#include <iostream>

#include<string>

#include<cctype>

using namespace std;

bool isMotionMeaningful(string motion);

bool translatable(string motion);

bool isoneSlash(string motion);

bool isendPrematurely(string motion);

bool holdLessThan2(string motion);

int badBeatF(string motion);

int translateMotion(string motion, string& instruction, int& badBeat);

string instructionF(string motion);

int main()

{

string motion = "0s/w/2w//";

string instruction;

int badBeat;

/\*if (isMotionMeaningful(motion))

{

cout << "The string " << motion << " is meaninful" << endl;

}

else

{

cout << "is not meaningful" << endl;

}

\*/

/\*if (translatable(motion))

cout << motion << "pass" << endl;

else

cout << motion << "fail" << endl;

\*/

//cout << badBeatF(motion);

cout << (translateMotion(motion, instruction, badBeat)) << endl;

cout << "badBeat is " << badBeat;

/\*{

cout << "instruction is " <<instruction << endl;

cout << "badBeat is " << badBeat;

}\*/

}

bool isMotionMeaningful(string motion)

{

int add = 0;

size\_t i = 0;

while (i < motion.size())

{

if (motion[i] == '/')

{

add++;

}

else if (isalpha(motion[i]))

{

if (motion[i] == 'w' || motion[i] == 'W' || motion[i] == 's' || motion[i] == 'S' || motion[i] == 'a' || motion[i] == 'A' || motion[i] == 'd' || motion[i] == 'D')

{

if (motion[i+1] == '/')

add += 2;

else

return false;

}

else

return false;

}

else if (isdigit(motion[i]))

{

if(isdigit(motion[i+1]))

{

if(motion[i+2] == 'w' || motion[i+2] == 'W' || motion[i+2] == 's' || motion[i+2] == 'S' || motion[i+2] == 'a' || motion[i+2] == 'A' || motion[i+2] == 'd' || motion[i+2] == 'D')

{

if (motion[i+3] == '/')

add += 4;

else

return false;

}

else

return false;

}

if (motion[i+1] == 'w' || motion[i+1] == 'W' || motion[i+1] == 's' || motion[i+1] == 'S' || motion[i+1] == 'a' || motion[i+1] == 'A' || motion[i+1] == 'd' || motion[i+1] == 'D')

{

add += 2;

}

else

return false;

}

else

return false;

i += add;

}

return true;

}

int translateMotion(string motion, string& instruction, int& badBeat)

{

if (translatable(motion))

{

instruction = instructionF(motion);

return 0;

}

else if (!isMotionMeaningful(motion))

{

return 1;

}

else if (isMotionMeaningful(motion) && !isendPrematurely(motion))

{

badBeat = 1 + badBeatF(motion);

return 3;

}

else if (isMotionMeaningful(motion) && !isoneSlash(motion))

{

badBeat = badBeatF(motion);

return 2;

}

else if (isMotionMeaningful(motion) && !holdLessThan2(motion))

{

badBeat = badBeatF(motion);

return 4;

}

//motionmeaninful but not translatable for more than one reason

}

bool translatable(string motion)

{

if(!(isMotionMeaningful(motion) && isoneSlash(motion) && isendPrematurely(motion) && holdLessThan2(motion)))

return false;

else

return true;

}

bool isoneSlash(string motion)

{

int a = 0;

int num = 0;

int count = 0;

int add = 0;

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

{

if (isdigit(motion[i]) && isdigit(motion[i+1]))

{

num = 10 \* (motion[i] - '0') + (motion[i+1] - '0');

size\_t k = i+2;

a = k + num;

for (; k <= a && k < motion.size(); k++)

if (motion[k] == '/')

count++;

i += num + 1;

}

else if (isdigit(motion[i]))

{

num = motion[i] - '0';

size\_t k = i+2;

a = k + num;

for (; k <= a && k < motion.size(); k++)

if (motion[k] == '/')

count++;

}

if (count != num)

{

return false;

}

}

return true;

}

bool isendPrematurely(string motion)

{

int a = 0;

int num = 0;

int count = 0;

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

{

if (isdigit(motion[i]) && isdigit(motion[i+1]))

{

num = 10 \* (motion[i]- '0') + motion[i+1] - '0';

size\_t k = i+2;

a = k + num;

for (; k < a; k++)

if(k >= motion.size())

return false;

}

else if (isdigit(motion[i]))

{

num = motion[i] - '0';

size\_t k = i+2;

a = k + num;

for (; k < a ; k++)

if(k >= motion.size())

return false;

}

}

return true;

}

bool holdLessThan2(string motion)

{

int num = 0;

int count = 0;

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

{

if (isdigit(motion[i]) && isdigit(motion[i+1]))

{

num = 10 \* (motion[i] - '0') + motion[i+1] - '0';

i += num + 1;

}

else if (isdigit(motion[i]))

{

num = motion[i] - '0';

}

else if (isalpha(motion[i]) || motion[i] == '/')

{

continue;

}

if (num == 0 || num == 1)

return false;

}

return true;

}

int badBeatF(string motion)

{

int a = 0;

int num = 0;

int count = 0;

int add = 0;

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

{

if (isdigit(motion[i]) && isdigit(motion[i+1]))

{

num = 10 \* (motion[i] - '0') + (motion[i+1] - '0');

size\_t k = i+2;

a = k + num;

for (; k <= a && k < motion.size(); k++)

if (motion[k] == '/')

count++;

i += num + 1;

}

else if (isdigit(motion[i]))

{

num = motion[i] - '0';

size\_t k = i+2;

a = k + num;

for (; k <= a && k < motion.size(); k++)

if (motion[k] == '/')

count++;

}

if (count != num)

{

break;

}

}

return num;

}

string instructionF(string motion)

{

string instruction = "";

string hold = "";

string direction = "";

string period = "";

int num = 0;

int add = 0;

size\_t i = 0;

while (i < motion.size())

{

if (isdigit(motion[i]) && isdigit(motion[i+1]))

{

num = 10 \* (motion[i] - '0') + motion[i+1] - '0';

for (int k = 0; k < num; k++)

hold += toupper(motion[i+2]);

i += num + 2;

}

else if (isdigit(motion[i]))

{

num = motion[i] - '0';

for (int k = 0; k < num; k++)

hold += toupper(motion[i+1]);

i += num + 1;

}

else if (isalpha(motion[i]))

{

direction = tolower(motion[i]);

i += 1;

}

else if (motion[i] == '/')

{

period = ".";

}

instruction += hold + direction + period;

i++;

}

return instruction;

}