Jesse and Cookies

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'cookies' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER k
 *  2. INTEGER_ARRAY A
 */

int cookies(int k, vector<int> A) {
    priority_queue<int, vector<int>, greater<int>> pq(A.begin(),
A.end());
    int ops = 0;

    while (!pq.empty() && pq.top() < k) {
        if (pq.size() < 2) return -1;
        int first = pq.top(); pq.pop();
        int second = pq.top(); pq.pop();
        int newCookie = first + 2 * second;
        pq.push(newCookie);
        ops++;
    }
    return ops;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string first_multiple_input_temp;
    getline(cin, first_multiple_input_temp);

    vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));
```

```
    int n = stoi(first_multiple_input[0]);

    int k = stoi(first_multiple_input[1]);

    string A_temp_temp;
    getline(cin, A_temp_temp);

    vector<string> A_temp = split(rtrim(A_temp_temp));

    vector<int> A(n);

    for (int i = 0; i < n; i++) {
        int A_item = stoi(A_temp[i]);

        A[i] = A_item;
    }

    int result = cookies(k, A);

    fout << result << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );
```

```
    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```