Office Hour #5

```
// min() and max() using templates

#include <iostream>
using namespace std;

template <typename T>
T min(T* begin, T* end) {
    T min_value = *begin++;

    while (begin != end) {
        if (*begin < min_value)
            min_value = *begin;
        begin++;
    }

    return min_value;
}

template <typename T>
T max(T* begin, T* end) {
    T max_value = *begin++;

    while (begin != end) {
        if (*begin > max_value)
            max_value = *begin;
        begin++;
    }

    return max_value;
}
```

```
1. Pair<T>
2. p.max_value = *begin;
3. p.min_value = *begin;
```

```
template <class T>
void myVector<T>::push_back(const T &value) {
    if (length == capacity) {
        T* new_storage = new T[capacity*2];         // dynamic allocation of template-type
array
        capacity *= 2;
        for (int i = 0; i < length; i++)
            new_storage[i] = arr[i];
        T* temp = arr;
        arr = new_storage;
        delete [] temp;
    }
    arr[length++] = value;
}
```

```
#include <iostream>
```

```
using namespace std;

template<class T>
class test {
public:
    test (T str): m_value(str) {}
    T front(){return m_value;}
private:
    T m_value;
};

int main() {
    test<char*> s("Hello World!");
    cout << s.front() << endl;
}
```

```
1
```