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

1 Checking Change

```
#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);
                        //cout << "vector size now " << arraysize;
                }
                counts[i] = 0;
        }

        for (vector<int>::iterator coins_it = coins.begin(); coins_it != coins.end(); coins_it++)
        {
                if (*coins_it <= max_coin)
                {
                        if (*coins_it >= arraysize)
                        {
                                arraysize += *coins_it + 1;
                                counts.resize(arraysize);
                                //cout << "vector size now " << arraysize;
                        }
                        counts[*coins_it] = 1;
                }
        }

        // iterate over counts, combine all minimums.
        for (int n = min_coin + 1; n < N; n++)
        {
            if (arraysize <= n)
            {
                arraysize += 1;
                counts.resize(arraysize);
                //cout << "vector size now " << arraysize;
            }

            signed int min = -1;
            for(int backward = n-1; backward >= min_coin; backward--) {

                if (counts[n] == 1)
                {
                    min = 1;
                } else {
                    if(counts[backward] != 0 && counts[n-backward] != 0) {
                        int new_min = counts[backward] + counts[n-backward];
                        //cout << n << ": counts[backward]: " << counts[backward] << " counts[n-backward]: " << counts[n-backward] << "new_min: "<< new_min << "\n";
                        if (min > new_min || min == -1)
                        {
                            min = new_min;
                        }
                    }
                }

            }
            if (min == -1)
            {
                min = 0;
            }
            counts[n] = min;
        }

        /*int i = 0;
        for (vector<int>::iterator elements = counts.begin(); elements != counts.end(); elements++)
        {
            cout << i++ << ": " << *elements << " \n";
        }*/

        for (vector<int>::iterator test = tests.begin(); test != tests.end(); test++)
        {
            int answer = counts[*test];

            stringstream ss;
            if (answer == 0)
            {
                ss << "not possible";
            } else {
                ss << answer;
            }

            answers.push_back(ss.str());
        }

    }

    for (vector<string>::iterator answer = answers.begin(); answer != answers.end(); answer++)
        cout << *answer << "\n";

    return 0;
}
```

2 Dominoes

```
/*
* Benjamin Gr hbiel
* 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++;
          }

          cout << counter << "\n";

        }
    }

    return 0;

}
```

3 Shelves

```
#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;
}
```

Even Pairs missing