1)
```
Attend & operator=(const Attend & att)
{
    if (&att == this) return *this;
    fio = att.fio;
    absend.clear();
    for (auto & elem: att.absend)
    {
        absend.insert(make_pair(elem.first, elem.second));
    }
    return *this;
}
```

2)
```
int & operator[](const int idx)
{
    map<int, string> months;
    vector<string> months = {"January", "February",
        ..., "December"};
    return absend[months[idx-1]];
}
```

3)
```
void addPass(const int idx)
{
    (*this)[idx]++; // Вызов операции []
}
```

6)
```
#include <algorithm>
int main()
{
    list<string> data{"aba", "123", "r5r"};
    list<string>::iterator pos;
    for (pos = data.begin(); pos != data.end(); ++pos)
    {
        string tmp(*pos);
        reverse(tmp.begin(), tmp.end());
        if (tmp == *pos) cout << *pos << " ";
    }
    for_each(data.begin(), data.end(), [](string &s)
    {
        string tmp(s);
        reverse(tmp.begin(), tmp.end());
        if (tmp == s) cout << s << " ";
    });
}
```

c)

```
mutex mtx;
conditional_variable cond;
bool evenPrinted = false;

int main()
{
    thread thrEven([](int n)
    {
        for (int i=0; i<n; ++i)
        {
            unique_lock<mutex> lock(mtx);
            cond.wait(lock, [] {return !evenPrinted;});
            cout << 2*i << " ";
            evenPrinted = true;
            cond.notify_all();
        }
    }, 10);

    thread thrOdd([](int n)
    {
        for (int i=0; i<n; ++i)
        {
            unique_lock<mutex> lock(mtx);
            cond.wait(lock, [] {return evenPrinted;});
            cout << 2*i+1;
            evenPrinted = false;
            cond.notify_all();
        }
    }, 10);

    thrEven.join();
    thrOdd.join();
}
```