Filling the Bucket

Amir Kirsh

```
1.  pair<bool, map<size_t, size_t>>
2.  canFill(size_t big_bucket, const vector<size_t>& small_buckets, size_t index = 0) {
3.    if(big_bucket == 0) return {true, {}};
4.    if(big_bucket < small_buckets.back()) return {false, {}};
5.    auto curr = small_buckets[index];
6.    if(big_bucket % curr == 0) return {true, { {curr, big_bucket / curr} }};
7.    if(index < small_buckets.size() - 1) {
8.      auto times = big_bucket / curr + 1;
9.      do {
10.       --times;
11.       auto rest = big_bucket - times * curr;
12.       auto result = canFill(rest, small_buckets, index + 1);
13.       if(result.first) {
14.         result.second[curr] = times;
15.         return {true, result.second};
16.       }
17.     } while(times > 0);
18.   }
19.   return {false, {}};
20. }
```

1. What will be printed?

```
std::cout << canFill(12, {4}).first << '\n';
```

A 1

B 0

C the code will not compile

D 3

1. What will be printed?

```
std::cout << canFill(12, {4}).first << '\n';
```

A	1	**C**	the code will not compile
B	0	**D**	3

2. What will be printed?

```
std::cout << canFill(12, {5, 7, 9}).first << '\n';
```

A 1

B 0

C the code will not compile

D 3

2. What will be printed?

```
std::cout << canFill(12, {5, 7, 9}).first << '\n';
```

A 1

B 0

C the code will not compile

D 3

3. How many times *‘canFill’* is called?

```
std::cout << canFill(12, {9, 7, 5}).first << '\n';
```

A	1	**C**	4
B	3	**D**	5

```
pair<bool, map<size_t, size_t>>
canFill(size_t big_bucket, const vector<size_t>& small_buckets, size_t index = 0) {
  if(big_bucket == 0) return {true, {}};
  if(big_bucket < small_buckets.back()) return {false, {}};
  auto curr = small_buckets[index];
  if(big_bucket % curr == 0) return {true, { {curr, big_bucket / curr} }};
  if(index < small_buckets.size() - 1) {
    auto times = big_bucket / curr + 1;
    do {
      --times;
      auto rest = big_bucket - times * curr;
      auto result = canFill(rest, small_buckets, index + 1);
      if(result.first) {
        result.second[curr] = times;
        return {true, result.second};
      }
    } while(times > 0);
  }
  return {false, {}};
}
```

3. How many times *'canFill'* is called?

```
std::cout << canFill(12, {9, 7, 5}).first << '\n';
```

A	1	**C**	4
B	3	**D**	5

Thank you!