, , , ,
229 //0,0,3, , ,
230 //0,1,0,3, , ,
231 }
232 else if (type == '7') {
233     //no win, a tie
234     //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'
235     game = new Board;
236     game->grid[0].push_back(0);
237     game->grid[1].push_back(1);
238     game->grid[0].push_back(0);
239     game->grid[2].push_back(1);
240     game->grid[0].push_back(0);
241     game->grid[0].push_back(1);
242     game->grid[3].push_back(0);
243     game->grid[1].push_back(1);
244     game->grid[2].push_back(0);
245     game->grid[1].push_back(1);
246     game->grid[1].push_back(0);
247     game->grid[2].push_back(1);
248     game->grid[3].push_back(0);
249     game->grid[3].push_back(1);
250     game->grid[5].push_back(0);
251     game->grid[3].push_back(1);
252     game->grid[5].push_back(0);
253     game->grid[5].push_back(1);
254     game->grid[4].push_back(0);
255     game->grid[4].push_back(1);
256     game->grid[4].push_back(0);
257     game->grid[5].push_back(1);
258     game->grid[4].push_back(0);
259     game->grid[2].push_back(1);
260
261     player = game->win(1);
262
263     //game->display();
264     //sample output:
265     //+---+---+---+---+

```

```

266 //|   |   |   |   |   |
267 //+---+---+---+---+---+
268 //|   |   |   |   |   |
269 //+---+---+---+---+---+
270 //|   |   |   |   |   |
271 //+---+---+---+---+---+
272 //|   |   |   |   |   |
273 //+---+---+---+---+---+
274 // 0 1 2 3 4 5
275 //no player wins
276 //1,0,1,1,0,1,
277 //0,1,1,1,0,1,
278 //0,1,0,0,1,0,
279 //0,1,1,0,0,0,
280 }
281 else if (type == '8') {
282     //msg = '0 1 0 2 1 3 0 4\n'
283     game = new Board;
284     game->grid[0].push_back(0);
285     game->grid[1].push_back(1);
286     game->grid[0].push_back(0);
287     game->grid[2].push_back(1);
288     game->grid[1].push_back(0);
289     game->grid[3].push_back(1);
290     game->grid[0].push_back(0);
291     game->grid[4].push_back(1);
292
293     player = game->win(4); //4 is the most recent bin with balls added
294
295     //game->display();
296     //sample output:
297     //+---+---+---+---+---+
298     //|   |   |   |   |   |
299     //+---+---+---+---+---+
300     //|   |   |   |   |   |
301     //+---+---+---+---+---+
302     //|   |   |   |   |   |
303     //+---+---+---+---+---+
304     //|   |   |   |   |   |
305     //+---+---+---+---+---+
306     // 0 1 2 3 4 5
307     //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);
323
324         player = game->win(1); //1 is the most recent bin with balls added
325
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     }
344
345     if (player != -1)
346         std::cout << "player " << player << " wins";
347     else std::cout << "no player wins";
348
349     std::cout << prompt << std::endl;
350
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())
354                 std::cout << game->grid[i][j] << ",";
355             else std::cout << " ,";
356         }
357         std::cout << std::endl;

```

```

358     }
359
360     delete game;
361     game = nullptr;
362
363     return 0;
364 }

```

6 Task C: define play method

Main steps of play method are as follows.

1. Display an empty grid (no element in any cell). Which method should be call?
2. Keep doing the followin until either a winner is found or every cell in the grid is filled.
 - (a) In each round, call add method on player 0 and 1 alternatively, where player 0 starts first.
 - (b) Display the grid after each call of add
3. Once we are out of the above loop, either a winner is found or all cells are filled and no one wins, in that situation, print “a tie”.
4. To pass gradescope, need to pay attention to the following details.
 - (a) In each round, player 0 (mapping to a red circle) plays first, and player 1 (mapping to a blue pentagon in display methoe) plays next.
 - (b) The prompt for add method must start with “Enter ...”. For example, “Enter an index in [0, *numBins*): ”, where numBins is the number of bins.
 - (c) Once the game is finished, output a line like “winner: red” or “winner: blue” or “a tie”.

6.1 Test codes locally

Here is main.cpp for Task C.

```

1  #include "Board.hpp"
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  //compile and link using command
7  //g++ -std=c++11 Board.cpp main.cpp -o test
8  //Then run
9  //./test A
10 //or
11 //./test B
12

```

```

13 int main(int argc, const char* argv[]) {
14     if (argc != 2) {
15         cout << "Need add an integer, like A, after ./a.out" << endl;
16         return -1;
17     }
18
19     switch (*argv[1]) {
20         case 'A':
21         {
22             Board game;
23             game.play();
24             break;
25         }
26         case 'B':
27         {
28             Board game(7,5);
29             game.play();
30             break;
31         }
32     }
33
34     return 0;
35 }

```

Compile and link the code using command

```

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

```

Then run the following command to call **play** method of a default Board object.

```

1 ./test A

```

6.2 a sample run

1. Display an empty grid. Enter 1 with return key to put red circle to bin 1.

```

+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
  0   1   2   3   4   5

```

```

add shape: 1, player: red
Enter a bin index (starting from 0): 1

```

2. Enter 0 with return key to put blue pentagon to bin 0. Note prompt “add shape: ” means the number of shapes to be added to the grid.

```

+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
|   | ● |   |   |   |   |
+---+---+---+---+---+---+
  0   1   2   3   4   5

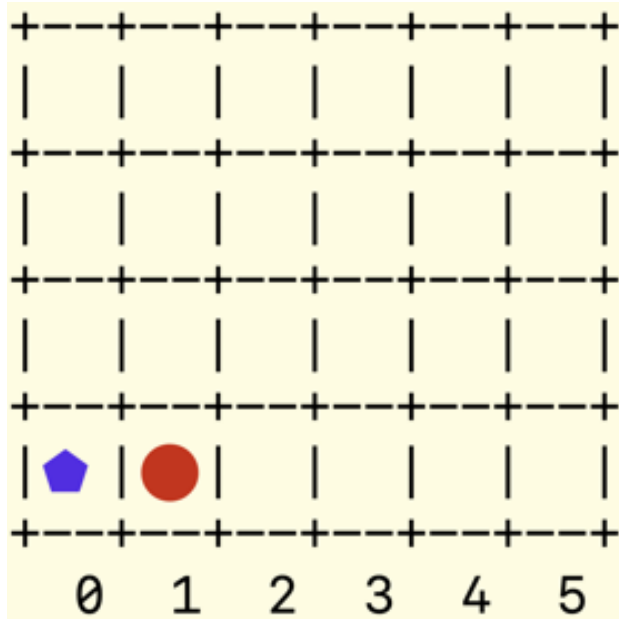
```

```

add shape: 2, player: blue
Enter a bin index (starting from 0): 0

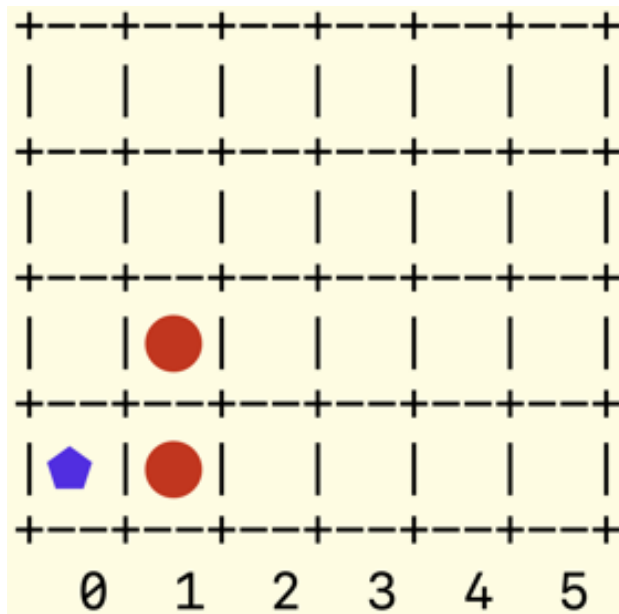
```

3. Enter 1 to put red circle to bin 1.



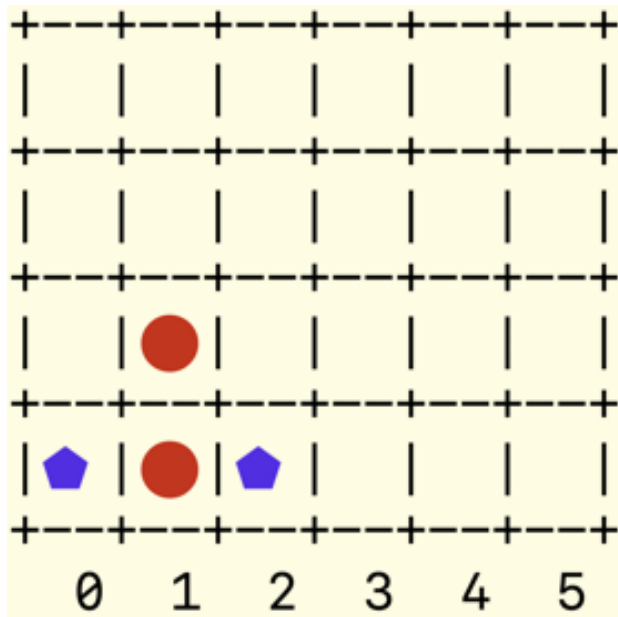
```
add shape: 3, player: red
Enter a bin index (starting from 0): 1
```

4. Enter 2 to put blue pentagon to bin 2.



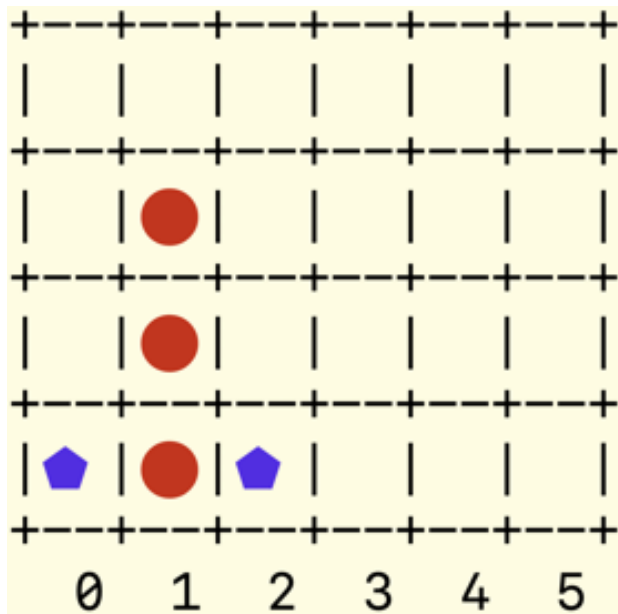
```
add shape: 4, player: blue
Enter a bin index (starting from 0): 2
```


5. Enter 1 to put red circle to bin 1.



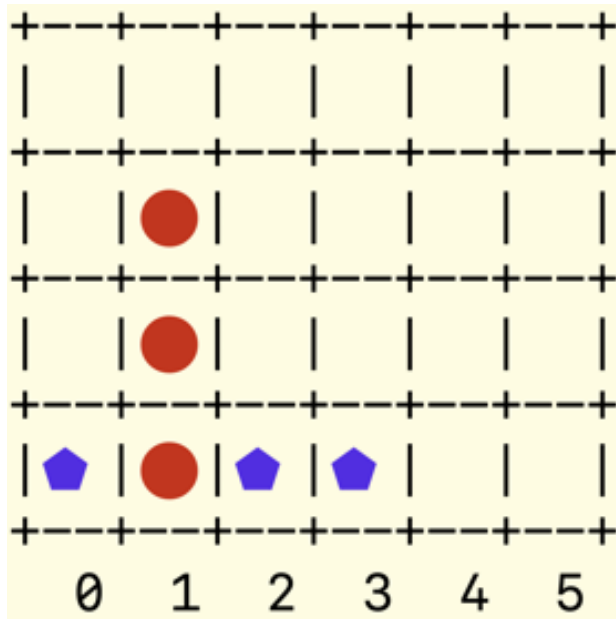
```
add shape: 5, player: red
Enter a bin index (starting from 0): 1
```

6. Enter 3 to put blue pentagon to bin 3.



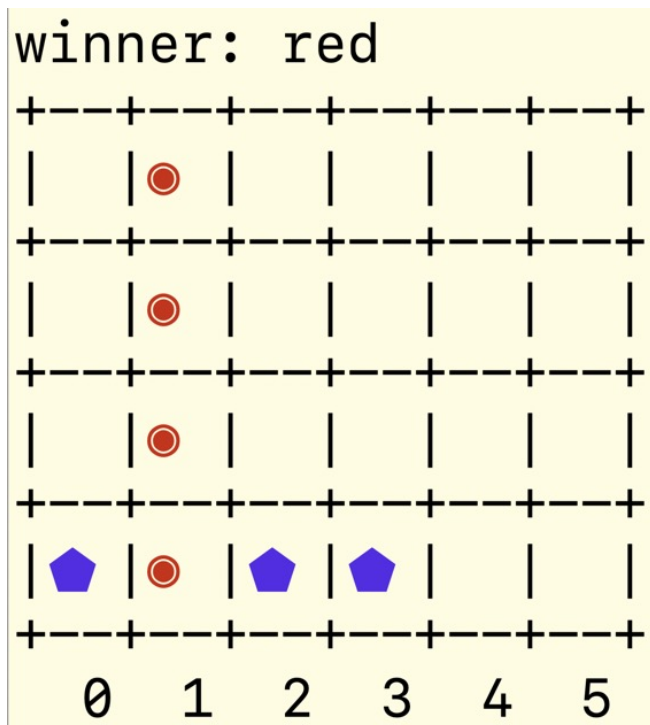
```
add shape: 6, player: blue
Enter a bin index (starting from 0): 3
```

7. Enter 1 to put red circle to bin 1.



```
add shape: 7, player: red
Enter a bin index (starting from 0): 1
```

8. The game ends. Declare the winner and show the consecutive elements.



6.3 Test play method of a non-default constructor

Run the following command to call `play` method of non-default `Board` object with 7 bins, each bin can hold at most 5 elements.

```
1 ./test B
```

Once your code works fine in a local computer, upload `Board.cpp` to gradescope.

6.4 BoardTest.cpp

Create `BoardTest.cpp` with the following contents. The purpose of `BoardTest.cpp` is to test constructors and methods defined in `Board.cpp`.

```
1 #include "Board.hpp"
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main() {
7     //TODO: declare a Board object called game using its default constructor
8
9     //TODO: call play method of Board object game.
10
11     return 0;
12 }
```

`Board.cpp` concentrates on defining a `Board` object while `BoardTest.cpp` focus on instantiating (aka, creating) a `Board` object and tests methods declared in `Board.hpp`.

6.5 makefile

Motivation: a large project may involve several files. We do not want to enter commands to compile and run each time. Also, we would like to re-compile or re-link only modified codes, not to re-compile or re-link each file.

Makefile comes to rescue.

6.6 Use makefile

For a large C++ project, it would better to use makefile, with which, only the modified source codes are recompiled and re-linked.

1. Edit a file called `makefile` with the following contents.

```
1 # This is an example Makefile for connect-4 project.
2 # This program uses Board and BoardTest modules.
3 # Typing 'make' or 'make run' will create the executable file.
4 #
5
6 # define some Makefile variables for the compiler and compiler flags
```

```

7 # to use Makefile variables later in the Makefile: $()
8 #
9 # -g      adds debugging information to the executable file
10 # -Wall  turns on most, but not all, compiler warnings
11 #
12 # for C++ define CC = g++
13 CC = g++ -std=c++11
14 #CFLAGS  = -g -Wall
15
16 # typing 'make' will invoke the first target entry in the file
17 # (in this case the default target entry)
18 # you can name this target entry anything, but "default" or "all"
19 # are the most commonly used names by convention
20 #
21 all: run
22
23 # To create the executable file connect4 (see -o connect4), we need the object
   files
24 # BoardTest.o and Board.o:
25 run: BoardTest.o Board.o
26     $(CC) -o connect4 BoardTest.o Board.o
27
28 # To create the object file BoardTest.o, we need the source
29 # files BoardTest.cpp, Competition.h
30 BoardTest.o: BoardTest.cpp
31     $(CC) -c BoardTest.cpp
32
33 # To create the object file Board.o, we need the source files
34 # Board.cpp.
35 # By default, $(CC) -c Board.cpp generates Board.o
36 Board.o: Board.cpp
37     $(CC) -c Board.cpp
38
39 # To start over from scratch, type 'make clean'. This
40 # removes the executable file, as well as old .o object
41 # files and *~ backup files:
42 #
43 clean:
44     $(RM) connect4 *.o *~

```

According to the command in this makefile,

```
$(CC) -o connect4 BoardTest.o Board.o
```

The generated runnable file is called `connect4`, which appears after `-o`.

2. Run make command.

```
make
```

3. If there is no error in the above command, run the following command, where dot (.) means current directory.

`./connect4`