#include <iostream>

#include <string>

#include <vector>

#include <sstream>

#include <iomanip>

using namespace std;

struct Inst

{

bool issued;

string I;

string N;

string D;

string S1;

string S2;

};

struct RS\_Inst

{

int order;

string O;

string s1;

string s2;

};

class RF

{

public:

RF()

{

int RF\_content = 0;

RFs.assign(6, to\_string(RF\_content));

for (int i = 1; i <= 5; i++)

{

RFs[i] = to\_string(RF\_content);

RF\_content += 2;

}

}

void printAll()

{

cout << " \_ RF \_\_\n";

cout << " F1 | " << setw(2) << RFs[1] << " | \n";

cout << " F2 | " << setw(2) << RFs[2] << " | \n";

cout << " F3 | " << setw(2) << RFs[3] << " | \n";

cout << " F4 | " << setw(2) << RFs[4] << " | \n";

cout << " F5 | " << setw(2) << RFs[5] << " | \n";

cout << " -------\n";

}

string get\_RFs(int i)

{

return RFs[i];

}

void set\_RFs(int i, string data)

{

RFs[i] = data;

}

private:

vector<string> RFs;

};

class RAT

{

public:

RAT()

{

string RAT\_content = " ";

RATs.assign(6, RAT\_content);

}

void print\_all()

{

cout << " \_ RAT \_\_\n";

cout << " F1 | " << RATs[1] << " |\n";

cout << " F2 | " << RATs[2] << " |\n";

cout << " F3 | " << RATs[3] << " |\n";

cout << " F4 | " << RATs[4] << " |\n";

cout << " F5 | " << RATs[5] << " |\n";

cout << " --------\n";

}

string get\_RATs(int i)

{

return RATs[i];

}

void set\_RATs(int i, string data)

{

RATs[i] = data;

}

bool empty()

{

for (int i = 1; i <= 5; i++)

{

if (RATs[i] != " ")

{

return false;

}

}

return true;

}

private:

vector<string> RATs;

};

class RS

{

public:

RS()

{

RS\_Inst RS\_content;

RS\_content.O = " ";

RS\_content.s1 = " ";

RS\_content.s2 = " ";

RSs.assign(6, RS\_content);

Buffers.assign(2, "empty");

}

void print\_all()

{

cout << " \_ RS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "RS1 | " << RSs[1].O << " | " << setw(3) << RSs[1].s1 << " | " << setw(3) << RSs[1].s2 << " |\n";

cout << "RS2 | " << RSs[2].O << " | " << setw(3) << RSs[2].s1 << " | " << setw(3) << RSs[2].s2 << " |\n";

cout << "RS3 | " << RSs[3].O << " | " << setw(3) << RSs[3].s1 << " | " << setw(3) << RSs[3].s2 << " |\n";

cout << " ----------------------\n";

cout << "BUFFER: " << Buffers[0] << "\n";

cout << "\n";

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "RS4 | " << RSs[4].O << " | " << setw(3) << RSs[4].s1 << " | " << setw(3) << RSs[4].s2 << " |\n";

cout << "RS5 | " << RSs[5].O << " | " << setw(3) << RSs[5].s1 << " | " << setw(3) << RSs[5].s2 << " |\n";

cout << " ----------------------\n";

cout << "BUFFER: " << Buffers[1] << "\n";

}

int left\_not\_full()

{

for (int i = 1; i <= 3; i++)

{

if (RSs[i].O == " " && RSs[i].s1 == " " && RSs[i].s2 == " ")

{

return i;

}

}

return 0;

}

int right\_not\_full()

{

for (int i = 4; i <= 5; i++)

{

if (RSs[i].O == " " && RSs[i].s1 == " " && RSs[i].s2 == " ")

{

return i;

}

}

return 0;

}

RS\_Inst get\_RSs(int i)

{

return RSs[i];

}

void set\_RSs(int i, RS\_Inst data)

{

RSs[i] = data;

}

void set\_RSs\_1(int i, string data)

{

RSs[i].s1 = data;

}

void set\_RSs\_2(int i, string data)

{

RSs[i].s2 = data;

}

string get\_Buffers(int i)

{

return Buffers[i];

}

void set\_Buffers(int i, string data)

{

Buffers[i] = data;

}

void change\_order(int i, int data)

{

RSs[i].order = data;

}

bool empty()

{

for (int i = 1; i <= 5; i++)

{

if (RSs[i].O != " " || RSs[i].s1 != " " || RSs[i].s2 != " ")

{

return false;

}

}

return true;

}

private:

vector<RS\_Inst> RSs;

vector<string> Buffers;

};

int main()

{

vector<Inst> IQ;

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

{

string buffer = "";

getline(cin, buffer);

Inst temp;

temp.I = buffer;

stringstream ss;

ss << buffer;

ss >> buffer;

temp.N = buffer;

ss >> buffer;

buffer.erase(buffer.end() - 1);

temp.D = buffer;

ss >> buffer;

buffer.erase(buffer.end() - 1);

temp.S1 = buffer;

ss >> buffer;

temp.S2 = buffer;

temp.issued = false;

IQ.emplace\_back(temp);

}

int cycle = 0;

RF register\_file;

RAT register\_allocation\_table;

RS reservation\_station;

while (cycle < 13)

{

++cycle;

for (int i = 1; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

reservation\_station.change\_order(i, reservation\_station.get\_RSs(i).order + 1);

}

}

vector<Inst>::iterator it = IQ.end();

for (int i = 0; i < IQ.size(); i++)

{

if (!IQ[i].issued)

{

it = IQ.begin() + i;

break;

}

}

if (it != IQ.end())

{

if (it->N == "ADDI" || it->N == "ADD" || it->N == "SUB")

{

if (reservation\_station.left\_not\_full() != 0)

{

RS\_Inst current;

if (it->N == "ADDI")

{

current.O = "+";

string register\_num\_1 = it->S1;

register\_num\_1.erase(register\_num\_1.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_1)) != " ")

{

current.s1 = register\_allocation\_table.get\_RATs(stoi(register\_num\_1));

}

else

{

current.s1 = register\_file.get\_RFs(stoi(register\_num\_1));

}

current.s2 = it->S2;

current.order = 0;

}

else if (it->N == "ADD")

{

current.O = "+";

string register\_num\_1 = it->S1;

register\_num\_1.erase(register\_num\_1.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_1)) != " ")

{

current.s1 = register\_allocation\_table.get\_RATs(stoi(register\_num\_1));

}

else

{

current.s1 = register\_file.get\_RFs(stoi(register\_num\_1));

}

string register\_num\_2 = it->S2;

register\_num\_2.erase(register\_num\_2.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_2)) != " ")

{

current.s2 = register\_allocation\_table.get\_RATs(stoi(register\_num\_2));

}

else

{

current.s2 = register\_file.get\_RFs(stoi(register\_num\_2));

}

current.order = 0;

}

else if (it->N == "SUB")

{

current.O = "-";

string register\_num\_1 = it->S1;

register\_num\_1.erase(register\_num\_1.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_1)) != " ")

{

current.s1 = register\_allocation\_table.get\_RATs(stoi(register\_num\_1));

}

else

{

current.s1 = register\_file.get\_RFs(stoi(register\_num\_1));

}

string register\_num\_2 = it->S2;

register\_num\_2.erase(register\_num\_2.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_2)) != " ")

{

current.s2 = register\_allocation\_table.get\_RATs(stoi(register\_num\_2));

}

else

{

current.s2 = register\_file.get\_RFs(stoi(register\_num\_2));

}

current.order = 0;

}

string register\_num = it->D;

register\_num.erase(register\_num.begin());

int left\_RS\_empty = reservation\_station.left\_not\_full();

switch (left\_RS\_empty)

{

case 1:

reservation\_station.set\_RSs(1, current);

register\_allocation\_table.set\_RATs(stoi(register\_num), "RS1");

break;

case 2:

reservation\_station.set\_RSs(2, current);

register\_allocation\_table.set\_RATs(stoi(register\_num), "RS2");

break;

case 3:

reservation\_station.set\_RSs(3, current);

register\_allocation\_table.set\_RATs(stoi(register\_num), "RS3");

break;

default:

break;

}

it->issued = true;

}

}

else if (it->N == "MUL" || it->N == "DIV")

{

if (reservation\_station.right\_not\_full() != 0)

{

RS\_Inst current;

if (it->N == "MUL")

{

current.O = "\*";

string register\_num\_1 = it->S1;

register\_num\_1.erase(register\_num\_1.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_1)) != " ")

{

current.s1 = register\_allocation\_table.get\_RATs(stoi(register\_num\_1));

}

else

{

current.s1 = register\_file.get\_RFs(stoi(register\_num\_1));

}

string register\_num\_2 = it->S2;

register\_num\_2.erase(register\_num\_2.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_2)) != " ")

{

current.s2 = register\_allocation\_table.get\_RATs(stoi(register\_num\_2));

}

else

{

current.s2 = register\_file.get\_RFs(stoi(register\_num\_2));

}

current.order = 0;

}

else if (it->N == "DIV")

{

current.O = "/";

string register\_num\_1 = it->S1;

register\_num\_1.erase(register\_num\_1.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_1)) != " ")

{

current.s1 = register\_allocation\_table.get\_RATs(stoi(register\_num\_1));

}

else

{

current.s1 = register\_file.get\_RFs(stoi(register\_num\_1));

}

string register\_num\_2 = it->S2;

register\_num\_2.erase(register\_num\_2.begin());

if (register\_allocation\_table.get\_RATs(stoi(register\_num\_2)) != " ")

{

current.s2 = register\_allocation\_table.get\_RATs(stoi(register\_num\_2));

}

else

{

current.s2 = register\_file.get\_RFs(stoi(register\_num\_2));

}

current.order = 0;

}

string register\_num = it->D;

register\_num.erase(register\_num.begin());

int right\_RS\_empty = reservation\_station.right\_not\_full();

switch (right\_RS\_empty)

{

case 4:

reservation\_station.set\_RSs(4, current);

register\_allocation\_table.set\_RATs(stoi(register\_num), "RS4");

break;

case 5:

reservation\_station.set\_RSs(5, current);

register\_allocation\_table.set\_RATs(stoi(register\_num), "RS5");

break;

default:

break;

}

it->issued = true;

}

}

}

if (reservation\_station.get\_Buffers(0) == "empty")

{

int min\_order\_1 = 10000;

for (int i = 1; i <= 3; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (reservation\_station.get\_RSs(i).s1.find('R') == string::npos && reservation\_station.get\_RSs(i).s2.find('R') == string::npos)

{

if (reservation\_station.get\_RSs(i).order > 0)

{

if (reservation\_station.get\_RSs(i).order < min\_order\_1)

{

min\_order\_1 = reservation\_station.get\_RSs(i).order;

}

}

}

}

}

string buffer\_content\_0 = "empty";

for (int i = 1; i <= 3; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (reservation\_station.get\_RSs(i).order == min\_order\_1)

{

buffer\_content\_0 = "(RS" + to\_string(i) + ") " + reservation\_station.get\_RSs(i).s1 + " " + reservation\_station.get\_RSs(i).O + " " + reservation\_station.get\_RSs(i).s2;

reservation\_station.set\_Buffers(0, buffer\_content\_0);

break;

}

}

}

}

else

{

for (int i = 1; i <= 3; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (i != reservation\_station.get\_Buffers(0)[3] - 48 && reservation\_station.get\_RSs(i).order > 0)

{

reservation\_station.change\_order(i, reservation\_station.get\_RSs(i).order - 1);

}

}

}

}

if (reservation\_station.get\_Buffers(0) == "empty" && reservation\_station.get\_Buffers(1) == "empty")

{

int min\_order\_2 = 10000;

for (int i = 4; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (reservation\_station.get\_RSs(i).s1.find('R') == string::npos && reservation\_station.get\_RSs(i).s2.find('R') == string::npos)

{

if (reservation\_station.get\_RSs(i).order > 0)

{

if (reservation\_station.get\_RSs(i).order < min\_order\_2)

{

min\_order\_2 = reservation\_station.get\_RSs(i).order;

}

}

}

}

}

string buffer\_content\_1 = "empty";

for (int i = 4; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (reservation\_station.get\_RSs(i).order == min\_order\_2)

{

buffer\_content\_1 = "(RS" + to\_string(i) + ") " + reservation\_station.get\_RSs(i).s1 + " " + reservation\_station.get\_RSs(i).O + " " + reservation\_station.get\_RSs(i).s2;

reservation\_station.set\_Buffers(1, buffer\_content\_1);

break;

}

}

}

}

else

{

for (int i = 4; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).O != " ")

{

if (i != reservation\_station.get\_Buffers(1)[3] - 48 && reservation\_station.get\_RSs(i).order > 0)

{

reservation\_station.change\_order(i, reservation\_station.get\_RSs(i).order - 1);

}

}

}

}

if (reservation\_station.get\_Buffers(0) != "empty" && reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).order >= 2)

{

int t1, t2, result;

if (reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).O == "+")

{

t1 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).s1);

t2 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).s2);

result = t1 + t2;

}

else if (reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).O == "-")

{

t1 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).s1);

t2 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(0)[3] - 48).s2);

result = t1 - t2;

}

for (int i = 1; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).s1 == reservation\_station.get\_Buffers(0).substr(1, 3))

{

reservation\_station.set\_RSs\_1(i, to\_string(result));

}

if (reservation\_station.get\_RSs(i).s2 == reservation\_station.get\_Buffers(0).substr(1, 3))

{

reservation\_station.set\_RSs\_2(i, to\_string(result));

}

if (register\_allocation\_table.get\_RATs(i) == reservation\_station.get\_Buffers(0).substr(1, 3))

{

register\_file.set\_RFs(i, to\_string(result));

register\_allocation\_table.set\_RATs(i, " ");

}

}

RS\_Inst new\_RS\_content;

new\_RS\_content.O = " ";

new\_RS\_content.s1 = " ";

new\_RS\_content.s2 = " ";

new\_RS\_content.order = 0;

reservation\_station.set\_RSs(reservation\_station.get\_Buffers(0)[3] - 48, new\_RS\_content);

reservation\_station.set\_Buffers(0, "empty");

}

else if (reservation\_station.get\_Buffers(1) != "empty" && reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).order >= 2)

{

int t1, t2, result;

if (reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).O == "\*")

{

t1 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).s1);

t2 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).s2);

result = t1 \* t2;

}

else if (reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).O == "/")

{

t1 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).s1);

t2 = stoi(reservation\_station.get\_RSs(reservation\_station.get\_Buffers(1)[3] - 48).s2);

result = t1 / t2;

}

for (int i = 1; i <= 5; i++)

{

if (reservation\_station.get\_RSs(i).s1 == reservation\_station.get\_Buffers(1).substr(1, 3))

{

reservation\_station.set\_RSs\_1(i, to\_string(result));

}

if (reservation\_station.get\_RSs(i).s2 == reservation\_station.get\_Buffers(1).substr(1, 3))

{

reservation\_station.set\_RSs\_2(i, to\_string(result));

}

if (register\_allocation\_table.get\_RATs(i) == reservation\_station.get\_Buffers(1).substr(1, 3))

{

register\_file.set\_RFs(i, to\_string(result));

register\_allocation\_table.set\_RATs(i, " ");

}

}

RS\_Inst new\_RS\_content;

new\_RS\_content.O = " ";

new\_RS\_content.s1 = " ";

new\_RS\_content.s2 = " ";

new\_RS\_content.order = 0;

reservation\_station.set\_RSs(reservation\_station.get\_Buffers(1)[3] - 48, new\_RS\_content);

reservation\_station.set\_Buffers(1, "empty");

}

cout << "Cycle: " << cycle << "\n\n";

register\_file.printAll();

cout << "\n";

register\_allocation\_table.print\_all();

cout << "\n";

reservation\_station.print\_all();

cout << "\n";

}

return 0;

}