topic	problem	name	author	location
ACM	*Given a set of intervals...*	Aliens	Ben	3
	Given ...	Checking Change	Ben	**??**

1 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 }
```

2 Dominoes

```
1  /*
2  * Benjamin Grhbiel
3  * Domino
4  */
5
6  #include <iostream>
7  #include <vector>
8  #include <map>
9  using namespace std;
10
11 int main (int argc, const char *argv[])
```

```
12 {
13
14   ios_base::sync_with_stdio(false);
15
16   int testcases;
17   cin >> testcases;
18
19   map< int, vector<int> > index;
20
21   for (int testcase = 0; testcase < testcases; testcase++) {
22
23     long int dominoes;
24     cin >> dominoes;
25
26     for (int dominoPos = 1; dominoPos <= dominoes; dominoPos++) {
27       int height;
28       cin >> height;
29       index[testcase].push_back(height);
30     }
31
32   }
33
34   for (map<int, vector<int> >::iterator it = index.begin(); it != ↙
       ↪ index.end(); it++) {
35       //cout << "Testcase: " << it->first << " Tiles: " << ↙
           ↪ it->second.size() << "\n";
36
37       vector<int> tiles = it->second;
38
39       if (tiles.size() == 0) {
40         cout << 0;
41       }
42       else
43       {
44         int intervalRight = 0;
45         int iteration = 0;
46         int counter = 0;
47
48         for (vector<int>::iterator tile_it = tiles.begin(); tile_it ↙
             ↪ != tiles.end(); tile_it++) {
49
50           if (iteration > intervalRight) {
51               //cout << "Break; iteration > intervalRight \n";
52               break;
53           }
54
55           int h = *tile_it;
56           int newIntervalRight = h + iteration - 1;
57
```

```
58            if(newIntervalRight > intervalRight) {
59              intervalRight = newIntervalRight;
60            }
61
62            iteration++;
63            //cout << "intervalRight: " << intervalRight << " ↙
                  ↳ iteration: " << iteration << "\n";
64            counter++;
65          }
66
67          cout << counter << "\n";
68
69        }
70    }
71
72    return 0;
73
74  }
```

3 Shelves

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(void) {
6          // speeds up read and write
7          ios_base::sync_with_stdio(false);
8
9          // number of testcases we need to run
10         int nrCases;
11         cin >> nrCases;
12
13         for(int i = 0; i < nrCases; i++) {
14                 // read the input for the test case
15                 int l, m, n;
16                 cin >> l >> m >> n;
17
18                 // number of the two shelves and remaining length
19                 int cm = 0;
20                 int cn = 0;
21                 int r = l;
22
23                 for(int tmpCn = l/n; tmpCn >= 0 && r != 0; tmpCn--) {
24                         // calculate the number of the small shelves
25                         int tmpCm  = (l - tmpCn * n) / m;
26                         if(tmpCm >= n) {
27                                 break;
28                         }
```

```
29
30                              // calculate the new remaining space and use ↙
                                    ↳ it when smaller
31                              int tmpR = l - tmpCn * n - tmpCm * m;
32                              if(tmpR < r) {
33                                      cn = tmpCn;
34                                      cm = tmpCm;
35                                      r = tmpR;
36                              }
37                      }
38
39                      // output the result
40                      cout << cm << " " << cn << " " << r << '\n';
41              }
42
43              return 0;
44      }
```

Even Pairs missing