#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);

string rtrim(const string &);

vector<string> split(const string &);

vector<int> rotateLeft(int d, vector<int> arr) {

int n = arr.size();

vector<int> rotatedArr(n);

for (int i = 0; i < n; i++) {

rotatedArr[i] = arr[(i + d) % n];

}

return rotatedArr;

}

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 d = stoi(first\_multiple\_input[1]);

string arr\_temp\_temp;

getline(cin, arr\_temp\_temp);

vector<string> arr\_temp = split(rtrim(arr\_temp\_temp));

vector<int> arr(n);

for (int i = 0; i < n; i++) {

int arr\_item = stoi(arr\_temp[i]);

arr[i] = arr\_item;

}

vector<int> result = rotateLeft(d, arr);

for (size\_t i = 0; i < result.size(); i++) {

fout << result[i];

if (i != result.size() - 1) {

fout << " ";

}

}

fout << "\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;

}