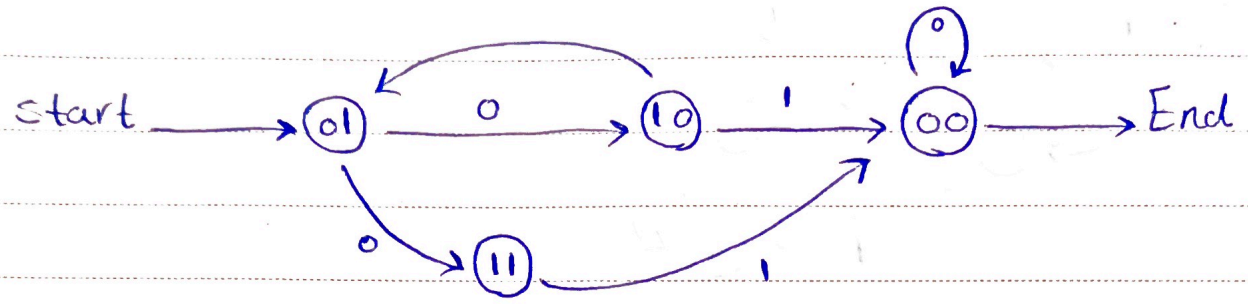
سوال ۱:

INTEGER = Digit$^+$ | 0(x|X)(Digit | Hex)$^+$

Digit = 0..9 Hex = a...f | A...F

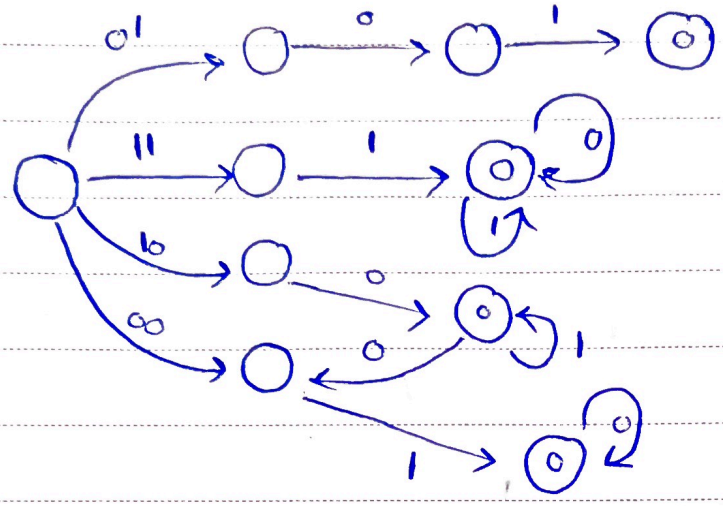
"Scanner" اتوماتا



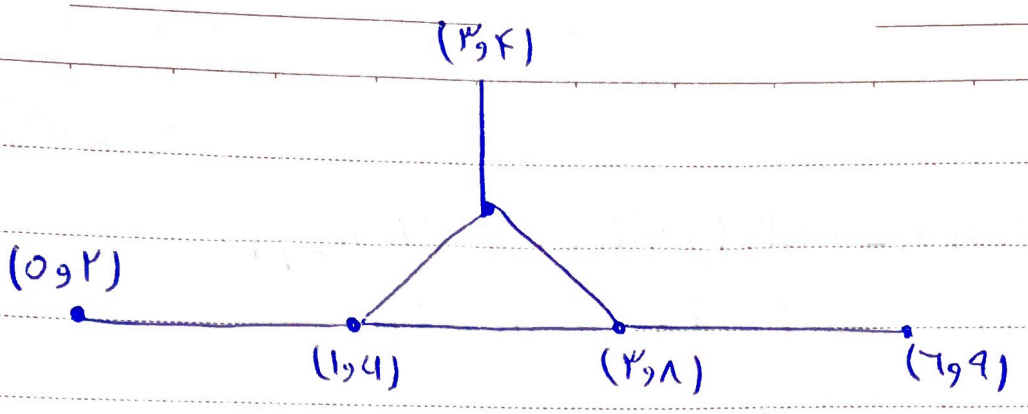
سوال ۲:

```
stksg   segment   stack
        db 64 dup (stack)
stksg   ends.

Datasg  segment   "Data".
        db
datasg  ends.

cseg    segment
assume  cs&cseg
main    proc
push    ds
push    0
mov     ax
mov     ds, ax
```

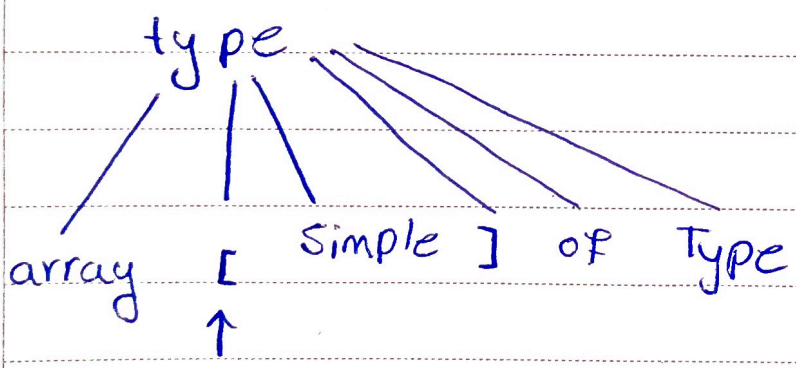
سوال ۵۳ نهایی Type را به فرم BNF بنویسید:

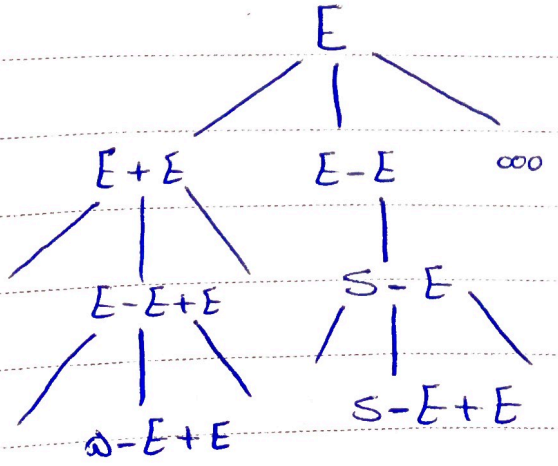


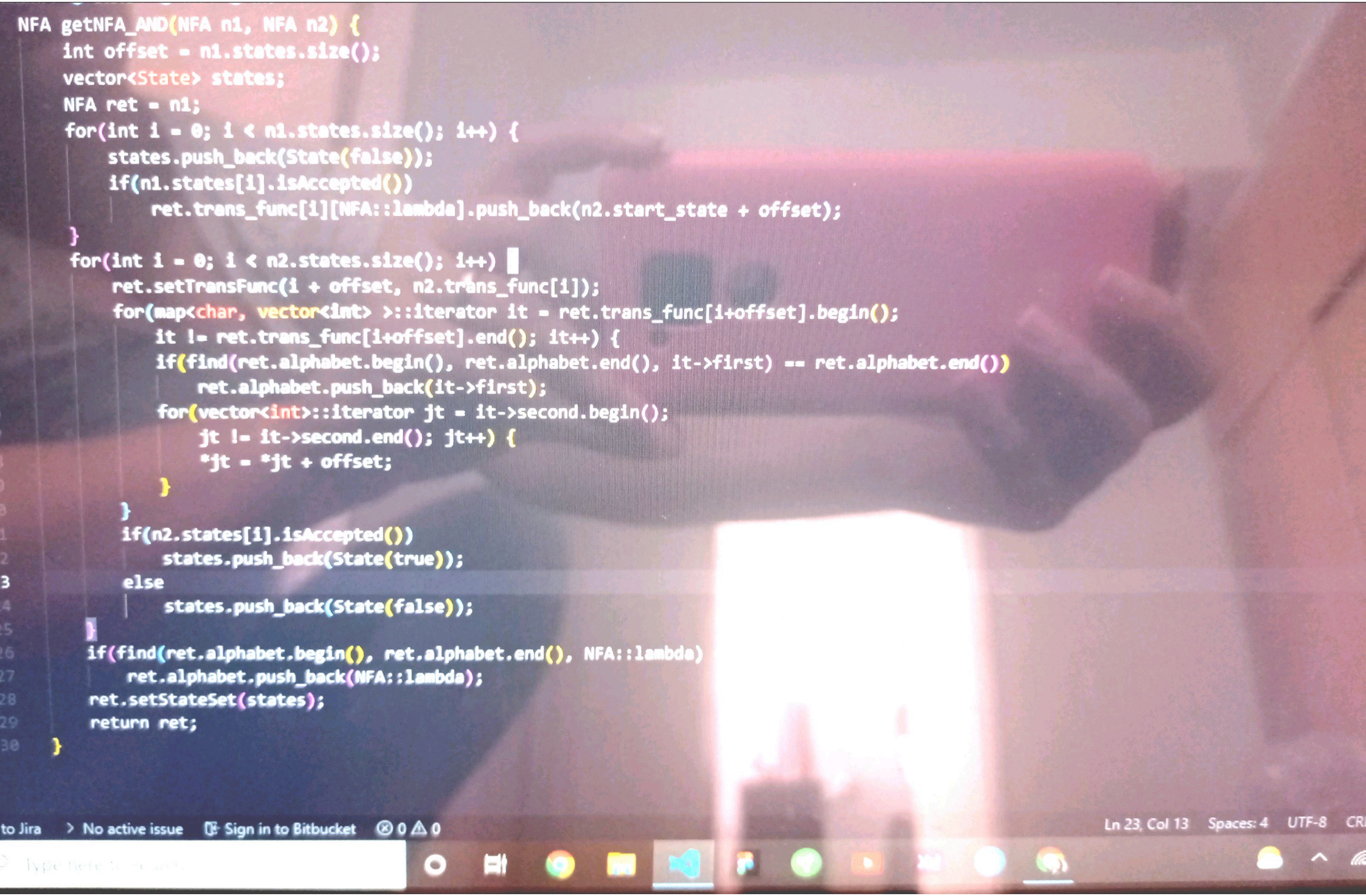
سؤال 4



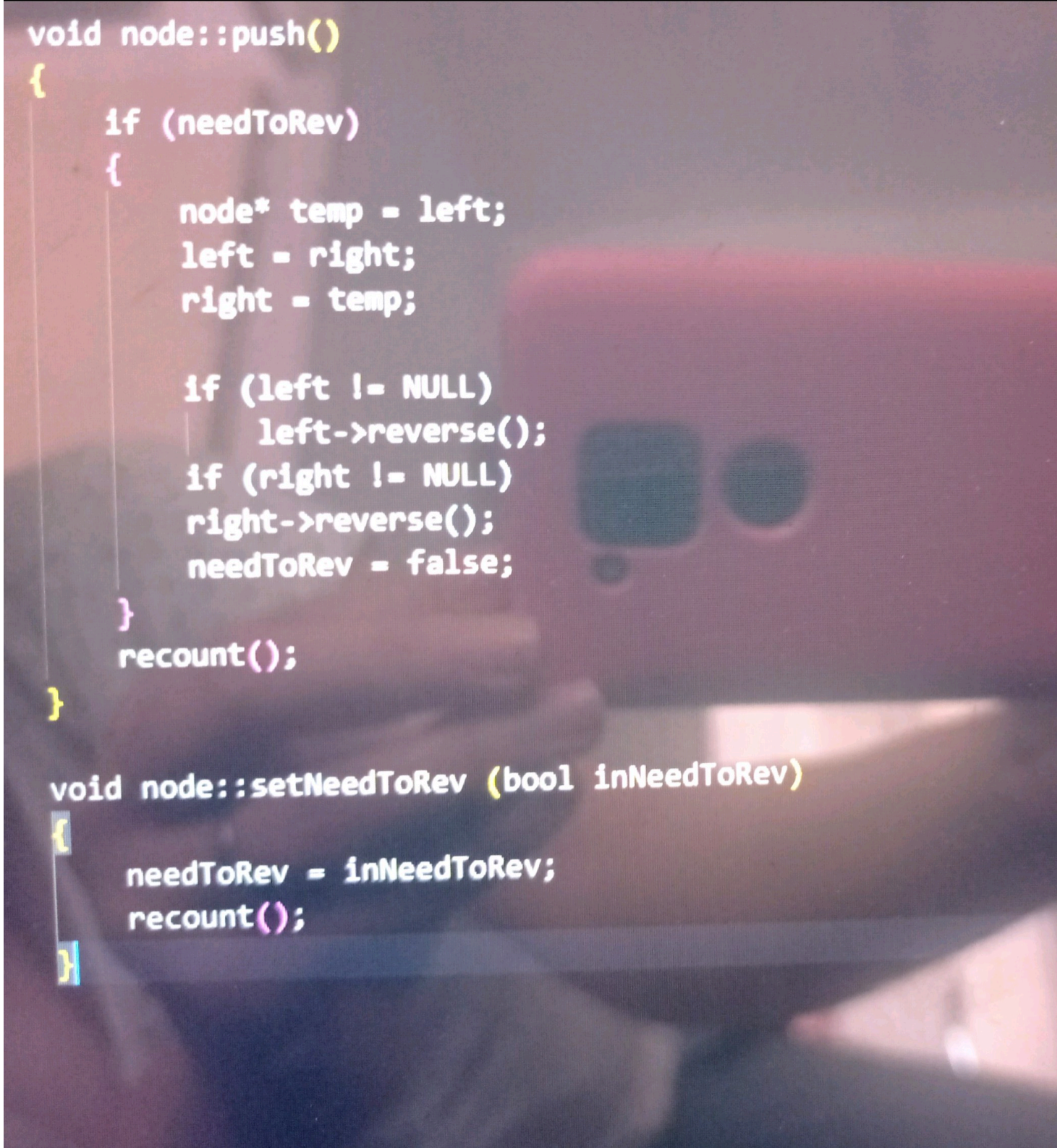
سؤال 5







```
NFA getNFA_AND(NFA n1, NFA n2) {
    int offset = n1.states.size();
    vector<State> states;
    NFA ret = n1;
    for(int i = 0; i < n1.states.size(); i++) {
        states.push_back(State(false));
        if(n1.states[i].isAccepted())
            ret.trans_func[i][NFA::lambda].push_back(n2.start_state + offset);
    }
    for(int i = 0; i < n2.states.size(); i++)
        ret.setTransFunc(i + offset, n2.trans_func[i]);
        for(map<char, vector<int> >::iterator it = ret.trans_func[i+offset].begin();
            it != ret.trans_func[i+offset].end(); it++) {
            if(find(ret.alphabet.begin(), ret.alphabet.end(), it->first) == ret.alphabet.end())
                ret.alphabet.push_back(it->first);
            for(vector<int>::iterator jt = it->second.begin();
                jt != it->second.end(); jt++) {
                *jt = *jt + offset;
            }
        }
        if(n2.states[i].isAccepted())
            states.push_back(State(true));
        else
            states.push_back(State(false));
    }
    if(find(ret.alphabet.begin(), ret.alphabet.end(), NFA::lambda)
        ret.alphabet.push_back(NFA::lambda);
    ret.setStateSet(states);
    return ret;
}
```



```
void node::push()
{
    if (needToRev)
    {
        node* temp = left;
        left = right;
        right = temp;

        if (left != NULL)
            left->reverse();
        if (right != NULL)
        right->reverse();
        needToRev = false;
    }
    recount();
}

void node::setNeedToRev (bool inNeedToRev)
{
    needToRev = inNeedToRev;
    recount();
}
```