1 Aliens

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <climits>
5  using namespace std;
6
7  typedef vector<pair<int, int>> vii;       // sorted by left, right.
8  bool sortDescAsc(const pair<int, int>& lhs, const pair<int, int>& ↙
       ↳ rhs) {
9      if(lhs.first == rhs.first)
10         return (lhs.second > rhs.second);
11     else
12         return lhs.first < rhs.first;
13 }
14
15 void testcase() {
16     int n, m;
17     cin >> n >> m;
18     vii intervals;
19     int superior = n;
20     for(int i = 0; i < n; ++i) {
21         int pi, qi;
22         cin >> pi >> qi;
23         if(pi == 0 && qi == 0) {
24             --superior;
25             continue;
26         }
27         pair<int, int> entry = make_pair(pi, qi);
28         intervals.push_back(entry);
29     }
30
31     sort(intervals.begin(), intervals.end(), sortDescAsc);
32
33     int left = 0;
34     int right = 0;
35     for(int i = 0; i < intervals.size(); ++i) {
36         if(i+1 < intervals.size() && intervals[i+1].first == ↙
               ↳ intervals[i].first && intervals[i+1].second == ↙
               ↳ intervals[i].second)
37             --superior;
38         else if(left == intervals[i].first && right == ↙
               ↳ intervals[i].second)
```

```
39              --superior;
40          else if(right >= intervals[i].second)
41              --superior;
42
43          if(right < intervals[i].second) {
44              left = intervals[i].first;
45              if(right != 0 && left-right > 1) {
46                  cout << "0\n";
47                  return;
48              }
49              right = intervals[i].second;
50          }
51      }
52
53      cout << superior << "\n";
54  }
55
56  int main() {
57      int TC;
58      cin >> TC;
59      while(TC--) testcase();
60  }
```

2 Checking Change

```
1  #include <vector>
2  #include <iostream>
3  #include <algorithm>
4  #include <string>
5  #include <sstream>
6  using namespace std;
7
8  vector<string> answers;
9
10 int main(int argc, char const *argv[])
11 {
12
13         int currencies;
14         cin >> currencies;
15
16         for (int currency = 0; currency < currencies; currency++)
17         {
18
19                 int coins_count;
20                 int testcases;
21
22                 cin >> coins_count >> testcases;
23
24                 vector<int> coins;
```

```
25              for (int coins_it = 0; coins_it < coins_count; ↙
                    ↳ coins_it++)
26              {
27                      int coin;
28                      cin >> coin;
29                      coins.push_back(coin);
30              }
31
32              vector<int> tests;
33              for (int testcase = 0; testcase < testcases; ↙
                    ↳ testcase++)
34              {
35                      int test;
36                      cin >> test;
37                      tests.push_back(test);
38              }
39
40              // find maximum of tests
41              vector<int>::iterator max_test_it = ↙
                    ↳ max_element(tests.begin(), tests.end());
42              int max_test = *max_test_it;
43              int N = max_test + 1;
44
45              vector<int>::iterator max_coin_it = ↙
                    ↳ max_element(coins.begin(), coins.end());
46              int max_coin = *max_coin_it;
47
48              vector<int>::iterator min_coin_it = ↙
                    ↳ min_element(coins.begin(), coins.end());
49              int min_coin = *min_coin_it;
50
51              // instantiate array with size max(tests)
52              int arraysize = 2;
53              vector<int> counts(arraysize);
54
55              // fill indices we already know -> coins, set to ↙
                    ↳ zero where index smaller than index of ↙
                    ↳ smallest coin.
56              for (int i = 0; i < min_coin; i++)
57              {
58                      if (min_coin >= arraysize)
59                      {
60                              arraysize += min_coin + 10;
61                              counts.resize(arraysize);
62                              //cout << "vector size now " << ↙
                                    ↳ arraysize;
63                      }
64                      counts[i] = 0;
65              }
```

```
66
67                  for (vector<int>::iterator coins_it = coins.begin(); ↙
                        ↳ coins_it != coins.end(); coins_it++)
68                  {
69                          if (*coins_it <= max_coin)
70                          {
71                                  if (*coins_it >= arraysize)
72                                  {
73                                          arraysize += *coins_it + 1;
74                                          counts.resize(arraysize);
75                                          //cout << "vector size now " ↙
                                               ↳ << arraysize;
76                                  }
77                                  counts[*coins_it] = 1;
78                          }
79                  }
80
81                  // iterate over counts, combine all minimums.
82                  for (int n = min_coin + 1; n < N; n++)
83                  {
84                          if (arraysize <= n)
85                          {
86                                  arraysize += 1;
87                                  counts.resize(arraysize);
88                                  //cout << "vector size now " << ↙
                                       ↳ arraysize;
89                          }
90
91                          signed int min = -1;
92                          for(int backward = n-1; backward >= ↙
                               ↳ min_coin; backward--) {
93
94                                  if (counts[n] == 1)
95                                  {
96                                          min = 1;
97                                  } else {
98                                          if(counts[backward] != 0 && ↙
                                               ↳ counts[n-backward] != ↙
                                               ↳ 0) {
99                                                  int new_min = ↙
                                                       ↳ counts[backward] ↙
                                                       ↳ + ↙
                                                       ↳ counts[n-backward];
100                                                 //cout << n << ": ↙
                                                       ↳ counts[backward]: ↙
                                                       ↳ " << ↙
                                                       ↳ counts[backward] ↙
                                                       ↳ << " ↙
                                                       ↳ counts[n-backward]: ↙
```

```
                                                        ↪ " << ↙
                                                        ↪ counts[n-backward] ↙
                                                        ↪ << "new_min: ↙
                                                        ↪ "<< new_min << ↙
                                                        ↪ "\n";
101                                                     if (min > new_min || ↙
                                                        ↪ min == -1)
102                                                     {
103                                                             min = new_min;
104                                                     }
105                                             }
106                                     }
107
108                             }
109                             if (min == -1)
110                             {
111                                     min = 0;
112                             }
113                             counts[n] = min;
114                     }
115
116                     /*int i = 0;
117                     for (vector<int>::iterator elements = ↙
                        ↪ counts.begin(); elements != counts.end(); ↙
                        ↪ elements++)
118                     {
119                             cout << i++ << ": " << *elements << " \n";
120                     }*/
121
122                     for (vector<int>::iterator test = tests.begin(); ↙
                        ↪ test != tests.end(); test++)
123                     {
124                             int answer = counts[*test];
125
126                             stringstream ss;
127                             if (answer == 0)
128                             {
129                                     ss << "not possible";
130                             } else {
131                                     ss << answer;
132                             }
133
134                             answers.push_back(ss.str());
135                     }
136
137             }
138
139             for (vector<string>::iterator answer = answers.begin(); ↙
                    ↪ answer != answers.end(); answer++)
```

```
140                     cout << *answer << "\n";
141
142             return 0;
143     }
```