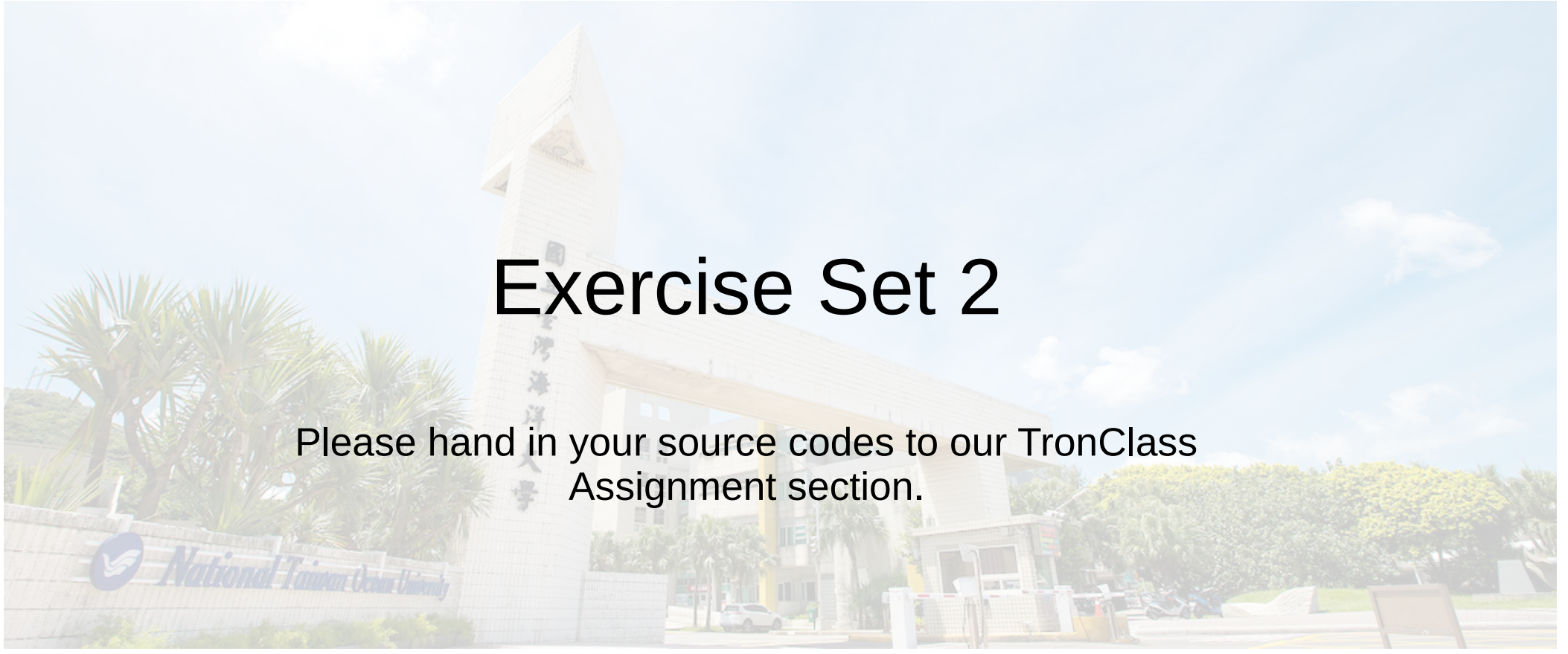


Exercise Set 2

Please hand in your source codes to our TronClass Assignment section.

Problem 01 (30%)

Please modify the following two classes to be template classes so that we can get the sample output using the designated main function.

```
class node {
public:
    int v;
    node *next;
    node(int x) { v = x; next = nullptr; }
};
class queue {
    node *start;
    node *end;
public:
    queue();
    bool empty();
    // enqueue nonrepeated elements
    void safe_push(int v);

    int front();
    void pop();
};
```

```
int main() {// do not modify main()
    queue<string> q;
    string s1, s2, s3, s4;
    cin >> s1 >> s2 >> s3 >> s4;
    q.safe_push(s1);
    q.safe_push(s2);
    q.safe_push(s3);
    q.safe_push(s4);
    q.pop();
    while (!q.empty()) {
        cout << q.front() << ' ';
        q.pop();
    }
    return 0;
}
```

Sample Input & Output

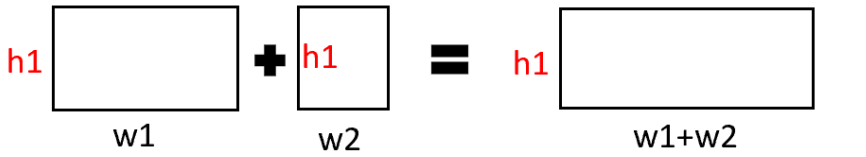
Sample input	A AA AA AAA	AA AAA
Sample output	I AM JOHN STOCKTON.	AM JOHN STOCKTON.

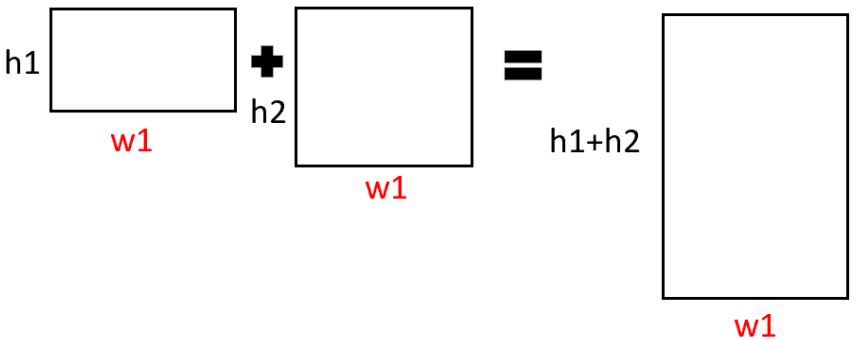
Problem 02 (30%)

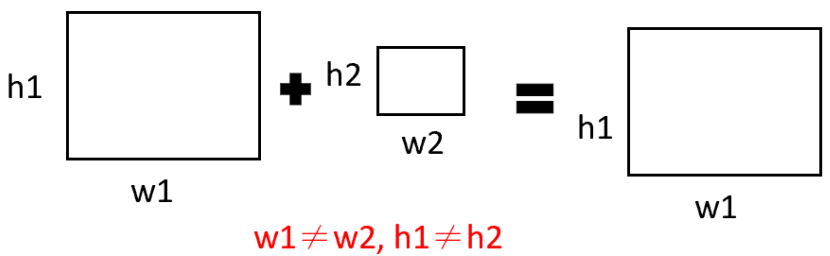
Consider the following class `rectangle`. Please modify and complete the program according to the instructions in the comments so that the main function can successfully complete the addition overloading and printing. (See the illustrating examples)

```
class rectangle {
private:
    int width, height; // width and height of a rectangle
public:
    /* complete the rectangle constructor(s) */
    rectangle operator+(const rectangle& r) {
        /* Complete the operator overloading */
    }
    rectangle operator=(const rectangle& r) {
        /* Complete the operator overloading */
    }
    friend ostream& operator<<(ostream& os, const rectangle& r);
};
ostream& operator<<(ostream& os, const rectangle& r) {
    /* Complete the operator overloading so that the rectangle
       can be printed */
}
```

Problem 02 (contd.)







$w1 \neq w2, h1 \neq h2$

Sample Input & Output

```
int main() { // DO NOT MODIFY
    int w1, h1, w2, h2;
    cin >> w1 >> h1 >> w2 >> h2;
    rectangle r1(w1,h1), r2(w2,h2), r;
    r = r1+r2;
    cout << r;
    return 0;
}
```

Sample input	2 2 2 3	3 2 2 2
Sample output	* * * * * * * * * *	* * * * * * * * * *

Problem 03: Poker Probability (40%)

- Refer to our lectures
  - https://josephcclin.github.io/courses/cpp/slides/CPP_lect_12_case_studies.pdf
- Add a function:

```
bool is_straight_flush(vector<card> &hand)
```

  to the program https://www.onlinegdb.com/vdGeAd2QIg

  to compute the number of fullhouses.

- Sample output:

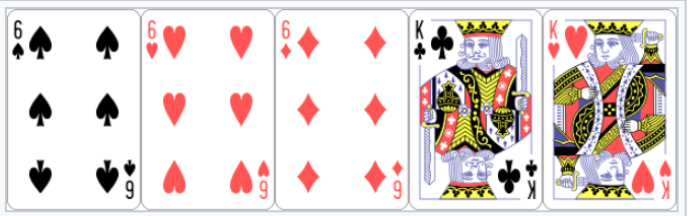
```
Flushes: 3 out of 1000
Straights: 6 out of 1000
Straight Flushes: 0 out of 1000
Fullhouses: 4 out of 1000
```

Note: Full House

- From Wikipedia:

Full house [edit]

A **full house**, also known as a *full boat* or a *tight* or a *boat* (and originally called a **full hand**), is a hand that contains three cards of one rank and two cards of another rank, such as 3♣ 3♠ 3♦ 6♣ 6♥ (a "full house, threes over sixes" or "threes full of sixes" or "threes full").[17][18] It ranks below four of a kind and above a flush.[5]



A full house, sixes over kings