Name	Soruce code

ints_cmp	#include <checkers/testlib.h>
	
	using namespace NTestlib;
	
	int main(int argc, char* argv[]) {
	InitChecker(argc, argv);
	
	int i;
	for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
	i64 x = Ans.ReadInt();
	i64 y = Out.ReadInt();
	if (x != y) {
	std::stringstream msg;
	msg << i + 1 << " integer don't match. Expected: " << x << ", got: " << y;
	QuitWith(WA, msg.str());
	}
	}
	if (Out.HasInput()) {
	QuitWith(PE, "solution has more data than expected");
	}
	if (Ans.HasInput()) {
	QuitWith(PE, "solution has less data than expected");
	}
	std::stringstream msg;
	msg << "Ok. " << i << " integers equal.";
	QuitWith(AC, msg.str());
	}

words_cmp	#include <checkers/testlib.h>
	
	using namespace NTestlib;
	
	int main(int argc, char* argv[]) {
	InitChecker(argc, argv);
	
	int i;
	for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
	std::string x = Ans.ReadWord();
	std::string y = Out.ReadWord();
	if (x != y) {
	std::stringstream msg;
	msg << i + 1 << " word don't match. Expected: " << x << ", got: " << y;
	QuitWith(WA, msg.str());
	}
	}
	if (Out.HasInput()) {
	QuitWith(PE, "solution has more data than expected");
	}
	if (Ans.HasInput()) {
	QuitWith(PE, "solution has less data than expected");
	}
	std::stringstream msg;
	msg << "Ok. " << i << " tokens equal.";
	QuitWith(AC, msg.str());
	}

doubles3_cmp

```
#include <checkers/testlib.h>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
  InitChecker(argc, argv);

  int i;
  for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
  double x = Ans.ReadDouble();
  double y = Out.ReadDouble();
  if (!DoubleEq(y, x, 1e-3)) {
  std::stringstream msg;
  msg << i + 1 << " doubles don't match. Expected: " << x << ", got: " << y;
  msg << ". Error: " << std::min(AbsError(x, y), RelError(y, x));
  QuitWith(WA, msg.str());
  }
  }
  if (Out.HasInput()) {
  QuitWith(PE, "solution has more data than expected");
  }
  if (Ans.HasInput()) {
  QuitWith(PE, "solution has less data than expected");
  }
  std::stringstream msg;
  msg << "Ok. " << i << " tokens equal.";
  QuitWith(AC, msg.str());
 }
```

doubles6_cmp

```
#include <checkers/testlib.h>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
  InitChecker(argc, argv);

  int i;
  for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
  double x = Ans.ReadDouble();
  double y = Out.ReadDouble();
  if (!DoubleEq(y, x, 1e-6)) {
  std::stringstream msg;
  msg << i + 1 << " doubles don't match. Expected: " << x << ", got: " << y;
  msg << ". Error: " << std::min(AbsError(x, y), RelError(y, x));
  QuitWith(WA, msg.str());
  }
  }
  if (Out.HasInput()) {
  QuitWith(PE, "solution has more data than expected");
  }
  if (Ans.HasInput()) {
  QuitWith(PE, "solution has less data than expected");
  }
  std::stringstream msg;
  msg << "Ok. " << i << " tokens equal.";
  QuitWith(AC, msg.str());
 }
```

doubles7_cmp

```
#include <checkers/testlib.h>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
  InitChecker(argc, argv);

  int i;
  for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
  double x = Ans.ReadDouble();
  double y = Out.ReadDouble();
  if (!DoubleEq(y, x, 1e-7)) {
  std::stringstream msg;
  msg << i + 1 << " doubles don't match. Expected: " << x << ", got: " << y;
  msg << ". Error: " << std::min(AbsError(x, y), RelError(y, x));
  QuitWith(WA, msg.str());
  }
  }
  if (Out.HasInput()) {
  QuitWith(PE, "solution has more data than expected");
  }
  if (Ans.HasInput()) {
  QuitWith(PE, "solution has less data than expected");
  }
  std::stringstream msg;
  msg << "Ok. " << i << " tokens equal.";
  QuitWith(AC, msg.str());
 }
```

compilation

```
#include <checkers/testlib.h>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
  InitChecker(argc, argv);

  std::stringstream msg;
  msg << "Ok";
  QuitWith(AC, msg.str());
 }
```

51

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <string>
 #include <vector>
 #include <algorithm>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 std::string out = "", answer = "";
 //File - programm input (from .in file)
 //Out - User output
 //Ans - Correct output (from .out file)
 out = Out.ReadLine();
 answer = Ans.ReadLine();
 while(out[out.size() -1] == 32)
 out = out.substr(0, out.size() - 1);
 while(answer[answer.size() -1] == 32)
 answer = answer.substr(0, answer.size() - 1);
 while(out[0] == 32)
 out = out.substr(1, out.size());
 while(answer[0] == 32)
 answer = answer.substr(1, answer.size());
 if (out==answer)
 QuitWith(AC, "Full solution");
 std::stringstream msg;
 msg << "Out " << out.c_str() << ", Correct answer: " << answer.c_str();
 QuitWith(WA, msg.str());
 }
```

```cpp
#include <checkers/testlib.h>
#include <iostream>
#include <string>
#include <vector>


using namespace NTestlib;

int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n = File.ReadInt();
int sum = File.ReadInt();
std::vector <int> a;
for (int i = 0; i < n; i++)
a.push_back(File.ReadInt());
int answer = Ans.ReadInt();
if(answer == -1) {
Out.ReadInt(-1, -1);
QuitWith(AC, "Correct output");
}
int i1 = Out.ReadInt(1, n), i2 = Out.ReadInt(1,n);
if(i1 == i2) {
QuitWith(WA,"First variable equal second");
}
if(a[i1-1] + a[i2-1] != sum) {
std::stringstream msg;
msg << "Incorrect sum. Get " << a[i1-1] + a[i2-1] << ". Correct answer: " << sum;
QuitWith(WA, msg.str());
}
QuitWith(AC, "Correct output");
}
```

```
#include <checkers/testlib.h>
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>

using namespace NTestlib;

int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n = File.ReadInt();
std::vector <int> a;
std::vector <int> b;
for (int i = 0; i < n; i++)
{
int x = File.ReadInt();
a.push_back(x);
b.push_back(x);
}
int ind1, ind2, cnt1 = 0, cnt2 = 0, max1 = 0, max2 = 0;
ind1 = Out.ReadInt(1, n);
ind2 = Ans.ReadInt();
a[ind1-1] = 1;
b[ind2-1] = 1;
for (int i = 0; i < n; i++)
{
if (a[i] == 0)
{
max1 = std::max(max1, cnt1);
cnt1 = 0;
}
else
cnt1++;
}
max1 = std::max(max1, cnt1);
for (int i = 0; i < n; i++)
{
if (b[i] == 0)
{
max2 = std::max(max2, cnt2);
cnt2 = 0;
}
else
cnt2++;
}
max2 = std::max(max2, cnt2);
if (max1 < max2)
{
std::stringstream msg;
msg << "User max value " << max1 << " less than correct max value " << max2;
QuitWith(WA, msg.str());
}
if (max1 == max2)
QuitWith(AC,"OK");
if (max1 > max2)
{
std::stringstream msg;
msg << "STUDENT SOLUTION RESULT " << max1 << " BETTER THAN SYSTEM " << max2;
QuitWith(EF, msg.str());
}
}
```

70

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector <int> a;
 for (int i = 0; i < n; i++)
 {
 a.push_back(File.ReadInt());
 }
 int ind = Out.ReadInt(1, n);
 long long int cnt1 = 0, cnt2 = 0;
 for (int i = 0; i < ind-1; i++) {
 cnt1+=a[i];
 }
 for (int i = ind; i < n; i++) {
 cnt2+=a[i];
 }
 if (cnt1 == cnt2)
 QuitWith(AC,"OK");
 std::stringstream msg;
 msg << "First part: " << cnt1 << "; Second part: "<<cnt2;
 QuitWith(WA, msg.str());
 }
```

72

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <string>
 #include <sstream>

 using namespace NTestlib;
 std::string inttostring(int Number)
 {
 std::stringstream convert;
 convert << Number;
 return convert.str();
 }
 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 std::multiset <int> a;
 std::set <std::string> b;
 int n = File.ReadInt();
 for (int i = 0; i < n; i++)
 {
 a.insert(File.ReadInt());
 }
 int m = File.ReadInt();
 int cnt = 0, cnt1 = 0;
 for(int i = 0; Ans.HasInput(); i++) {
 for (int j = 0; j < m; j++) {
 int x = Ans.ReadInt();
 }
 cnt++;
 }
 for(int i = 0; Out.HasInput(); i++) {
 std::map<int, int> cntmap;
 std::string check = "";
 for (int j = 0; j < m; j++) {
 int x = Out.ReadInt();
 cntmap[x]++;
 if(cntmap[x] > a.count(x)) {
 std::stringstream msg;
 msg << "Map: " << cntmap[x] << "; Count: "<< a.count(x) << "; Number: " << x;
 QuitWith(WA, msg.str());
 }
 check+=inttostring(x) + '|';
 }
 cnt1++;
 }
 if (cnt1 < cnt)
 QuitWith(WA, "Users count less than correct");
 if (cnt1 == cnt)
 QuitWith(AC,"OK");
 if (cnt1 > cnt)
 QuitWith(WA, "Users count greater than correct");

 }
```

79

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector <int> a;
 for (int i = 0; i < n; i++)
 {
 a.push_back(File.ReadInt());
 }
 int m = File.ReadInt();
 for (int i = 0; i < m; i++) {
 int index = Out.ReadInt(), target = File.ReadInt(), answ = Ans.ReadInt();
 if (answ == index)
 continue;
 if (index < 0 || index > n) {
 QuitWith(WA, "Incorrect user output");
 }
 if (a[index-1] != target) {
 QuitWith(WA, "Incorrect user output");
 }
 }
 QuitWith(AC,"OK");
 }
```

113

```
#include <checkers/testlib.h>
 #include <cmath>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector<std::vector<int>> t(3);
 for (int i = 1; i <= n; i++)
 t[0].push_back(n-i+1);
 for (int i = 0; i < pow(2,n) - 1; i++)
 {
 int a = Out.ReadInt(1, n), b = Out.ReadInt(1,3), c = Out.ReadInt(1, 3);
 if (t[b-1].back() != a || !t[c-1].empty())
 if (t[c-1].back() < a)
 QuitWith(WA, "Wrong");
 t[c-1].push_back(a);
 t[b-1].pop_back();
 }
 if (t[0].size() != 0 || t[1].size() != 0 || t[2].size() != n)
 QuitWith(WA, "Wrong");
 QuitWith(AC,"OK");
 }
```

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector <int> a;
 for (int i = 0; i < n; i++)
 {
 a.push_back(File.ReadInt());
 }
 int m = File.ReadInt();
 for (int i = 0; i < m; i++) {
 int index = Out.ReadInt(), target = File.ReadInt(), answ = Ans.ReadInt();
 if (answ == index)
 continue;
 if (index < 0 || index > n) {
 QuitWith(WA, "Incorrect user output");
 }
 if (a[index-1] != target) {
 QuitWith(WA, "Incorrect user output");
 }
 }
 QuitWith(AC,"OK");
 }
```

125

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <sstream>
using namespace NTestlib;
int main(int argc, char* argv[])
{
InitInteractor(argc, argv);
int n = File.ReadInt();
std::cout << n << std::endl;
std::vector <int> a;
for (int i = 0; i < n; i++)
a.push_back(File.ReadInt());
int m = File.ReadInt();
std::cout << m << std::endl;
for (int i = 0; i < m; i++)
{
int cnt = 0;
int val = File.ReadInt();
std::cout << val << std::endl;
int answer;
answer = Ans.ReadInt();
while(true) {
if (cnt >= 50) {
QuitWith(WA, "Too many requests");
}
char requestType = Out.ReadChar();
bool isFind = false;
switch (requestType) {
case '!':
{
int userAnswer = Out.ReadInt();
if (userAnswer != answer) {
std::stringstream msg;
msg << "Incorrect user output. User answer: " << userAnswer << ". Correct answer: " << answer;
QuitWith(WA, msg.str());
}
isFind = true;
break;
}
case '?':
{
int userAnswer = Out.ReadInt(1, n);
if (userAnswer > 0 && userAnswer <= n)
std::cout << a[userAnswer - 1] << std::endl;
else
{
std::stringstream msg;
msg << "Wrong index " << userAnswer;
QuitWith(PE, msg.str());
}
break;
}
default:
{
std::stringstream msg;
msg << "Incorrect input command: " << requestType;
QuitWith(PE, msg.str());
}
break;
}
Out.Match('\n');
if (isFind) {
break;
}
cnt++;
}
}
QuitWith(AC, "OK");
std::cerr << "INT\n";
return 0;
}
```

131

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector<std::string> a;
 for (int i = 0; i < n; i++)
 {
 a.push_back(File.ReadWord());
 }
 int m = File.ReadInt();
 for (int i = 0; i < m; i++) {
 std::string target = File.ReadWord();
 int answ = Out.ReadInt();
 if ((answ < 1 || answ > n) && (answ != -1))
 QuitWith(PE, "Presentation error. Incorrect index");
 if ((answ == -1) && (std::find(a.begin(), a.end(), target) != a.end()))
 QuitWith(PE, "Presentation error. Can't find word with this index");
 if (answ == -1)
 continue;
 if (a[answ-1] != target)
 QuitWith(WA, "Incorrect user output");
 }
 QuitWith(AC,"OK");
 }
```

140

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <string>
 #include <vector>
 #include <algorithm>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 std::string out = "", answer = "";
 //File - programm input (from .in file)
 //Out - User output
 //Ans - Correct output (from .out file)
 out = Out.ReadLine();
 answer = Ans.ReadLine();
 while(out[out.size() -1] == 32)
 out = out.substr(0, out.size() - 1);
 while(answer[answer.size() -1] == 32)
 answer = answer.substr(0, answer.size() - 1);
 if (out==answer)
 QuitWith(AC, "Full solution");
 std::stringstream msg;
 msg << "Out " << out.c_str() << ", Correct answer: " << answer.c_str();
 QuitWith(WA, msg.str());
 }
```

153

```
#include <checkers/testlib.h>
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>

using namespace NTestlib;

int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n = File.ReadInt(), w = File.ReadInt();
std::vector <int> a;
std::vector <int> b;
std::vector <bool> c(n, false);
for (int i = 0; i < n; i++)
{
a.push_back(File.ReadInt());
}
for (int i = 0; i < n; i++)
{
b.push_back(File.ReadInt());
}
int rescnt = 0;
while (Ans.HasInput()) {
int x = Ans.ReadInt(1,n);
rescnt+= a[x-1];
}
int oufcnt = 0, oufw = 0;
while (Out.HasInput()) {
int x = Out.ReadInt(1,n);
if (c[x - 1]) {
QuitWith(WA, "Same ingot");
}
c[x - 1] = true;
oufcnt += a[x - 1];
oufw += b[x - 1];
if (oufw > w) {
QuitWith(WA, "Rucksack overflow");
}
}
if (oufcnt < rescnt) {
QuitWith(WA, "Wrong answer");
}
if (oufcnt == rescnt) {
QuitWith(AC, "Full solution");
}
if (oufcnt > rescnt) {
QuitWith(EF, "Contestant output better than jury's");
}
}
```

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <string>
 #include <vector>
 #include <algorithm>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 int g[100][100];
 int in[100][100];
 for (int i =0; i < n; i++)
 for (int j =0; j < n; j++)
 in[i][j] = File.ReadInt();
 while (Out.HasInput())
 {
 int a = Out.ReadInt(1, n);
 int b = Out.ReadInt(1, n);
 a--;
 b--;
 g[a][b] = 1;
 g[b][a] = 1;
 }
 for (int i = 0; i < n; i++)
 for (int j =0; j < n; j++)
 if (g[i][j] != in[i][j])
 {
 std::stringstream msg;
 msg << "WA wrong cell " << i << " " << j;
 QuitWith(WA, msg.str());
 }
 QuitWith(AC, "OK");
 }
```
172

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = 0, n2 = 0, x;
  std::set <int> s1, s2;
 while(Ans.HasInput()){
 x = Ans.ReadInt();
 n1++;
 s1.insert(x);
 }
 while(Out.HasInput()) {
 x = Out.ReadInt();
 n2++;
 s2.insert(x);
 }
 if (n1 != n2) {
 QuitWith(WA, "WA");
 }
 for (std::set<int>::iterator i = s1.begin(); i != s1.end(); i++) {
 if (s2.find(*i) == s2.end())
 QuitWith(WA, "WA");
 }
 QuitWith(AC, "Full solution");
 }
```
178

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = Ans.ReadInt(), n2 = Out.ReadInt();
 if (n1 != n2)
 QuitWith(WA, "Wrong answer. Incorrect first value");
 for (int i = 0; i < n1; i++) {
 int nn1 = Ans.ReadInt(), nn2 = Out.ReadInt();
 if (nn1 != nn2) {
 QuitWith(WA, "Wrong answer. Incorrect value");
 }
 std::set<int> s;
 for (int j = 0; j < nn1; j++) {
 s.insert(Ans.ReadInt());
 }
 for (int j = 0; j < nn1; j++) {
 if (s.find(Out.ReadInt()) == s.end())
 QuitWith(WA, "Wrong answer. Incorrect value");
 }
 }
 QuitWith(AC, "Full solution");
 }
```

180

182

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = Out.ReadInt();
 int n2 = Ans.ReadInt();
 if (n1 > n2)
 {
 QuitWith(WA, "WA");
 }
 if (n1 == -1)
 if (n2 == -1)
 QuitWith(AC, "OK");
 else
 QuitWith(WA, "WA");
 int n = File.ReadInt();
 int m = File.ReadInt();
 int x1 = File.ReadInt();
 int y1 = File.ReadInt();
 int x2 = File.ReadInt();
 int y2 = File.ReadInt();
 std::vector <int> x, y;
 for (int i = 0; i < n1 + 1; i++)
 {
 int x3 = Out.ReadInt();
 int y3 = Out.ReadInt();
 x.push_back(x3);
 y.push_back(y3);
 }
 if (x[0] != x1 || y[0] != y1 || x[x.size() - 1] != x2 || y[y.size() - 1] != y2)
 QuitWith(WA, "WA1");

 for (int i = 0; i < x.size() - 1; i++)
 {
 int dx = x[i + 1] - x[i], dy = y[i + 1] - y[i];
 if ((!(dx == 2 && dy == 1) && !(dx == 2 && dy == -1) && !(dx == 1 && dy == 2) && !(dx == 1 && dy == -2) &&
 !(dx == -1 && dy == 2) && !(dx == -1 && dy == -2) && !(dx == -2 && dy == -1) && !(dx == -2 && dy == 1)) ||
 x[i] > n || x[i] == 0 || x[i + 1] > n || x[i + 1] == 0 || y[i] > m || y[i] == 0 || y[i + 1] > m || y[i + 1] == 0)
 QuitWith(WA, "WA2");
 }
 QuitWith(AC, "OK");
 }
```

183

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = Out.ReadInt();
 int n2 = Ans.ReadInt();
 if (n1 == 0 && n2 == 0)
 QuitWith(AC, "OK");
 if (n1 > n2)
 QuitWith(WA, "WA");
 if (n1 == -1 && n2 == -1)
 QuitWith(AC, "OK");
 int n = 8;
 int m = 8;
 int x = File.ReadInt();
 int y = File.ReadInt();
 int x5 = File.ReadInt();
 int y5 = File.ReadInt();
 std::vector <int> x1, y1, x2, y2;
 x1.push_back(x);
 y1.push_back(y);
 x2.push_back(x5);
 y2.push_back(y5);
 for (int i = 0; i < n1; i++)
 {
 int x3 = Out.ReadInt();
 int y3 = Out.ReadInt();
 int x4 = Out.ReadInt();
 int y4 = Out.ReadInt();
 x1.push_back(x3);
 y1.push_back(y3);
 x2.push_back(x4);
 y2.push_back(y4);
 }
 if (x1.back() != x2.back() || y1.back() != y2.back())
 {
 QuitWith(WA, "WA1");
 }
 for (int i = 0; i < x1.size() - 1; i++)
 {
 int dx1 = x1[i + 1] - x1[i];
 int dy1 = y1[i + 1] - y1[i];
 int dx2 = x2[i + 1] - x2[i];
 int dy2 = y2[i + 1] - y2[i];
 int dx[8] = { 1,1,-1,-1,2,2,-2,-2 };
 int dy[8] = { 2,-2,2,-2,1,-1,1,-1 };
 bool f1 = 0;
 bool f2 = 0;
 for (int j = 0; j < 8; j++)
 {
 if (dx1 == dx[j] && dy1 == dy[j])
 f1 = 1;
 if (dx2 == dx[j] && dy2 == dy[j])
 f2 = 1;
 }
 if (!f1 || !f2)
 QuitWith(WA, "WA2");
 }
 QuitWith(AC, "OK");
 }
```

```
#include <checkers/testlib.h>
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
#include <map>
#include <string>
#include <sstream>

using namespace NTestlib;
struct node
{
int x1, y1, x2, y2;
bool l;
};
int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n1 = Out.ReadInt();
int n2 = Ans.ReadInt();
if (n1 > n2)
QuitWith(WA, "WA");
if (n1 == -1 && n2 == -1)
QuitWith(AC, "OK");
int n = 8;
int m = 8;
int x1 = File.ReadInt();
int y1 = File.ReadInt();
int x2 = File.ReadInt();
int y2 = File.ReadInt();
node now = { x1,y1,x2,y2,1 };
bool f = 0;
for (int i = 0; i < n1; i++)
{
int l3 = Out.ReadInt();
int x3 = Out.ReadInt();
int y3 = Out.ReadInt();
l3--;
if (l3 == 1)
{
if (now.l == l3)
QuitWith(WA, "WA1");
int dx[8] = { 2,2,-2,-2,1,1,-1,-1 };
int dy[8] = { -1,1,-1,1,-2,2,-2,2 };
bool ff = 0;
for (int i = 0; i < 8; i++)
if (now.x2 - x3 == dx[i] && now.y2 - y3 == dy[i])
ff = 1;
if (ff = 0)
QuitWith(WA, "WA2");
}
else
{
if (now.l == l3)
QuitWith(WA, "WA3");
int dx[8] = { 2,2,-2,-2,1,1,-1,-1 };
int dy[8] = { -1,1,-1,1,-2,2,-2,2 };
bool ff = 0;
for (int i = 0; i < 8; i++)
if (now.x1 - x3 == dx[i] && now.y1 - y3 == dy[i])
ff = 1;
if (ff = 0)
QuitWith(WA, "WA4");
}

if (l3 == 1)
{
now.x2 = x3;
now.y2 = y3;
}
else
{
now.x1 = x3;
now.y1 = y3;
}
now.l = !now.l;
}
if (f == 1)
QuitWith(WA, "WA5");
if (now.x1 != x2 || now.y1 != y2 || now.x2 != x1 || now.y2 != y1)
QuitWith(WA, "WA6");
QuitWith(AC, "OK");
}
```

minimum_spanning_tree

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 void dfs(std::vector<std::vector<int>> &a, std::vector<int> &u, int s, int &n)
 {
 for (int i = 0; i < n; i++)
 if (a[s][i] && !u[i])
 {
 u[i]++;
 dfs(a, u, i, n);
 }
 }

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = Ans.ReadInt(), n2 = Out.ReadInt();
 if (n1 > n2) {
 QuitWith(WA, "Wrong answer. User output greater than correct");
 }
 int n = File.ReadInt();
 std::vector<std::vector<int>> a(n, std::vector<int>(n));
 int s = 0;
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 {
 a[i][j] = Out.ReadInt();
 s += a[i][j];
 }
 if (n2 != s/2)
 QuitWith(WA, "Wrong answer");
 std::vector<int> u(n, 0);
 u[0]++;
 dfs(a, u, 0, n);
 int count = 0;
 for (int i = 0; i < n; i++)
 if (u[i])
 count++;
 if (count != n)
 QuitWith(WA, "Wrong answer");
 QuitWith(AC, "Full solution");
 }
```

190

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 std::vector <std::vector <int> > g(n, std::vector <int>(n, 0));
 for (int i = 0; i < n; i++)
 for (int j =0; j < n; j++)
 g[i][j] = File.ReadInt();

 std::string res = Ans.ReadWord();
 std::string ans = Out.ReadWord();
 if (res != ans)
 QuitWith(WA, "NOPE");
 if (res == "NO")
 QuitWith(AC, "OK");
 int cnt1 = Out.ReadInt(2,2*n);
 int x = Out.ReadInt(1,n);
 int cnt = 0;
 for (int i = 0; i < cnt1-1; i++)
 {
 int y = Out.ReadInt(1,n);
 if (g[x-1][y-1] == 100000)
 {
 std::stringstream msg;
 msg << "WA two unconnected vertices " << x << " " << y;
 QuitWith(WA, msg.str());
 }
 cnt+=g[x-1][y-1];
 x = y;
 }
 if (cnt >= 0)
 QuitWith(WA, "ABOVE ZERO");
 QuitWith(AC, "OK");
 }
```

195

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;
 int g[100][100];

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 int m = 0;
 int a[100][100];
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 {
 a[i][j] = File.ReadInt();
 if (a[i][j])
 m++;
 }
 m /= 2;
 int p = Out.ReadInt();
 int anss = Ans.ReadInt();
 if ((anss == -1 && p != -1 ))
 QuitWith(WA, "-1");
 if (anss == -1 && p == -1)
 QuitWith(AC, "OK");
 p--;
 int f = p;
 for (int i = 0; i < m; i++)
 {
 int r = Out.ReadInt();
 r--;
 if (a[p][r] == 0)
 QuitWith(WA, "WA1");
 if(g[r][p])
 QuitWith(WA, "WA3");
 g[r][p] = 1;
 g[p][r] = 1;
 p = r;
 }
 if (f != p)
 QuitWith(WA, "WA2");
 QuitWith(AC, "OK");
 }
```

196

```
#include <checkers/testlib.h>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace NTestlib;

int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n = File.ReadInt();
int start = File.ReadInt();
std::vector <std::vector <int> > g(n, std::vector <int>(n, 0));
for (int i = 0; i < n; i++)
for (int j =0; j < n; j++)
g[i][j] = File.ReadInt();
int x = Out.ReadInt(start,start);
int y;
std::vector <int> visited(n, 0);
for (int i = 0; i < n; i++)
{
y = Out.ReadInt(1, n);
if (!g[x-1][y-1])
{
std::stringstream msg;
msg << "WA two unconnected vertices " << x << " " << y;
QuitWith(WA, msg.str());
}
visited[y-1]++;
x = y;
}
if (y != start)
QuitWith(WA, "WA last != start");
for (int i = 0; i < n; i++)
if (visited[i] != 1)
QuitWith(WA, "vertice visited not 1 time");
QuitWith(AC, "OK");
}
```

198

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <string>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <string>
 #include <sstream>

 using namespace NTestlib;
 int w[1000][1000], m[1000][1000];

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 int so = Out.ReadInt();
 int sa = Ans.ReadInt();
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 {
 w[i][j] = File.ReadInt();
 }
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 {
 m[i][j] = File.ReadInt();
 }
 int s = 0;
 std::vector <int> uw(n,0);
 for (int i = 0; i < n; i++)
 {
 int k = Out.ReadInt(1, n);
 k--;
 if (uw[k])
 QuitWith(WA, "WA2");
 uw[k]++;
 s += w[i][k];
 s += m[k][i];
 }
 if (s != so)
 QuitWith(WA, "WA3");
 if (so < sa)
 QuitWith(WA, "WA1");
 QuitWith(AC, "OK");
 }
```

207

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>
 #include <regex>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int a[100][100];
 int cnt = 0;
 std::cout << 0;
 while (Ans.HasInput())
 {
 int x = Ans.ReadInt();
 int y = Ans.ReadInt();
 if (x != y)
 {
 a[x][y] = 1;
 a[y][x] = 1;
 cnt++;
 }
 }
 long double cnt1 = 0;
 while (Out.HasInput())
 {
 int x = Out.ReadInt(1,100);
 int y = Out.ReadInt(1,100);
 x--;
 y--;
 if (a[x][y] == 1)
 cnt1+=1;
 else
 cnt1-=0.5;
 }
 if ((cnt1 * 1L) / (cnt * 1L) > 0.5)
 {
 std::stringstream msg;
 msg << "OK " << cnt << " " << cnt1;
 QuitWith(AC, msg.str());
 }
 else
 {
 std::stringstream msg;
 msg << "WA " << int((cnt1 * 100) / cnt);
 QuitWith(WA, msg.str());
 }
 }
```

217

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>
 #include <regex>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 std::vector <bool> a;
 std::vector <std::string> b;
 while (File.HasInput())
 b.push_back(File.ReadLine());
 int cnt1 = 0;
 while (Ans.HasInput())
 {
 int x = Ans.ReadInt();
 if (x)
 cnt1++;
 a.push_back((x));
 }
 int cnt = 0;
 while (Out.HasInput())
 {
 int x = Out.ReadInt(1, a.size());
 int y = Out.ReadInt(1, b[x-1].length() - 13);
 if (!a[x - 1])
 QuitWith(WA, "Merlin was silent");
 if (b[x - 1].substr(y - 1, 14) != "Avada-ke-davra")
 QuitWith(WA, "Merlin didn't say that");
 cnt++;
 }
 if (cnt == cnt1)
 QuitWith(AC, "OK");
 else
 QuitWith(WA, "Merlin said more");
 }
```

```
#include <checkers/testlib.h>
#include <iostream>
#include <cmath>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <iomanip>
#include <sstream>

using namespace NTestlib;
double func1(double a, double b, double x)
{
return pow(x+a, 2) + b;
}

double func2(double a, double b, double c, double x)
{
return std::sin(x) + std::abs(x + a) - std::abs(x + b) + std::abs(x + c);
}

int main(int argc, char* argv[]) {
InitInteractor(argc, argv);
int test = File.ReadInt();
int cnt = 0;
if (test > 10)
{
double a = File.ReadDouble();
double b = File.ReadDouble();
double c = File.ReadDouble();
double an;
if (std::abs(a - b) > std::abs(b - c))
an = -a;
else
an = -c;
while (1)
{
char requestType = Out.ReadChar();
if (cnt > 1000000000)
QuitWith(WA, "WA2");
switch (requestType)
{
case '!':
{
double x = Out.ReadDouble();
if (std::abs(x - an) > 0.25 && (std::abs(func2(a,b,c,x) - func2(a,b,c,an)) > 1.9))
{
std::stringstream msg;
msg << "WA x:" << x << ", ans: " << an << ", abs1: " << std::abs(x - an) << ", abs2: " << func2(a, b, c, an) - func2(a, b, c, x);
QuitWith(WA, msg.str());
}
else
QuitWith(AC, "OK");
break;
}
case '?':
{
double x = Out.ReadDouble();
std::cout << std::setprecision(20) << func2(a, b, c, x) << std::endl;
break;
}
default:
{
std::stringstream msg;
msg << "Wrong query: " << requestType;
QuitWith(PE, msg.str());
break;
}
}
Out.Match('\n');
cnt++;
}
}
else
{
double a = File.ReadDouble();
double b = File.ReadDouble();
double an = -a;
while (1)
{
char requestType = Out.ReadChar();
if (cnt > 1000000000)
QuitWith(WA, "Too many requests");
switch (requestType)
{
```

226

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <iomanip>
#include <sstream>
#include <cmath>
using namespace NTestlib;

double func(double x)
{
return 2 * std::cos(0.5 * x);
}

int main(int argc, char* argv[])
{
InitInteractor(argc, argv);
int n = File.ReadInt();
double a = File.ReadDouble();
double b = File.ReadDouble();
std::vector<double> ans;
for (int i = 0; i < n; i++)
ans.push_back(File.ReadDouble());
std::cout << std::setprecision(20) << n << " " << a << " " << b << std::endl;
int cnt = 0;
while(true) {
cnt++;
char requestType = Out.ReadChar();
switch (requestType) {
case '!':
{
for (int i = 0; i < n; i++) {
double r = Out.ReadDouble();
if (std::abs(r - ans[i]) > 0.25) {
std::stringstream msg;
msg << "Wrong answer. Abs: " << r << " " << ans[i] <<" number: " << i;
QuitWith(WA, msg.str());
}
QuitWith(AC, "OK");
}
}
case '?':
{
double x = Out.ReadDouble();
std::cout << std::setprecision(20) << func(x) << std::endl;
break;
}
default:
{
std::stringstream msg;
msg << "Incorrect input command: " << requestType;
QuitWith(PE, msg.str());
}
break;
}
Out.Match('\n');
}
std::cerr << "INT\n";
return 0;
}
```

227

```
#include <checkers/testlib.h>
#include <cmath>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <algorithm>
#include <vector>
#include <iomanip>
#include <sstream>

using namespace NTestlib;

double func(int a, int b, double x, double y)
{
return pow(x + a, 2) + pow(y + b, 2);
}

double dx(int a, double x)
{
return 2 * (x + a);
}

double dy(int b, double y)
{
return 2 * (b + y);
}

double dist(double anX, double anY, double x, double y)
{
return std::sqrt(pow(anX - x, 2) + pow(anY - y, 2));
}

int main(int argc, char* argv[]) {
InitInteractor(argc, argv);
int a = File.ReadInt();
int b = File.ReadInt();
double anX = -a;
double anY = -b;
int cnt = 0;
double x, y;

while(1)
{
char requestType = Out.ReadChar();
if (cnt > 1000000000)
QuitWith(WA, "Too many requests");
switch(requestType)
{
case '!':
{
x = Out.ReadDouble();
y = Out.ReadDouble();
if (dist(anX, anY, x, y) > 0.1)
{
std::stringstream msg;
msg << "Incorrect user output. User answer: " << x << ", " << y << ". abs2: " << dist(anX, anY, x, y);
QuitWith(WA, msg.str());
}
else
QuitWith(AC, "OK");
break;
}
case '?':
{
x = Out.ReadDouble();
y = Out.ReadDouble();
std::cout << std::setprecision(20) << func(a, b, x, y) << " " << dx(a, x) << " " << dy(b, y) << std::endl;
break;
}
default:
{
std::stringstream msg;
msg << "Wrong query: " << requestType;
QuitWith(PE, msg.str());
break;
}
}
Out.Match('\n');
cnt++;
}
std::cerr << "INT\n";
return 0;
}
```

228

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <fstream>
 #include <stdlib.h>
 #include <vector>
 #include <iomanip>
 #include <sstream>

 using namespace NTestlib;

 #define ld long double
 #define eps 0.001
 ld f1(ld x, ld y)
 {
 return x * x + y * y - 0.1*std::abs(1 - x) - 0.1*std::abs(1 - y);
 }
 ld f2(ld x, ld y)
 {
 return 20 * std::abs(x - 50)*std::abs(y - 25) + 10 * (std::abs(x - 10)*std::abs(y - 10) + std::abs(x - 50));
 }


 int main(int argc, char* argv[])
 {
 InitInteractor(argc, argv);
 int f = File.ReadInt();
 int k = 0;
 while(Out.HasInput()) {
 k++;
 if (k > 100000)
 QuitWith(WA, "Too many requests");
 char requestType = Out.ReadChar();
 ld x = Out.ReadDouble();
 ld y = Out.ReadDouble();
 switch (requestType) {
 case '!':
 {
 if(f)
 if (std::abs(x+0.05) < eps && abs(y+0.05) < eps)
 QuitWith(AC, "Full solution");
 else {
 std::stringstream msg;
 msg << "Incorrect output: " << x << " " << y;
 QuitWith(WA, msg.str());
 }
 else
 if (std::abs(50-x) < eps && std::abs(10 - y) < eps)
 QuitWith(AC, "Full solution");
 else {
 std::stringstream msg;
 msg << "Incorrect output: " << x << " " << y;
 QuitWith(WA, msg.str());
 }
 }
 case '?':
 {
 if (f)
 std::cout << f1(x,y) << std::endl;
 else
 std::cout << f2(x,y) << std::endl;
 break;
 }
 default:
 {
 std::stringstream msg;
 msg << "Incorrect input command: " << requestType;
 QuitWith(PE, msg.str());
 }
 break;
 }
 Out.Match('\n');
 }
 QuitWith(EF, "Incorrect: Input after EOF");
 std::cerr << "INT\n";
 return 0;
 }
```

salesman

```
#include <checkers/testlib.h>
#include <cmath>
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
#include <map>
#include <set>
#include <string>
#include <sstream>

using namespace NTestlib;

double dist(std::pair<int, int> a, std::pair<int, int> b)
{
return std::sqrt(pow(a.first - b.first, 2) + pow(a.second - b.second, 2));
}

int main(int argc, char* argv[]) {
InitChecker(argc, argv);
int n = File.ReadInt();
double answer = Ans.ReadDouble();
double userAnswer = Out.ReadDouble();
std::vector<std::pair<int, int>> vec(n);
for (int i = 0; i < n; i++)
{
vec[i].first = File.ReadInt();
vec[i].second = File.ReadInt();
}
std::vector<int> ind(n);
std::vector<int> u(n, 0);
for (int i = 0; i < n; i++) {
ind[i] = Out.ReadInt();
if (u[ind[i] - 1]) {
QuitWith(PE, "Value already exist");
}
u[ind[i] - 1]++;
}
double res = 0;
for (int i = 0; i < n; i++) {
res += dist(vec[ind[i] - 1], vec[ind[(i + 1) % n] - 1]);
}
if (std::abs(res - userAnswer) > 0.5) {
std::stringstream msg;
msg << "Incorrect output. User answer: " << userAnswer << ". Correct output: " << res;
QuitWith(WA, msg.str());
}
if (userAnswer - answer > answer * 0.1)
{
std::stringstream msg;
msg << "Wrong answer. Absolute diff: " << userAnswer;
QuitWith(WA, msg.str());
}
QuitWith(AC, "Full solution");
}
```

aStar

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;
 int a[10000][10000];

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n = File.ReadInt();
 int m = File.ReadInt();
 int px1 = File.ReadInt();
 int py1 = File.ReadInt();
 int px2 = File.ReadInt();
 int py2 = File.ReadInt();

 for (int i = 0; i < n; i++)
 for (int j = 0; j < m; j++)
 a[i][j] = File.ReadInt();

 int dx, dy;
 int ln = Out.ReadInt();
 dx = Out.ReadInt();
 dy = Out.ReadInt();

 if(dx != px1 || dy != py1)
 QuitWith(WA, "Wrong start position");

 for(int i = 0; i < ln-1; i++)
 {
 dx = Out.ReadInt();
 dy = Out.ReadInt();
 if ((dx <= m && dx >= 0 && dy <= n && dy >= 0) && a[dy - 1][dx - 1] == 0 && ((dy == py1 + 1 && dx == px1) || (dy == py1 - 1 && dx == px1) || (dy == py1 && dx == px1 + 1) || (dy == py1 && dx == px1 - 1)))
 {
 px1 = dx;
 py1 = dy;
 }
 else{
 std::stringstream msg;
 msg << "Impossible movement " << px1 << ", " << py1 << ", " << dx << ", " << dy;
 QuitWith(WA, msg.str());
 }
 }

 if(px1 != px2 || py1 != py2)
 QuitWith(WA, "Wrong end position");

 int opCount = Ans.ReadInt();
 int eps = 3;

 if(ln - opCount > eps)
 QuitWith(WA, "Too long way");

 QuitWith(AC, "Full solution");
 }
```

234

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>
 #include <set>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);

 double cnt = 0;
 int x;
 for(int i = 0; i < 3; i++)
 x = File.ReadInt();
 for(int i = 0; i < x; i++)
 {
 std::string x = Ans.ReadWord();
 std::string y = Out.ReadWord();
 if (x == y)
 cnt++;
 }
 if (cnt / x > 0.8){
 std::stringstream msg;
 msg << "OK " << cnt;
 QuitWith(AC, msg.str());
 }
 else{
 std::stringstream msg;
 msg << "WA " << cnt/50;
 QuitWith(WA, msg.str());
 }
 }
```

235

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;
 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);

 int counter = 0;

 for (int i = 0; i < 5; i++)
 {
 int a1 = Out.ReadInt();
 int b1 = Out.ReadInt();
 int a2 = Ans.ReadInt();
 int b2 = Ans.ReadInt();
 if (a1 == a2 && b1 == b2)
 counter++;
 }
 if (counter < 4) {
 std::stringstream msg;
 msg << "Wrong answer. Only " << counter << " of 5 was right";
 QuitWith(WA, msg.str());
 }
 QuitWith(AC, "Full solution");
 }
```

236

```
#include <checkers/testlib.h>
 #include <cmath>
 #include <iostream>
 #include <string>
 #include <vector>
 #include <algorithm>

 using namespace NTestlib;

 std::vector<std::pair<int, int> > dots;
 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 double cnt = 0;
 int n = File.ReadInt();
 int k = File.ReadInt();
 for(int i = 0; i < n; i++)
 {
 int x = File.ReadInt();
 int y = File.ReadInt();
 dots.push_back(std::make_pair(x, y));
 }
 double reference = Ans.ReadDouble();
 double result = Out.ReadDouble();
 if (result > reference * 1.3)
 QuitWith(WA, "Not correct clasterisation");

 double cluster_sum = 0;
 for(int i = 0; i < k; i++)
 {
 double x = Out.ReadDouble();
 double y = Out.ReadDouble();
 int num = Out.ReadInt();
 double tmp = 0;
 for (int j = 0; j < num; j++)
 {
 int cluster = Out.ReadInt() - 1;
 tmp += std::abs(std::sqrt(pow(x - dots[cluster].first, 2) + pow(y - dots[cluster].second, 2)));
 }
 cluster_sum += tmp;
 }
 if (std::abs(result - cluster_sum) < 10)
 QuitWith(AC, "OK");

 std::stringstream msg;
 msg << "WTF. Result:" << result << ", Clister sum:" << cluster_sum;
 QuitWith(WA, msg.str());
 }
```

```
#define _CRT_SECURE_NO_DEPRECATE
#define _USE_MATH_DEFINES
#include <iostream>
#include <checkers/testlib.h>
#include <time.h>
#include <vector>
#include <algorithm>
#include <cmath>
#include <string>
#include <set>
#include <vector>
#include <map>
#include <sstream>
#include <iomanip>
#include <stack>
#include <cstdio>
#include <fstream>
#include <cstdlib>
#include <numeric>
#include <cstring>
#include <complex>
#include <cassert>
#include <iterator>
#include <functional>
using namespace NTestlib;
struct node
{
int key;
unsigned char height;
node* left;
node* right;
node(int k) { key = k; left = right = 0; height = 1; }
};

unsigned char height(node* p)
{
return p ? p->height : 0;
}

int bfactor(node* p)
{
return height(p->right) - height(p->left);
}

void fixheight(node* p)
{
unsigned char hl = height(p->left);
unsigned char hr = height(p->right);
p->height = (hl>hr ? hl : hr) + 1;
}

node* rotateright(node* p)
{
node* q = p->left;
p->left = q->right;
q->right = p;
fixheight(p);
fixheight(q);
return q;
}

node* rotateleft(node* q)
{
node* p = q->right;
q->right = p->left;
p->left = q;
fixheight(q);
fixheight(p);
return p;
}

node* balance(node* p)
{
fixheight(p);
if (bfactor(p) == 2)
{
if (bfactor(p->right) < 0)
p->right = rotateright(p->right);
return rotateleft(p);
}
if (bfactor(p) == -2)
{
if (bfactor(p->left) > 0)
p->left = rotateleft(p->left);
```

```
#define _CRT_SECURE_NO_DEPRECATE
 #define _USE_MATH_DEFINES
 #include <checkers/testlib.h>
 #include <iostream>

 using namespace NTestlib;

 #define mp std::make_pair
 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int x1 = File.ReadInt();
 int y1 = File.ReadInt();
 int x2 = File.ReadInt();
 int y2 = File.ReadInt();
 std::pair<int, int> ans11 = mp( x1 + (y2 - y1), y1 - (x2 - x1));
 std::pair<int, int> ans21 = mp( x1 - (y2 - y1), y1 + (x2 - x1));
 std::pair<int, int> ans12 = mp( x2 + (y2 - y1), y2 - (x2 - x1));
 std::pair<int, int> ans22 = mp( x2 - (y2 - y1), y2 + (x2 - x1) );
 std::pair<int, int> ans1;
 ans1.first = Out.ReadInt();
 ans1.second = Out.ReadInt();
 std::pair<int, int> ans2;
 ans2.first = Out.ReadInt();
 ans2.second = Out.ReadInt();
 if (((ans1 == ans11 && ans2 == ans12) || (ans1 == ans12 && ans2 == ans11)) || ((ans1 == ans22 && ans2 == ans21) || (ans2 == ans21 && ans1 == ans22)))
 QuitWith(AC, "OK");

 std::stringstream msg;
 msg << "WA ans1: " << ans1.first << ", " << ans1.second << "; ans2: " << ans2.first << ", " << ans2.second << "; ans11: " << ans11.first << ", " << ans11.second <<"; ans12: " << ans12.first << ", " << ans12.second << "; ans21: " << ans21.first << ", " << ans21.second << "; ans22: " << ans22.first << ", " << ans22.second;
 QuitWith(WA, msg.str());
 }
```
247

```
#include <checkers/testlib.h>
 #define _CRT_SECURE_NO_DEPRECATE
 #define _USE_MATH_DEFINES
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #define eps 1e-6

 using namespace NTestlib;
 #define mp std::make_pair
 typedef long double ld;

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 ld x1 = File.ReadDouble();
 ld y1 = File.ReadDouble();
 ld x2 = File.ReadDouble();
 ld y2 = File.ReadDouble();
 std::pair <ld, ld> ans11 = mp((x1 + x2) / 2 + (y1 - y2) / 2, (y1 + y2) / 2 + (x2 - x1) / 2);
 std::pair<ld, ld> ans12 = mp((x1 + x2) / 2 + (y2 - y1) / 2, (y1 + y2) / 2 + (x1 - x2) / 2);
 std::pair<ld, ld> ans1;
 ans1.first = Out.ReadDouble();
 ans1.second = Out.ReadDouble();
 std::pair<ld, ld> ans2;
 ans2.first = Out.ReadDouble();
 ans2.second = Out.ReadDouble();
 if ((ans1 == ans11 && ans2 == ans12) || (ans1 == ans12 && ans2 == ans11))
 QuitWith(AC, "OK");
 QuitWith(WA, "WA");
 }
```
248

```
#define _CRT_SECURE_NO_DEPRECATE
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>

#include <vector>
#include <checkers/testlib.h>
#include <algorithm>


typedef long long int64;
typedef long double ld;

//const int maxn = 1e+6;
#define eps 1e-6
#define mp std::make_pair
#define pb push_back
using namespace NTestlib;

typedef std::pair<ld, ld> vec;
typedef std::vector<int64> vint;
typedef std::vector<vint> vvint;

vec operator-(const vec& lhs, const vec& rhs)
{
return mp( lhs.first - rhs.first,lhs.second - rhs.second );
}

ld dist(vec a, vec b)
{
return sqrtl((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second - b.second));
}
ld len(vec x)
{
return sqrtl(x.first*x.first + x.second*x.second);
}
ld angleaa(vec a, vec b)
{
return acosl((a.first * b.first + a.second*b.second) / (len(a)*len(b)));
}
ld ar_tr(vec a, vec b)
{
return a.first*b.second - b.first*a.second;
}





int main(int argc, char ** argv)
{

InitChecker(argc, argv);
ld lena = File.ReadDouble(), lenb = File.ReadDouble(), angle = File.ReadDouble();
vec a = mp(Out.ReadDouble(), Out.ReadDouble()), b = mp(Out.ReadDouble(), Out.ReadDouble());
vec c = mp(Out.ReadDouble(), Out.ReadDouble());
std::vector<vec> t;
t.pb(a); t.pb(b); t.pb(c);
for (int i = 0; i < 3; ++i)
{
vec first = (t[(i + 1) % 3] - t[i]);
vec second = (t[(i + 2) % 3] - t[i]);
if (((std::abs(len(first) - lena) < eps && std::abs(len(second) - lenb) < eps) || (std::abs(len(first) - lenb) < eps && std::abs(len(second) - lena) < eps)) && std::abs(angleaa(first, second) * 180 / M_PI - angle) < eps)
QuitWith(AC, "OK");
}
QuitWith(WA, "Wrong answer");
}
```
249

```
#define _CRT_SECURE_NO_DEPRECATE
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>
#include <checkers/testlib.h>
typedef long long int64;
typedef long double ld;

//const int maxn = 1e+6;
#define mp std::make_pair
#define eps 1e-6
#define pb push_back
using namespace NTestlib;

typedef std::pair<ld, ld> vec;
typedef std::vector <int64> vint;
typedef std::vector <vint> vvint;


vec operator-(const vec& lhs, const vec& rhs)
{
return mp( lhs.first - rhs.first,lhs.second - rhs.second );
}

ld dist(vec a, vec b)
{
return sqrtl((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second - b.second));
}
ld len(vec x)
{
return sqrtl(x.first*x.first + x.second*x.second);
}
ld angleaa(vec a, vec b)
{
return acosl((a.first * b.first + a.second*b.second) / (len(a)*len(b)));
}
ld ar_tr(vec a, vec b)
{
return a.first*b.second - b.first*a.second;
}


int main(int argc, char ** argv)
{
InitChecker(argc, argv);
ld lena = File.ReadDouble(), anglea = File.ReadDouble(), angleb = File.ReadDouble();
vec a = mp(Out.ReadDouble(), Out.ReadDouble()), b = mp(Out.ReadDouble(), Out.ReadDouble());
vec c = mp(Out.ReadDouble(), Out.ReadDouble());
std::vector<vec> t;
t.pb(a); t.pb(b); t.pb(c);
for (int i = 0; i < 3; ++i)
{
vec first = (t[(i + 1) % 3] - t[i]);
vec second = (t[(i + 2) % 3] - t[i]);
vec third = (t[(i + 1) % 3] - t[(i + 2) % 3]);
ld angle1 = angleaa(first, third) * 180 / M_PI;
ld angle2 = angleaa(second, first) * 180 / M_PI;
if (std::abs(len(first) - lena) < eps && ((std::abs(angle1 - anglea) && std::abs(angle2 - angleb) < eps) || (std::abs(angle1 - angleb) < eps && std::abs(angle2 - anglea) < eps)))
QuitWith(AC, "OK");
}
QuitWith(WA, "Wrong answer");
}
```

252

```
#define _CRT_SECURE_NO_DEPRECATE
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
#include <checkers/testlib.h>

#include <vector>
#include <algorithm>

typedef long long int64;
typedef long double ld;
const double eps = 1e-6;
#define mp std::make_pair
//const int maxn = 1e+6;


#define pb push_back
using namespace NTestlib;

typedef std::pair<ld, ld> vec;
typedef std::vector <int64> vint;
typedef std::vector <vint> vvint;
vint primes;
vint HH, pp;

vec operator-(const vec& lhs, const vec& rhs)
{
return mp( lhs.first - rhs.first,lhs.second - rhs.second );
}
vec operator+(const vec& lhs, const vec& rhs)
{
return mp(lhs.first + rhs.first,lhs.second + rhs.second );
}
vec operator+(const vec& lhs, const int& rhs)
{
return mp( lhs.first /rhs,lhs.second/rhs);
}

ld dist(vec a, vec b)
{
return sqrtl((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second - b.second));
}
ld len(vec x)
{
return sqrtl(x.first*x.first + x.second*x.second);
}
ld angle(vec a, vec b)
{
return acosl((a.first * b.first + a.second*b.second) / (len(a)*len(b)));
}
ld ar_tr(vec a, vec b)
{
return a.first*b.second - b.first*a.second;
}

int main(int argc, char ** argv)
{
InitChecker(argc, argv);
ld lena = File.ReadDouble(), lenb = File.ReadDouble(), lenc = File.ReadDouble();
vec a = mp( Out.ReadDouble(), Out.ReadDouble() ), b = mp( Out.ReadDouble(),Out.ReadDouble() ), c = mp( Out.ReadDouble(),Out.ReadDouble() );
std::vector <ld> t;
t.pb(len(a-b)); t.pb(len(b-c)); t.pb(len(c-a));
std::vector <ld> ans;
ans.pb(lena); ans.pb(lenb); ans.pb(lenc);
std::sort(ans.begin(), ans.end());
std::sort(t.begin(), t.end());
for (int i = 0; i < 3; ++i)
{
if (std::abs(ans[i] - t[i]) > eps)
QuitWith(WA, "Wrong answer");
}
QuitWith(AC, "OK");
}
```

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <fstream>
 #include <stdlib.h>
 #include <vector>
 #include <sstream>
 using namespace NTestlib;
 int main(int argc, char* argv[])
 {
 InitInteractor(argc, argv);
 int n = File.ReadInt();
 std::cout << n << std::endl;
 std::vector <int> a;
 for (int i = 0; i < n; i++)
 a.push_back(File.ReadInt());
 int cnt = 0;
 int answer;
 answer = Ans.ReadInt();
 while(true) {
 if (cnt >= 50) {
 QuitWith(WA, "Too many requests");
 }
 char requestType = Out.ReadChar();
 switch (requestType) {
 case '!':
 {
 int userAnswer = Out.ReadInt();
 if (userAnswer != answer) {
 std::stringstream msg;
 msg << "Incorrect user output. User answer: " << userAnswer << ". Correct answer: " << answer;
 QuitWith(WA, msg.str());
 }
 QuitWith(AC, "OK");
 break;
 }
 case '?':
 {
 int userAnswer = Out.ReadInt(1, n);
 if (userAnswer > 0 && userAnswer <= n)
 std::cout << a[userAnswer - 1] << std::endl;
 else
 {
 std::stringstream msg;
 msg << "Wrong index " << userAnswer;
 QuitWith(PE, msg.str());
 }
 break;
 }
 default:
 {
 std::stringstream msg;
 msg << "Incorrect input command: " << requestType;
 QuitWith(PE, msg.str());
 }
 break;
 }
 Out.Match('\n');
 cnt++;
 }
 std::cerr << "INT\n";
 return 0;
 }
```

128

130

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <sstream>
using namespace NTestlib;
int main(int argc, char* argv[])
{
InitInteractor(argc, argv);
int n = File.ReadInt();
std::cout << n << std::endl;
int cnt = 0;
int answer;
answer = Ans.ReadInt();
while(true) {
if (cnt >= 50) {
QuitWith(WA, "Too many requests");
}
char requestType = Out.ReadChar();
switch (requestType) {
case '!':
{
int userAnswer = Out.ReadInt();
if (userAnswer != answer) {
std::stringstream msg;
msg << "Incorrect user output. User answer: " << userAnswer << ". Correct answer: " << answer;
QuitWith(WA, msg.str());
}
QuitWith(AC, "OK");
break;
}
case '?':
{
int userAnswer = Out.ReadInt(1, n);
if (userAnswer > 0 && userAnswer <= n)
{
if (userAnswer > n - answer)
std::cout << 1 << std::endl;
else
std::cout << 0 << std::endl;
}
else
{
std::stringstream msg;
msg << "Wrong index " << userAnswer;
QuitWith(PE, msg.str());
}
break;
}
default:
{
std::stringstream msg;
msg << "Incorrect input command: " << requestType;
QuitWith(PE, msg.str());
}
break;
}
Out.Match('\n');
cnt++;
}
std::cerr << "INT\n";
return 0;
}
```

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <sstream>
using namespace NTestlib;
int main(int argc, char* argv[])
{
InitInteractor(argc, argv);
int n = File.ReadInt();
std::cout << n << std::endl;
int cnt = 0;
int answer;
answer = Ans.ReadInt();
std::vector<int> a;
for (int i = 0; i < n; i++)
{
a.push_back(File.ReadInt());
}
while(true) {
if (cnt >= 50) {
QuitWith(WA, "Too many requests");
}
char requestType = Out.ReadChar();
switch (requestType) {
case '!':
{
int userAnswer = Out.ReadInt(1, n);
int res = userAnswer - 1;
if (res == 0) {
if (a[0] >= a[1])
QuitWith(AC, "OK");
else
QuitWith(WA, "Element isn't a peak one");
}
else if (res == n - 1) {
if (a[n-1] >= a[n-2])
QuitWith(AC, "OK");
else
QuitWith(WA, "Element isn't a peak one");
} else if (a[res] >= a[res - 1] && a[res] >= a[res + 1])
{
QuitWith(AC, "OK");
}
else {
QuitWith(WA, "Element isn't a peak one");
}
}
case '?':
{
int userAnswer = Out.ReadInt(1, n);
if (userAnswer > 0 && userAnswer <= n)
{
std::cout << a[userAnswer - 1] << std::endl;
}
else
{
std::stringstream msg;
msg << "Wrong index " << userAnswer;
QuitWith(PE, msg.str());
}
break;
}
default:
{
std::stringstream msg;
msg << "Incorrect input command: " << requestType;
QuitWith(PE, msg.str());
}
break;
}
Out.Match('\n');
cnt++;
}
std::cerr << "INT\n";
return 0;
}
```

146

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <sstream>
using namespace NTestlib;
int main(int argc, char* argv[])
{
 InitInteractor(argc, argv);
 int n = File.ReadInt();
 std::cout << n << std::endl;
 std::vector <int> a;
 for (int i = 0; i < n; i++)
 a.push_back(File.ReadInt());
 int cnt = 0;
 int answer;
 answer = Ans.ReadInt();
 while(true) {
 if (cnt >= 50) {
 QuitWith(WA, "Too many requests");
 }
 char requestType = Out.ReadChar();
 switch (requestType) {
 case '!':
 {
 int userAnswer = Out.ReadInt();
 if (userAnswer != answer) {
 std::stringstream msg;
 msg << "Incorrect user output. User answer: " << userAnswer << ". Correct answer: " << answer;
 QuitWith(WA, msg.str());
 }
 QuitWith(AC, "OK");
 break;
 }
 case '?':
 {
 int userAnswer = Out.ReadInt();
 if (userAnswer > 0 && userAnswer <= n)
 std::cout << a[userAnswer - 1] << std::endl;
 else
 {
 std::stringstream msg;
 msg << "Wrong index " << userAnswer;
 QuitWith(PE, msg.str());
 }
 break;
 }
 default:
 {
 std::stringstream msg;
 msg << "Wrong query: " << requestType;
 QuitWith(PE, msg.str());
 }
 break;
 }
 Out.Match('\n');
 cnt++;
 }
 std::cerr << "INT\n";
 return 0;
}
```

224

```
#include <checkers/testlib.h>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <vector>
#include <iomanip>
#include <sstream>
using namespace NTestlib;

double func(double a, double b, double c, double d, double x)
{
return x*x*x*a + x*x*b + x*c + d;
}

double diff(double a, double b, double c, double d, double x)
{
return 3 * x*x*a + 2 * x*b + c;
}

int main(int argc, char* argv[])
{
InitInteractor(argc, argv);
double a = File.ReadDouble();
double b = File.ReadDouble();
double c = File.ReadDouble();
double d = File.ReadDouble();
double an = File.ReadDouble();
an *= -1;
int cnt = 0;
while(true) {
if (cnt > 50)
QuitWith(WA, "Too many requests");
char requestType = Out.ReadChar();
switch (requestType) {
case '!':
{
double x = Out.ReadDouble();
if (std::min(std::min(std::abs(x - an), std::abs(func(a,b,c,d,x) - func(a,b,c,d,an))), std::abs(diff(a,b,c,d,x) - diff(a,b,c,d,an))) > 1e-6) {
std::stringstream msg;
msg << "Wrong answer. x: " << x << " ans: " << an << " abs: " << std::abs(x - an);
QuitWith(WA, msg.str());
} else {
QuitWith(AC, "OK");
}
}
case '?':
{
double x = Out.ReadDouble();
std::cout << std::setprecision(20) << func(a, b, c, d, x) << " " << diff(a, b, c, d, x) << std::endl;
break;
}
default:
{
std::stringstream msg;
msg << "Incorrect input command: " << requestType;
QuitWith(PE, msg.str());
}
break;
}
cnt++;
Out.Match('\n');
}
std::cerr << "INT\n";
return 0;
}
```

126

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <algorithm>
 #include <string>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int a = File.ReadInt(), b = File.ReadInt(), c = File.ReadInt(), d = File.ReadInt();
 double x = Out.ReadDouble();
 if (std::abs(a*x*x*x + b*x*x + c*x + d) < 0.000001) {
 QuitWith(AC,"OK");
 }
 QuitWith(WA, "Incorrect user output");
 }
```

```
#include <checkers/testlib.h>
 #include <algorithm>
 #include <string>
 #include <iostream>
 #include <map>
 #include <sstream>
 #include <vector>
 using namespace NTestlib;

 std::vector<int> ReadInts(TInputStream& in) {
  std::stringstream ss(in.ReadLine());
  std::vector<int> res;
  int x;
  while (ss >> x) {
  res.push_back(x);
  }
  return res;
 }

 std::string Msg1(int line, int expected, int got) {
  std::stringstream out;
  out << "Wrong number of ints on line: " << line;
  out << ". Expected: " << expected << ", got: " << got;
  return out.str();
 }

 std::string Msg2(int key) {
  std::stringstream out;
  out << "Key: " << key << " is not a valid key";
  return out.str();
 }

 int main(int argc, char* argv[])
 {
  InitChecker(argc, argv);
  int n = File.ReadUInt();
  std::map<int, int> mp;
  for (int i = 0; i < n; ++i) {
  int type = File.ReadUInt();
  if (type == 1) {
  int key = File.ReadUInt();
  int val = File.ReadUInt();
  mp[key] += val;
  continue;
  }
  // read skip value
  File.ReadUInt();

  auto a = ReadInts(Ans);
  auto b = ReadInts(Out);

  // check output for duplicates
  sort(begin(a), end(a));
  sort(begin(b), end(b));
  if (unique(begin(a), end(a)) != end(a)) {
  QuitWith(EF, "Some key was printed more than once!");
  }
  if (unique(begin(b), end(b)) != end(b)) {
  QuitWith(WA, "Some key was printed more than once!");
  }

  // check size
  if (a.size() != b.size()) {
  QuitWith(WA, Msg1(i, a.size(), b.size()));
  }

  // check that all keys are the keys that was already given, not arbitary ones.
  int n = (int)a.size();
  for (int i = 0; i < n; ++i) {
  if (mp.find(a[i]) == end(mp)) {
  QuitWith(EF, Msg2(a[i]));
  }
  if (mp.find(b[i]) == end(mp)) {
  QuitWith(WA, Msg2(b[i]));
  }
  }

  // now sort and compare values. we can't compare just keys because there may be keys with same value
  // and in such cases it is allowed to output any value
  // in other words multiset of values corresponding to ans keys and solution keys show be equal
  auto f = [&] (int x) {
  return mp[x];
  };
  std::transform(begin(a), end(a), begin(a), f);
```

```
#include <checkers/testlib.h>
 #include <algorithm>
 #include <string>
 #include <iostream>
 #include <map>
 #include <sstream>
 #include <vector>
 using namespace NTestlib;

 std::vector<int> ReadInts(TInputStream& in) {
  std::stringstream ss(in.ReadLine());
  std::vector<int> res;
  int x;
  while (ss >> x) {
  res.push_back(x);
  }
  return res;
 }

 std::string Msg1(int line, int expected, int got) {
  std::stringstream out;
  out << "Wrong number of ints on line: " << line;
  out << ". Expected: " << expected << ", got: " << got;
  return out.str();
 }

 std::string Msg2(int key) {
  std::stringstream out;
  out << "Key: " << key << " is not a valid key";
  return out.str();
 }

 int main(int argc, char* argv[])
 {
  InitChecker(argc, argv);
  int n = File.ReadUInt();
  std::map<int, int> mp;
  for (int i = 0; i < n; ++i) {
  int key = File.ReadUInt();
  int val = File.ReadUInt();
  mp[key] += val;

  auto a = ReadInts(Ans);
  auto b = ReadInts(Out);

  // check output for duplicates
  sort(begin(a), end(a));
  sort(begin(b), end(b));
  if (unique(begin(a), end(a)) != end(a)) {
  QuitWith(EF, "Some key was printed more than once!");
  }
  if (unique(begin(b), end(b)) != end(b)) {
  QuitWith(WA, "Some key was printed more than once!");
  }

  // check size
  if (a.size() != b.size()) {
  QuitWith(WA, Msg1(i, a.size(), b.size()));
  }

  // check that all keys are the keys that was already given, not arbitary ones.
  int n = (int)a.size();
  for (int i = 0; i < n; ++i) {
  if (mp.find(a[i]) == end(mp)) {
  QuitWith(EF, Msg2(a[i]));
  }
  if (mp.find(b[i]) == end(mp)) {
  QuitWith(WA, Msg2(b[i]));
  }
  }

  // now sort and compare values. we can't compare just keys because there may be keys with same value
  // and in such cases it is allowed to output any value
  // in other words multiset of values corresponding to ans keys and solution keys show be equal
  auto f = [&] (int x) {
  return mp[x];
  };
  std::transform(begin(a), end(a), begin(a), f);
  std::transform(begin(b), end(b), begin(b), f);
  std::sort(begin(a), end(a));
  std::sort(begin(b), end(b));
  if (a > b) {
  QuitWith(WA, "Solution not optimal");
  }
```

```
#define _CRT_SECURE_NO_DEPRECATE
 #define _USE_MATH_DEFINES
 #include <iostream>
 #include <cmath>
 #include <vector>
 #include <algorithm>
 #include <checkers/testlib.h>

 typedef long long int64;
 typedef long double ld;

 const double eps = 5 * 1e-3;

 //const int maxn = 1e+6;

 #define mp std::make_pair
 #define pb push_back
 using namespace NTestlib;

 typedef std::pair<ld, ld> vec;
 typedef std::vector <int64> vint;
 typedef std::vector <vint> vvint;

 vec operator-(const vec& lhs, const vec& rhs)
 {
 return mp( lhs.first - rhs.first,lhs.second - rhs.second );
 }
 vec operator+(const vec& lhs, const vec& rhs)
 {
 return mp( lhs.first + rhs.first,lhs.second + rhs.second );
 }
 vec operator/(const vec& lhs, const int& rhs)
 {
 return mp( lhs.first /rhs,lhs.second/rhs);
 }

 ld dist(vec a, vec b)
 {
 return sqrtl((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second - b.second));
 }
 ld len(vec x)
 {
 return sqrtl(x.first*x.first + x.second*x.second);
 }
 ld angle(vec a, vec b)
 {
 return acosl((a.first * b.first + a.second*b.second) / (len(a)*len(b)));
 }
 ld ar_tr(vec a, vec b)
 {
 return a.first*b.second - b.first*a.second;
 }

 bool CheckBySinusTheoreme(std::vector <ld> sortedSideLengths, std::vector <ld> sortedAngles){
  bool flag = false;
  ld tempValue = sortedSideLengths[0]/sin(sortedAngles[0] * M_PI/180);
  for(int i = 1; i < 3; i++) {
  if (std::abs(sortedSideLengths[i]/sin(sortedAngles[i] * M_PI/180) - tempValue) > eps) {
  return false;
  }
  }
  return true;
 }

 int main(int argc, char ** argv)
 {
  //TODO: Napishi menya normalno
 InitChecker(argc, argv);
 ld lena = File.ReadDouble(), anglea = File.ReadDouble(), angleb = File.ReadDouble(), anglec= 180.0 - anglea - angleb;
 vec a = mp( Out.ReadDouble(), Out.ReadDouble() ), b = mp( Out.ReadDouble(),Out.ReadDouble() ), c = mp( Out.ReadDouble(),Out.ReadDouble() );
 //sort angle by ASC = vector<ld> sortedAngles
  //Find length of all lines and sort = vector <ld> sortedLines
  //Check by sinus theoreme
  //If previous == true =>
  //Check cosinus theroeme
  //If previous == true =>
  //OK
  //else =>
  std::vector <vec> t;
  std::vector <ld> sortedAngles;
  sortedAngles.pb(anglea);sortedAngles.pb(angleb);sortedAngles.pb(anglec);
  std::sort(sortedAngles.begin(),sortedAngles.end());
  std::vector <ld> sortedSideLengths;
  sortedSideLengths.pb(len(b-a));sortedSideLengths.pb(len(c-a));sortedSideLengths.pb(len(c-b));
```

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int correctAnswer = Ans.ReadInt();
 int userLength = Out.ReadInt();
 if(correctAnswer < userLength) {
 QuitWith(WA, "Too long way");
 }
  if(correctAnswer == 0 && userLength == -1) {
  if(Out.HasInput()) {
  QuitWith(PE, "Too many input data");
  }
  QuitWith(WA, "Incorrect answer. Path exist");
  }
 if (correctAnswer == userLength && correctAnswer == 0 || correctAnswer == userLength && correctAnswer == -1) {
  if(Out.HasInput()) {
  QuitWith(PE, "Too many input data");
  }
 QuitWith(AC, "Full solution");
 }
 int n = File.ReadInt();
 std::vector<std::vector<int>> graph;
 for (int i = 0; i < n; i++) {
 std::vector<int> line;
 for (int j = 0; j < n; j++)
 line.push_back(File.ReadInt());
 graph.push_back(line);
 }
 int start = File.ReadInt(1,n), finish = File.ReadInt(1,n);
 std::vector<int> userPath;
 while (Out.HasInput()) {
```

```
#include <checkers/testlib.h>
 #include <iostream>
 #include <vector>
 #include <set>
 #include <algorithm>
 #include <map>
 #include <set>
 #include <string>
 #include <sstream>

 using namespace NTestlib;

 int main(int argc, char* argv[]) {
 InitChecker(argc, argv);
 int n1 = Out.ReadInt(), n2 = Ans.ReadInt(), n = File.ReadInt();
 if (n1 == n2 && (n1 == 0 || n1 == -1)) {
  if(Out.HasInput()) {
  QuitWith(PE, "Too many input data");
  }
  QuitWith(AC, "OK");
  }
 if ((n1 == -1) ^ (n2 == -1))
 {
 std::stringstream msg;
 msg << "WA";
 QuitWith(WA, msg.str());
 }
 if (n1 > n2)
 {
 std::stringstream msg;
 msg << "WA1";
 QuitWith(WA, msg.str());
 }
 std::vector <std::vector <int> > a(n, std::vector <int>(n));
 std::vector <int> path(n1);
 for (int i = 0; i < n; i++)
 for (int j = 0; j < n; j++)
 a[i][j] = File.ReadInt();
 int s = File.ReadInt();
 int t = File.ReadInt();
 for (int i = 0; i < n1 + 1; i++)
 {
```

```
#include <checkers/testlib.h>

 #include <iostream>

 #include <vector>

 #include <set>

 #include <algorithm>

 #include <map>

 #include <set>

 #include <string>

 #include <sstream>


 using namespace NTestlib;


 int f[8] = { -2,-2,-1,-1,1,1,2,2 };

 int s[8] = { 1,-1,2,-2,-2,2,-1,1 };


 int main(int argc, char* argv[]) {

 InitChecker(argc, argv);

 int n = File.ReadInt(), m = File.ReadInt();

 std::vector<std::vector<int>> c(n+1, std::vector<int>(m+1, 0));

 int temp = Out.ReadInt();

 if (temp == -1)

 if (n == 4 && m == 4)

 QuitWith(AC, "Full solution");

 else

 QuitWith(WA, "Wrong answer");

 if (temp < 1 || temp > n) {

 QuitWith(PE, "Index out of range");

 }

 int x = temp, y = Out.ReadInt(1, m);

 c[x][y] = 1;

 int x2 = x, y2 = y;

 for (int i = 0; i < n * m - 1; i++)

 {

  bool isFind = false;

 int x1 = Out.ReadInt(1, n), y1 = Out.ReadInt(1, m);

 for (int j = 0; j < 8; j++)

 {

 if (x1 - x == f[j] && y1 - y == s[j]) {

  isFind = true;

  break;

 }

 }
```

```
#include <checkers/testlib.h>

 using namespace NTestlib;

 int count(char i) {
 switch (i) {
 case 'L':
 return 1;
 case 'R':
 return 3;
 case 'U':
 return 7;
 case 'D':
 return 15;
 default:
 QuitWith(PE, "Incorrect input");
 }
 }

 int main(int argc, char* argv[]) {
  InitChecker(argc, argv);
  int counterAnswer = 0;
  int counterUser = 0;
  int i = 0;
  for (i = 0; Ans.HasInput() && Out.HasInput(); ++i) {
  std::string x = Ans.ReadWord();
  std::string y = Out.ReadWord();
  for(int j = 0; j < x.length(); j++) {
  counterAnswer += count(x[j]);
  }
  for(int j = 0; j < x.length(); j++) {
  counterUser += count(y[j]);
  }
  }
  if (Out.HasInput()) {
  QuitWith(WA, "Incorrect path. Your path is too long");
  }
  if (Ans.HasInput()) {
  QuitWith(WA, "Incorrect path. Your path is too short.");
  }
  if (counterAnswer != counterUser) {
   QuitWith(WA, "Incorrect path. Path is not available");
```