Submission

ID	DATE	PROBLEM	STATUS	CPU	LANG
	TEST CASES				
8192193	01:57:08	Life Forms	✔ **Accepted**	0.38 s	C++
	✔✔				

Submission contains 1 file: download zip archive

FILENAME	FILESIZE	SHA-1 SUM	
lifeforms.cpp	3674 bytes	`5ca32610cdb49beeee27d7354ec6bb18bdbfa208`	download

Edit and resubmit this submission.

lifeforms.cpp

```
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3
 4 const int maxsize = 100500;
 5 vector<int> _suffix_arr,_pos,_lcp,_map_string_color;
 6 int N,indx;
 7 string S;
 8
 9 bool comp(int a, int b){
10     if (_pos[a] != _pos[b])
11         return _pos[a] < _pos[b];
12     .ndx;
13     b+=indx;
```

Help

```
    return (a<N &&b<N)?_pos[a]<_pos[b]:a>b;
}

vector<int> suffix_array(){
    N = S.size();
    for (int i = 0; i < N; i++){
        _suffix_arr.push_back(i);
        _pos.push_back(S[i]);
    }
    int tmp[maxsize];
    for (indx = 1;; indx <<= 1) {
        sort(_suffix_arr.begin(), _suffix_arr.end(), comp);
        for (int i = 0; i < N-1; i++)
            tmp[i+1] = tmp[i] + comp(_suffix_arr[i], _suffix_arr[i+1]);
        for (int i = 0; i < N; i++)
            _pos[_suffix_arr[i]] = tmp[i];
        if (tmp[N - 1] == N - 1)
            break;
    }
    return _suffix_arr;
}

vector<int> lcp(){
    vector<int>__lcp(N);
    for (int i=0,k=0; i<N;i++){
        if (_pos[i] != N-1){
            int j = _suffix_arr[_pos[i] + 1];
            while (S[i + k] == S[j + k])
                k++;
            __lcp[_pos[i]+1] = k;
            if (k) k--;
        }
    }
    _lcp=__lcp;
    return __lcp;
}

int n_strings;

bool LCS_size_l(int l,bool print){
         counted[n_strings];
    Help count;
    for (int i = 1; i < N; ++i) {
        if (_lcp[i] >= l) {
```

```
58              for(int _i=0;_i<n_strings;_i++) counted[_i] = false;
59              count = 1;
60              counted[_map_string_color[_suffix_arr[i-1]]] = true;
61              int j = i;
62              while (j < N && _lcp[j] >= l) {
63                  if (_map_string_color[_suffix_arr[j]] != _map_string_color[_suffix_arr[j]+l-1]){
64                      break;
65                  }
66                  if (!counted[_map_string_color[_suffix_arr[j]]]) {
67                      counted[_map_string_color[_suffix_arr[j]]] = true;
68                      ++count;
69                  }
70                  ++j;
71              }
72              if (count > n_strings/2) {
73                  if(print){
74                      cout<<S[_suffix_arr[i]];
75                      for (int k = 1; k < l; ++k)
76                          cout<<S[_suffix_arr[i]+k];
77                      cout<<endl;
78                  }
79                  else return true;
80              }
81              i = j-1;
82          }
83      }
84      return false;
85  }
86
87  int LCS(){
88      vector<string> entrada;
89      int min_string_size=maxsize;
90      S="";
91
92      for(int i=0;i<n_strings;i++){
93          string temp;
94          cin>>temp;
95          for(int j=0;j<temp.size();j++){
96              S+=temp[j];
97              _map_string_color.push_back(i);
98          }
99          if(temp.size()<min_string_size)
100             min_string_size=temp.size();
101         S+=123+i;
```

```
102         _map_string_color.push_back(0);
103     }
104     S.pop_back();
105     _map_string_color.pop_back();
106     suffix_array();
107     lcp();
108 
109     int min=1,_ans=-1;
110     while(min_string_size >= min){
111         int  mid = (min_string_size-min)/2 + min;
112         if(LCS_size_l(mid,false)){
113             min=mid+1;
114             _ans=max(_ans,mid);
115         }
116         else
117             min_string_size = mid - 1;
118     }
119     if(_ans==-1){
120         cout<<"?";
121         return _ans;
122     }
123     LCS_size_l(_ans,true);
124     return _ans;
125 }
126 
127 
128 int main(){
129 
130     n_strings=-1;
131 
132     while(true){
133         cin>>n_strings;
134         if(n_strings==0) return 0;
135         LCS();
136         cout<<"\n";
137 
138         //clear buffer
139         S.clear();
140         N = 0;
141         _suffix_arr.clear();
142         _pos.clear();
143         _lcp.clear();
144         _map_string_color.clear();
145     }
```

```
146
147     return 0;
148 }
```