

# Submission

[illegible]

Submission contains 1 file: [download zip archive](#)

FILENAME	FILESIZE	SHA-1 SUM	
suffixarrayreconstruction.cpp	2243 bytes	1997f7f35093b094366e07f253c60bede2a494c9	<a href="#">download</a>

[Edit and resubmit this submission.](#)

**suffixarrayreconstruction.cpp**

```

1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <algorithm>
5
6 struct Suffix {
7     int position;
8     std::string suffix;
9 };
10
11 const char PLACEHOLDER = '#';
12 const char WILDCARD = '*';
13
14 inline bool insert_suffix(std::string& target, Suffix& s) {
15     bool wildcard = false;
16     for (int suffix_pos = 0, target_pos = s.position;
17         target_pos < target.size();
18         suffix_pos++, target_pos++) {
19
20         if (s.suffix[suffix_pos] == WILDCARD) {
21             wildcard = true;
22             break;
23         }
24
25         if (target[target_pos] != PLACEHOLDER &&
26             target[target_pos] != s.suffix[suffix_pos])
27             return false;
28     }
29     else
30         target[target_pos] = s.suffix[suffix_pos];
31
32     if (wildcard) {
33         for (int suffix_pos = s.suffix.size() - 1, target_pos = target.size() - 1;
34             s.suffix[suffix_pos] != WILDCARD;
35             suffix_pos--, target_pos--) {
36
37             if (target[target_pos] != PLACEHOLDER &&
38                 target[target_pos] != s.suffix[suffix_pos])
39                 return false;
40         }
41         else
42             target[target_pos] = s.suffix[suffix_pos];
43     }
44 }
45
46 bool insert_suffix(std::string& target, Suffix& s) {
47     return insert_suffix(target, s);
48 }
49
50 int main() {
51     std::string target = "1234567890";
52     Suffix s1 = { 5, "1234567890"};
53     Suffix s2 = { 5, "1234567890*"};
54     Suffix s3 = { 5, "1234567890*"};
55     Suffix s4 = { 5, "1234567890*"};
56     Suffix s5 = { 5, "1234567890*"};
57     Suffix s6 = { 5, "1234567890*"};
58     Suffix s7 = { 5, "1234567890*"};
59     Suffix s8 = { 5, "1234567890*"};
60     Suffix s9 = { 5, "1234567890*"};
61     Suffix s10 = { 5, "1234567890*"};
62     Suffix s11 = { 5, "1234567890*"};
63     Suffix s12 = { 5, "1234567890*"};
64     Suffix s13 = { 5, "1234567890*"};
65     Suffix s14 = { 5, "1234567890*"};
66     Suffix s15 = { 5, "1234567890*"};
67     Suffix s16 = { 5, "1234567890*"};
68     Suffix s17 = { 5, "1234567890*"};
69     Suffix s18 = { 5, "1234567890*"};
70     Suffix s19 = { 5, "1234567890*"};
71     Suffix s20 = { 5, "1234567890*"};
72     Suffix s21 = { 5, "1234567890*"};
73     Suffix s22 = { 5, "1234567890*"};
74     Suffix s23 = { 5, "1234567890*"};
75     Suffix s24 = { 5, "1234567890*"};
76     Suffix s25 = { 5, "1234567890*"};
77     Suffix s26 = { 5, "1234567890*"};
78     Suffix s27 = { 5, "1234567890*"};
79     Suffix s28 = { 5, "1234567890*"};
80     Suffix s29 = { 5, "1234567890*"};
81     Suffix s30 = { 5, "1234567890*"};
82     Suffix s31 = { 5, "1234567890*"};
83     Suffix s32 = { 5, "1234567890*"};
84     Suffix s33 = { 5, "1234567890*"};
85     Suffix s34 = { 5, "1234567890*"};
86     Suffix s35 = { 5, "1234567890*"};
87     Suffix s36 = { 5, "1234567890*"};
88     Suffix s37 = { 5, "1234567890*"};
89     Suffix s38 = { 5, "1234567890*"};
90     Suffix s39 = { 5, "1234567890*"};
91     Suffix s40 = { 5, "1234567890*"};
92     Suffix s41 = { 5, "1234567890*"};
93     Suffix s42 = { 5, "1234567890*"};
94     Suffix s43 = { 5, "1234567890*"};
95     Suffix s44 = { 5, "1234567890*"};
96     Suffix s45 = { 5, "1234567890*"};
97     Suffix s46 = { 5, "1234567890*"};
98     Suffix s47 = { 5, "1234567890*"};
99     Suffix s48 = { 5, "1234567890*"};
100    Suffix s49 = { 5, "1234567890*"};
101    Suffix s50 = { 5, "1234567890*"};
102    Suffix s51 = { 5, "1234567890*"};
103    Suffix s52 = { 5, "1234567890*"};
104    Suffix s53 = { 5, "1234567890*"};
105    Suffix s54 = { 5, "1234567890*"};
106    Suffix s55 = { 5, "1234567890*"};
107    Suffix s56 = { 5, "1234567890*"};
108    Suffix s57 = { 5, "1234567890*"};
109    Suffix s58 = { 5, "1234567890*"};
110    Suffix s59 = { 5, "1234567890*"};
111    Suffix s60 = { 5, "1234567890*"};
112    Suffix s61 = { 5, "1234567890*"};
113    Suffix s62 = { 5, "1234567890*"};
114    Suffix s63 = { 5, "1234567890*"};
115    Suffix s64 = { 5, "1234567890*"};
116    Suffix s65 = { 5, "1234567890*"};
117    Suffix s66 = { 5, "1234567890*"};
118    Suffix s67 = { 5, "1234567890*"};
119    Suffix s68 = { 5, "1234567890*"};
120    Suffix s69 = { 5, "1234567890*"};
121    Suffix s70 = { 5, "1234567890*"};
122    Suffix s71 = { 5, "1234567890*"};
123    Suffix s72 = { 5, "1234567890*"};
124    Suffix s73 = { 5, "1234567890*"};
125    Suffix s74 = { 5, "1234567890*"};
126    Suffix s75 = { 5, "1234567890*"};
127    Suffix s76 = { 5, "1234567890*"};
128    Suffix s77 = { 5, "1234567890*"};
129    Suffix s78 = { 5, "1234567890*"};
130    Suffix s79 = { 5, "1234567890*"};
131    Suffix s80 = { 5, "1234567890*"};
132    Suffix s81 = { 5, "1234567890*"};
133    Suffix s82 = { 5, "1234567890*"};
134    Suffix s83 = { 5, "1234567890*"};
135    Suffix s84 = { 5, "1234567890*"};
136    Suffix s85 = { 5, "1234567890*"};
137    Suffix s86 = { 5, "1234567890*"};
138    Suffix s87 = { 5, "1234567890*"};
139    Suffix s88 = { 5, "1234567890*"};
140    Suffix s89 = { 5, "1234567890*"};
141    Suffix s90 = { 5, "1234567890*"};
142    Suffix s91 = { 5, "1234567890*"};
143    Suffix s92 = { 5, "1234567890*"};
144    Suffix s93 = { 5, "1234567890*"};
145    Suffix s94 = { 5, "1234567890*"};
146    Suffix s95 = { 5, "1234567890*"};
147    Suffix s96 = { 5, "1234567890*"};
148    Suffix s97 = { 5, "1234567890*"};
149    Suffix s98 = { 5, "1234567890*"};
150    Suffix s99 = { 5, "1234567890*"};
151    Suffix s100 = { 5, "1234567890*"};
152    Suffix s101 = { 5, "1234567890*"};
153    Suffix s102 = { 5, "1234567890*"};
154    Suffix s103 = { 5, "1234567890*"};
155    Suffix s104 = { 5, "1234567890*"};
156    Suffix s105 = { 5, "1234567890*"};
157    Suffix s106 = { 5, "1234567890*"};
158    Suffix s107 = { 5, "1234567890*"};
159    Suffix s108 = { 5, "1234567890*"};
160    Suffix s109 = { 5, "1234567890*"};
161    Suffix s110 = { 5, "1234567890*"};
162    Suffix s111 = { 5, "1234567890*"};
163    Suffix s112 = { 5, "1234567890*"};
164    Suffix s113 = { 5, "1234567890*"};
165    Suffix s114 = { 5, "1234567890*"};
166    Suffix s115 = { 5, "1234567890*"};
167    Suffix s116 = { 5, "1234567890*"};
168    Suffix s117 = { 5, "1234567890*"};
169    Suffix s118 = { 5, "1234567890*"};
170    Suffix s119 = { 5, "1234567890*"};
171    Suffix s120 = { 5, "1234567890*"};
172    Suffix s121 = { 5, "1234567890*"};
173    Suffix s122 = { 5, "1234567890*"};
174    Suffix s123 = { 5, "1234567890*"};
175    Suffix s124 = { 5, "1234567890*"};
176    Suffix s125 = { 5, "1234567890*"};
177    S
```

```
48
49 std::string solve(int length, std::vector<Suffix>& suffixes) {
50     std::string result;
51     result.resize(length, PLACEHOLDER);
52
53     for (Suffix& s : suffixes) {
54         if (!insert_suffix(result, s)) return "";
55     }
56
57     for (char c : result) {
58         if (c == PLACEHOLDER) return "";
59     }
60
61     return result;
62 }
63
64 int main() {
65     int test_count;
66     std::cin >> test_count;
67
68     while (test_count--) {
69         int length, suffix_count;
70         std::cin >> length >> suffix_count;
71
72         std::vector<Suffix> suffixes;
73         suffixes.reserve(suffix_count);
74
75         while (suffix_count--) {
76             Suffix s;
77             std::cin >> s.position >> s.suffix;
78             s.position--; // Make position zero indexed.
79             suffixes.push_back(s);
80         }
81
82         auto result = solve(length, suffixes);
83
84         if (result.empty()) printf("IMPOSSIBLE\n");
85         else printf("%s\n", result.c_str());
86     }
87 }
```