Practical 3

18BCE243

Code of Practical 3

```
#include <algorithm>
#include <vector>

static bool __all_zeros(const std::vector<int> &degrees, const int &i, const int &j)
{
    for ( int k=i ; k<=j ; k++ ) if ( degrees[k] != 0 ) return false;
    return true;
}

bool havel_hakimi(const std::vector<int> &degrees)
{
    if ( degrees.size() == 0 ) return true;
    for ( size_t k=0 ; k<degrees.size() ; k++ ) if ( degrees[k] < 0 ) return false;

    std::vector<int> degcopy = degrees;                  // [1, 1, 2, 2, 5, 5, 6]

    if ( !std::is_sorted(degcopy.begin(), degcopy.end()) ) {
        std::sort(degcopy.begin(), degcopy.end());
    }
    int i = degcopy.size() - 1;                          // 6
    int last = degcopy[i];                               // 6
    do {
        if ( i-last < 0 ) return false;                  // 0
        for ( int j=i-1 ; j>=i-last ; j-- ) {            // j=5 ; j>=0 ; j--
            if ( degcopy[j] <= 0 ) return false;         // skipped...
            degcopy[j]--;                                // [0, 0, 1, 1, 4, 4]
        }
        i--;                                             // 5
        std::sort(degcopy.begin(), degcopy.begin()+i+1); // 0, 6 ==> [0, 0, 1, 1, 4, 4, (6)]
        last = degcopy[i];                               // 4
    } while ( !__all_zeros(degcopy, 0, i) );             // 0, 5 ==> false
    return true;
}
```

Test Driver (with Inputs)

```
#include <iostream>
#include "practical3.h"

int main()
```

```
{
    std::vector<std::vector<int> > degrees = {
                                                {1, 1, 1, 1, 1},            // 1.  does NOT
                                                {1, 1, 1, 1, 1, 1},         // 2.  does
                                                {1, 1},                     // 3.  does
                                                {1, 1, 1},                  // 4.  does NOT
                                                {-1, -1, -1},               // 5.  does NOT
                                                {0, 0, 0, 0, 0},            // 6.  does
                                                {0, 0, 0, 0, 0, 0},         // 7.  does
                                                {1, 1, 2, 2, 5, 5, 6},      // 8.  does NOT
                                                {3, 3, 3, 3},               // 9.  does
                                                {},                         // 10. does
                                                {0},                        // 11. does
                                                {1},                        // 12. does NOT
                                                {-1, 1, 1, 1, 3},           // 13. does NOT
                                                {1, 1, 1, 3, 2, 2, 2, 2}    // 14. does
                                               };

    for ( size_t i=0 ; i<degrees.size() ; i++ )
        std::cout << i+1 \
                  << ". The given sequence " \
                  << (havel_hakimi(degrees[i]) ? "does" : "does NOT") \
                  << " constitute a graph!" << std::endl;

    return 0;
}
```

Output

```
1. The given sequence does NOT constitute a graph!
2. The given sequence does constitute a graph!
3. The given sequence does constitute a graph!
4. The given sequence does NOT constitute a graph!
5. The given sequence does NOT constitute a graph!
6. The given sequence does constitute a graph!
7. The given sequence does constitute a graph!
8. The given sequence does NOT constitute a graph!
9. The given sequence does constitute a graph!
10. The given sequence does constitute a graph!
11. The given sequence does constitute a graph!
12. The given sequence does NOT constitute a graph!
13. The given sequence does NOT constitute a graph!
14. The given sequence does constitute a graph!
```