Contents

1 Checking Change

1.1 Description

1.2 Solution

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <string>
#include <sstream>
using namespace std;

vector<string> answers;

int main(int argc, char const *argv[])
{

    int currencies;
    cin >> currencies;

    for (int currency = 0; currency < currencies; currency++)
    {

        int coins_count;
        int testcases;

        cin >> coins_count >> testcases;

        vector<int> coins;
        for (int coins_it = 0; coins_it < coins_count; coins_it++)
        {
            int coin;
            cin >> coin;
            coins.push_back(coin);
        }

        vector<int> tests;
        for (int testcase = 0; testcase < testcases; testcase++)
        {
            int test;
            cin >> test;
            tests.push_back(test);
        }

        // find maximum of tests
        vector<int>::iterator max_test_it = max_element(tests.begin(), tests.end());
        int max_test = *max_test_it;
        int N = max_test + 1;

        vector<int>::iterator max_coin_it = max_element(coins.begin(), coins.end());
        int max_coin = *max_coin_it;

        vector<int>::iterator min_coin_it = min_element(coins.begin(), coins.end());
        int min_coin = *min_coin_it;

        // instantiate array with size max(tests)
        int arraysize = 2;
        vector<int> counts(arraysize);

        // fill indices we already know -> coins, set to zero where index smaller than index ↙
            ↳ of smallest coin.
        for (int i = 0; i < min_coin; i++)
        {
            if (min_coin >= arraysize)
            {
                arraysize += min_coin + 10;
                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

```
/*
* Benjamin Grhbiel
* Domino
*/

#include <iostream>
#include <vector>
#include <map>
using namespace std;

int main (int argc, const char *argv[])
{

  ios_base::sync_with_stdio(false);

  int testcases;
  cin >> testcases;

  map< int, vector<int> > index;

  for (int testcase = 0; testcase < testcases; testcase++) {

    long int dominoes;
    cin >> dominoes;

    for (int dominoPos = 1; dominoPos <= dominoes; dominoPos++) {
      int height;
      cin >> height;
      index[testcase].push_back(height);
    }

  }

  for (map<int, vector<int> >::iterator it = index.begin(); it != index.end(); it++) {
      //cout << "Testcase: " << it->first << " Tiles: " << it->second.size() << "\n";

      vector<int> tiles = it->second;

      if (tiles.size() == 0) {
        cout << 0;
      }
      else
      {
        int intervalRight = 0;
        int iteration = 0;
        int counter = 0;

        for (vector<int>::iterator tile_it = tiles.begin(); tile_it != tiles.end(); tile_it++) {

          if (iteration > intervalRight) {
              //cout << "Break; iteration > intervalRight \n";
              break;
          }

          int h = *tile_it;
          int newIntervalRight = h + iteration - 1;

          if(newIntervalRight > intervalRight) {
            intervalRight = newIntervalRight;
          }

          iteration++;
          //cout << "intervalRight: " << intervalRight << " iteration: " << iteration << "\n";
          counter++;
        }

```

```
67            cout << counter << "\n";
68
69        }
70    }
71
72    return 0;
73
74 }
```

3 Shelves

```cpp
#include <iostream>

using namespace std;

int main(void) {
    // speeds up read and write
    ios_base::sync_with_stdio(false);

    // number of testcases we need to run
    int nrCases;
    cin >> nrCases;

    for(int i = 0; i < nrCases; i++) {
        // read the input for the test case
        int l, m, n;
        cin >> l >> m >> n;

        // number of the two shelves and remaining length
        int cm = 0;
        int cn = 0;
        int r = l;

        for(int tmpCn = l/n; tmpCn >= 0 && r != 0; tmpCn--) {
            // calculate the number of the small shelves
            int tmpCm  = (l - tmpCn * n) / m;
            if(tmpCm >= n) {
                break;
            }

            // calculate the new remaining space and use it when smaller
            int tmpR = l - tmpCn * n - tmpCm * m;
            if(tmpR < r) {
                cn = tmpCn;
                cm = tmpCm;
                r = tmpR;
            }
        }

        // output the result
        cout << cm << " " << cn << " " << r << '\n';
    }

    return 0;
}
```

4 Even Pairs

Even Pairs missing