

```
1: #include<iostream>
2: #include<vector>
3: #include<map>
4: #include<string>
5: #include<unordered_map>
6: #include<string.h>
7: #include<stdlib.h>
8:
9: using namespace std;
10:
11: bool s1[5000][5001] = {{false,false}};
12:
13: bool isP(string& str) {
14:     if(str.length() == 1) {
15:         return true;
16:     } else {
17:         return str[0] == str[str.length()-1];
18:     }
19: }
20:
21: void init(string& in) {
22:     int len = in.length();
23:     int lmax = len < 3 ? len : 3;
24:
25:     for(int i = 0; i < len; i++) {
26:         s1[i][0]=true;
27:     }
28:     for(int l = 1; l <= lmax; l++) {
29:         for(int pos = 0; pos <= (int)len - l; pos++) {
30:             string subStr = in.substr(pos, l);
31:             if(isP(subStr)) {
32:                 s1[pos][l]=true;
33:             }
34:         }
35:     }
36: }
37:
38: void dp(string& in) {
39:     int len = in.length();
40:
41:     for(int l = 4; l <= len; l++) {
42:         for(int pos=0; pos<=len-l; pos++) {
43:             s1[pos][l] = (s1[pos+2][l-4] && (in[pos] == in[pos+l-1]));
44:         }
45:     }
46: }
47:
48: typedef struct _ab { _ab* p; _ab* na; _ab* nb; int count; } AB;
49: AB* r;
50: void buildRTree(string& in) {
51:     r = new AB({0,0,0,0});
52:     int newA = 0;
53:     int newB = 0;
54:     for(int pos = 0; pos < (int) in.length(); pos++) {
55:         AB* c = r;
56:         for(int l = 1; l <= (int) in.length() - pos; l++) {
57:             if(in[pos+l-1] == 'a') {
58:                 if(!c->na) { c->na = new AB({c,0,0,0}); newA++; }
59:                 c = c->na;
60:                 if(s1[pos][l])
61:                     c->count++;
62:             } else {
63:                 if(!c->nb) { c->nb = new AB({c,0,0,0}); newB++; }
64:                 c = c->nb;
65:                 if(s1[pos][l])
66:                     c->count++;
67:             }
68:         }
69:     }
70:     //cout << newA << ", " << newB << endl;
71: }
72: string sol;
73: void traverse(AB* c, int& k) {
74:     //cout << sol << ',' << c->count << endl;
75:     if(k<=0) {
76:         return;
77:     }
78:     k -= c->count;
79:     if(k <= 0) {
80:         cout << sol << endl;
81:     }
82:     if(c->na) {
83:         sol.push_back('a');
84:         traverse(c->na, k);
85:         sol.pop_back();
86:     }
```

```
87:     if(c->nb) {
88:         sol.push_back('b');
89:         traverse(c->nb, k);
90:         sol.pop_back();
91:     }
92: }
93:
94: int main() {
95:     string in;
96:     int k;
97:     getline(cin, in);
98:     cin >> k;
99:
100:    init(in);
101:    dp(in);
102:    buildRTree(in);
103:
104:    AB* c = r;
105:    traverse(c, k);
106:    return 0;
107: }
```