```cpp
//
//  knuth_morris_pratt.cpp
//  helloworld
//
//  Created by BETA on 12/14/16.
//  Copyright © 2016 BETA. All rights reserved.
//

#include <stdio.h>
#include <string>
#include <vector>
#include <iostream>
using namespace std;

class KMP {
private:
    string pat;
    int M;
    int R; // 256:extended ASCII
    vector<vector<int>> dfa;

public:
    KMP(string pat){
        this->pat = pat;
        M = pat.length();
        R = 256;
        dfa = vector<vector<int>>(R, vector<int>(M, 0));

        // build DFA table from pattern
        //buildDFA();
        dfa[pat[0]][0] = 1; // 因为buildDFA表的时候从1开始
        for(int X = 0, j=1; j<M; j++){ // 关键在这里j=1，如果j=0开始，的话在状态j=1
            开始时X=1，而X应该是0
            for(int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][X]; // dismatch
            dfa[pat[j]][j] = j+1; // match
            X = dfa[pat[j]][X]; // keep track of X state
        }
    }

    int search(string txt){
        int i, j, N = txt.length();
        // simulate operation of DFA on text, frame comes from brute force
            search v2
        for(i=0, j=0; i<N && j< M; i++)
            j = dfa[txt[i]][j]; // dfa simulation
        if(j==M) return i-M; // found
        return N; // not found
    }

    void print() {
        for(int i='a'; i<='z'; i++) {
            cout<<char(i)<<":";
            for(int j=0; j<M; j++)
                cout<< dfa[i][j];
```

```
                cout<<endl;
            }
        }

        int size(){return M;}

    };



    int main2() {
        string txt = "abababababcde";
        KMP k("ababcde");
        cout<<k.search(txt)<<endl;
        k.print();
        return 0;

    }


    int main() {

        string txt = "From 1820 to 1850, Jacksonian democracy began a set of
            reforms which included wider white male suffrage; it led to the rise of
            the Second Party System of Democrats and Whigs as the dominant parties
            from 1828 to 1854.";
        KMP k("white");
        k.print();
        int i =k.search(txt);
        cout<< i<<endl;
        cout<< txt.substr(i, k.size())<<endl;
        return 0;

    }


    // only on pat
    // 自己写的，思路比较复杂，不如algs4清晰，建议抛弃
    //    void buildDFA() {
    //        int X = 0;
    //        for(int j=1; j<M; j++){
    //            for(int c=0; c<R; c++){
    //                if(pat[j] == c) dfa[pat[j]][j] = j+1; // match
    //                else {
    //                    dfa[c][j] = dfa[c][X]; // dismatch
    //                    // 不能在这里更新X，X更新早了一步，应该是这轮结束时跟新，如果在这里更
        新，那么X实际上是下一轮的X
    //                }
    //            }
    //            X = dfa[pat[j]][X]; // keep track of X state,
    //
    //        }
    //    }

    // 这个版本也是在找问题，但也是错的，问题在于下面的j应该从1开始
```

```
//    void buildDFA2() {
//        int X = 0;
//        for(int j=0; j<M; j++){
//            for(int c=0; c<R; c++)
//                dfa[c][j] = dfa[c][X]; // dismatch
//            dfa[pat[j]][j] = j+1; // match
//            X = dfa[pat[j]][X]; // keep track of X state
//        }
//    }

/*
// 思路清晰， 先从状态X拷贝所有过来，再set match的那个值，最后更新X
void buildDFA() {
    dfa[pat[0]][0] = 1; // 因为buildDFA表的时候从1开始
    for(int X = 0, j=1; j<M; j++){ // 关键在这里j＝1，如果j＝0开始，的话在状态j＝1开始
        时X＝1，而X应该是0
        for(int c=0; c<R; c++)
            dfa[c][j] = dfa[c][X]; // dismatch
        dfa[pat[j]][j] = j+1; // match
        X = dfa[pat[j]][X]; // keep track of X state
    }
}

*/
```