**#Solution-1:**

#include <iostream>

using namespace std;

#define MAX 10

class student {

private:

char name[30];

int courseNum;

int total;

float perc;

public:

void getDetails(void);

void putDetails(void);

};

void student::getDetails(void)

{

cout << "Enter name: ";

cin >> name;

cout << "Enter course number: ";

cin >> courseNum;

cout << "Enter total marks out of 500: ";

cin >> total;

perc = (float)total / 500 \* 100;

}

void student::putDetails(void)

{

cout << "Student details:\n";

cout << "Name: " << name << ", Course Number: " << courseNum

<< ", Total: " << total << ", Percentage: " << perc;

}

int main(void) {

int numStudents;

cout << "Enter total number of students: ";

cin >> numStudents;

student students[MAX]; // array of objects creation

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

cout << "Enter details of student " << i + 1 << ":\n";

students[i].getDetails();

}

cout << endl;=

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

cout << "Details of student " << i + 1 << ":\n";

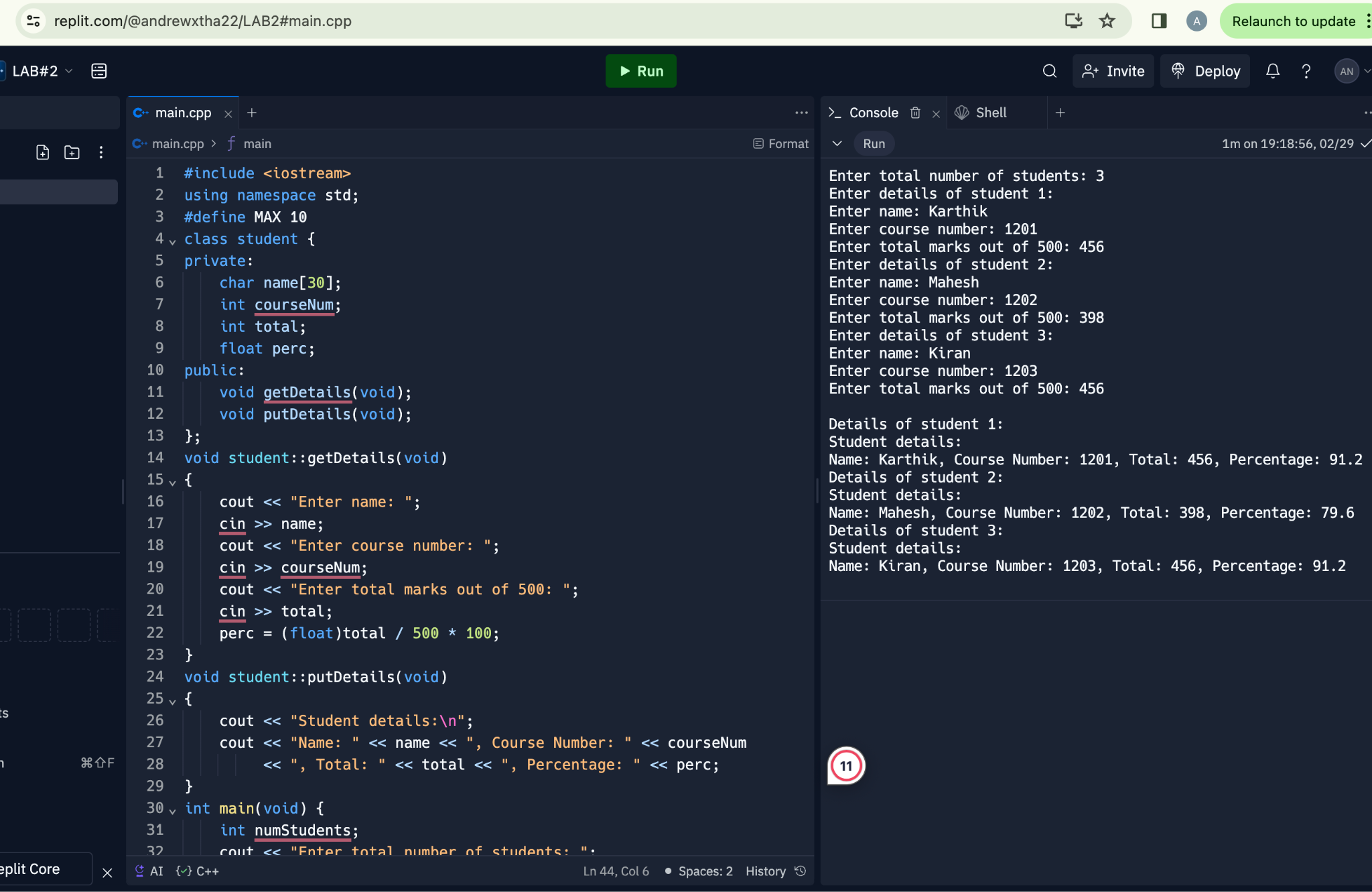
students[i].putDetails();

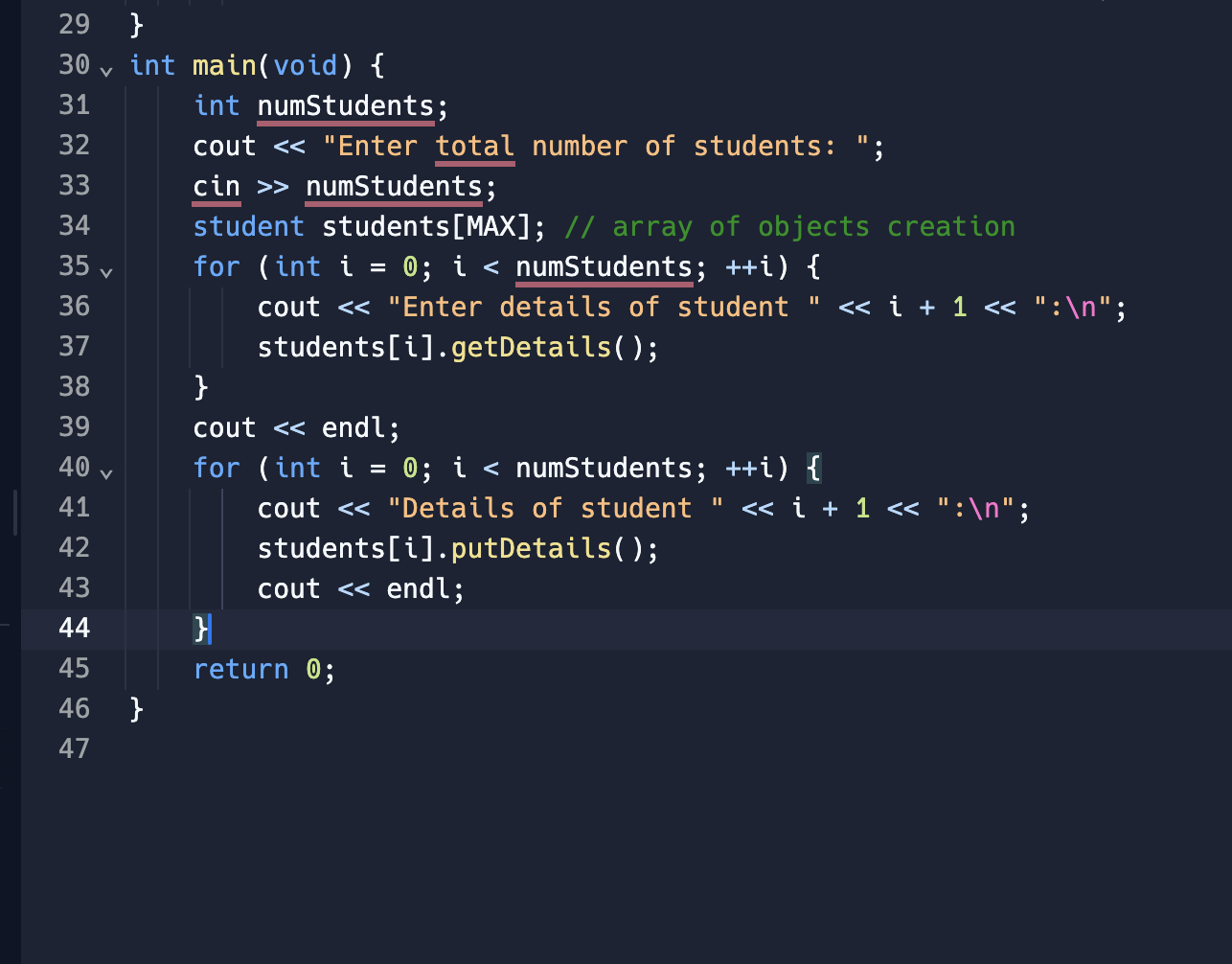
cout << endl;

}

return 0;

}

****

****

**#Solution-2:**

#include <iostream>

using namespace std;

class sample {

private:

int a;

char b;

float c;

public:

void get\_data() {

cout << "Enter an integer value: ";

cin >> a;

cout << "Enter a character: ";

cin >> b;

cout << "Enter a float value: ";

cin >> c;

}

void print\_data() {

cout << "Values read from keyboard are:\n";

cout << "Integer value: " << a << endl;

cout << "Character is: " << b << endl;

cout << "Float value is: " << c << endl;

}

};

int main(void) {

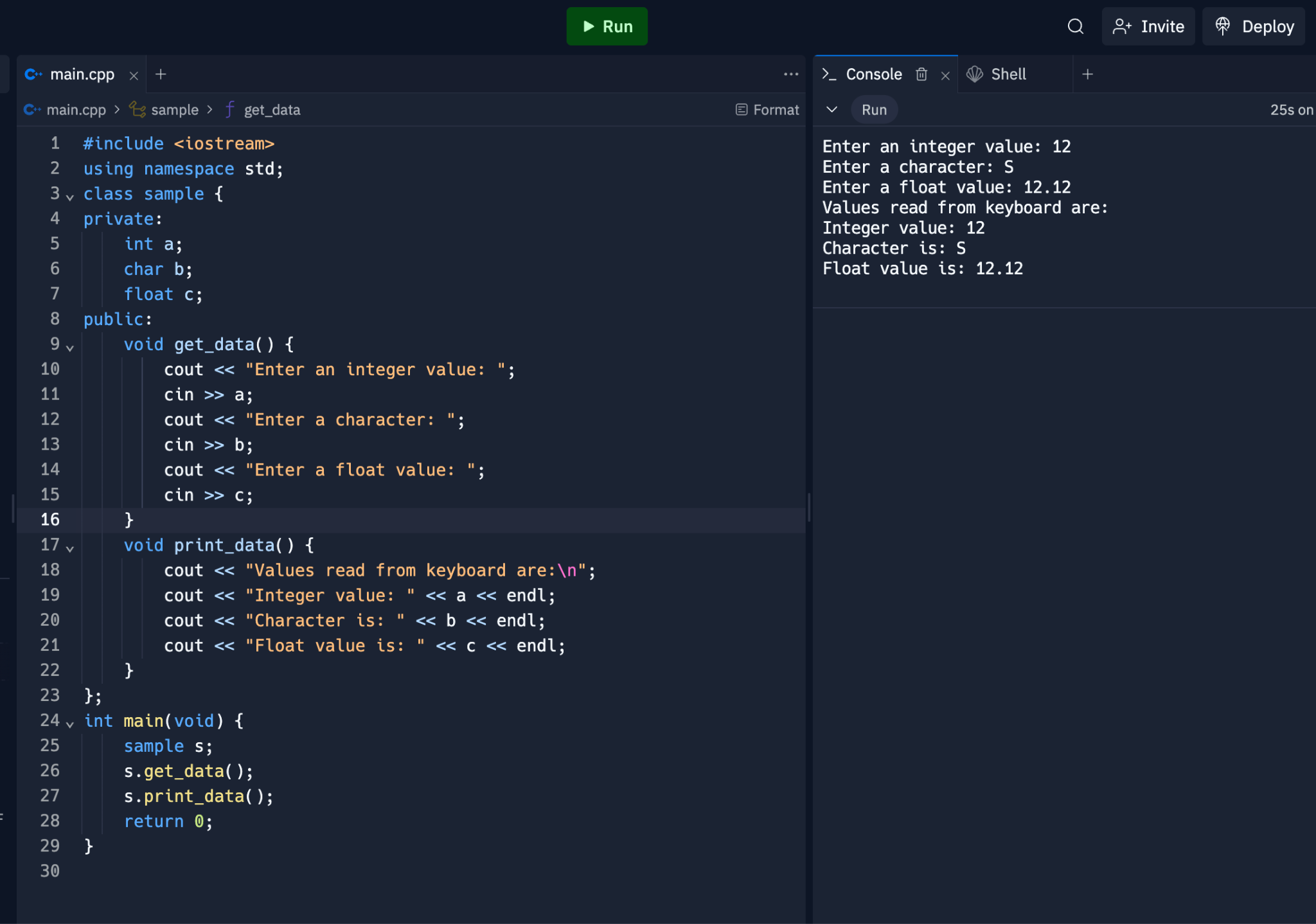
sample s;

s.get\_data();

s.print\_data();

return 0;

}



**#Solution-3:**

#include <iostream>

class Rectangle {

private:

float length;

float width;

public:

Rectangle(float len, float wid) : length(len), width(wid) {}

void set\_length(float len) {

// Setting the length data member.

length = len;

}

void set\_width(float wid) {

// Setting the width data member.

width = wid;

}

float perimeter() {

// Calculate and return the perimeter of the rectangle.

return 2 \* (length + width);

}

float area() {

// Calculate and return the area of the rectangle.

return length \* width;

}

void show() {

// Display the length and width of the rectangle.

std::cout << "Rectangle: Length = " << length << ", Width = " << width << std::endl;

}

int same\_area(Rectangle other\_rectangle) {

// Check if two rectangles have the same area.

// Returns 1 if they have the same area, otherwise returns 0.

return area() == other\_rectangle.area();

}

};

int main() {

// Create two rectangle objects

Rectangle rectangle1(5, 2.5);

Rectangle rectangle2(5, 18.9);

// Display information for each rectangle

rectangle1.show();

std::cout << "Area: " << rectangle1.area() << ", Perimeter: " << rectangle1.perimeter() << std::endl;

rectangle2.show();

std::cout << "Area: " << rectangle2.area() << ", Perimeter: " << rectangle2.perimeter() << std::endl;

// Check if the rectangles have the same area

if (rectangle1.same\_area(rectangle2)) {

std::cout << "The two rectangles have the same area." << std::endl;

} else {

std::cout << "The two rectangles do not have the same area." << std::endl;

}

// Modify rectangle1 dimensions

rectangle1.set\_length(15);

rectangle1.set\_width(6.3);

// Display updated information for rectangle1

rectangle1.show();

std::cout << "Area: " << rectangle1.area() << ", Perimeter: " << rectangle1.perimeter() << std::endl;

// Check again if the rectangles have the same area

if (rectangle1.same\_area(rectangle2)) {

std::cout << "The two rectangles have the same area." << std::endl;

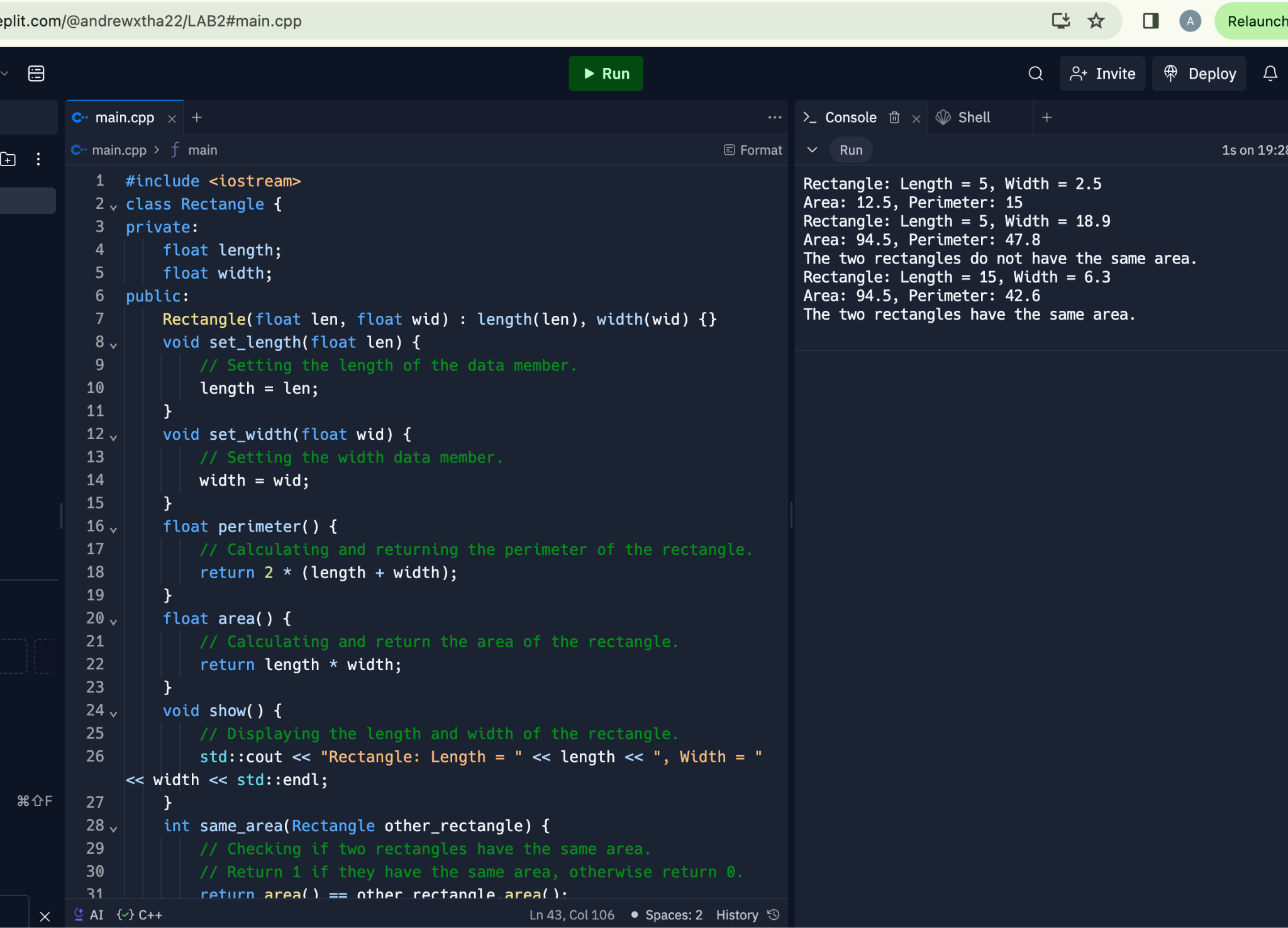
} else {

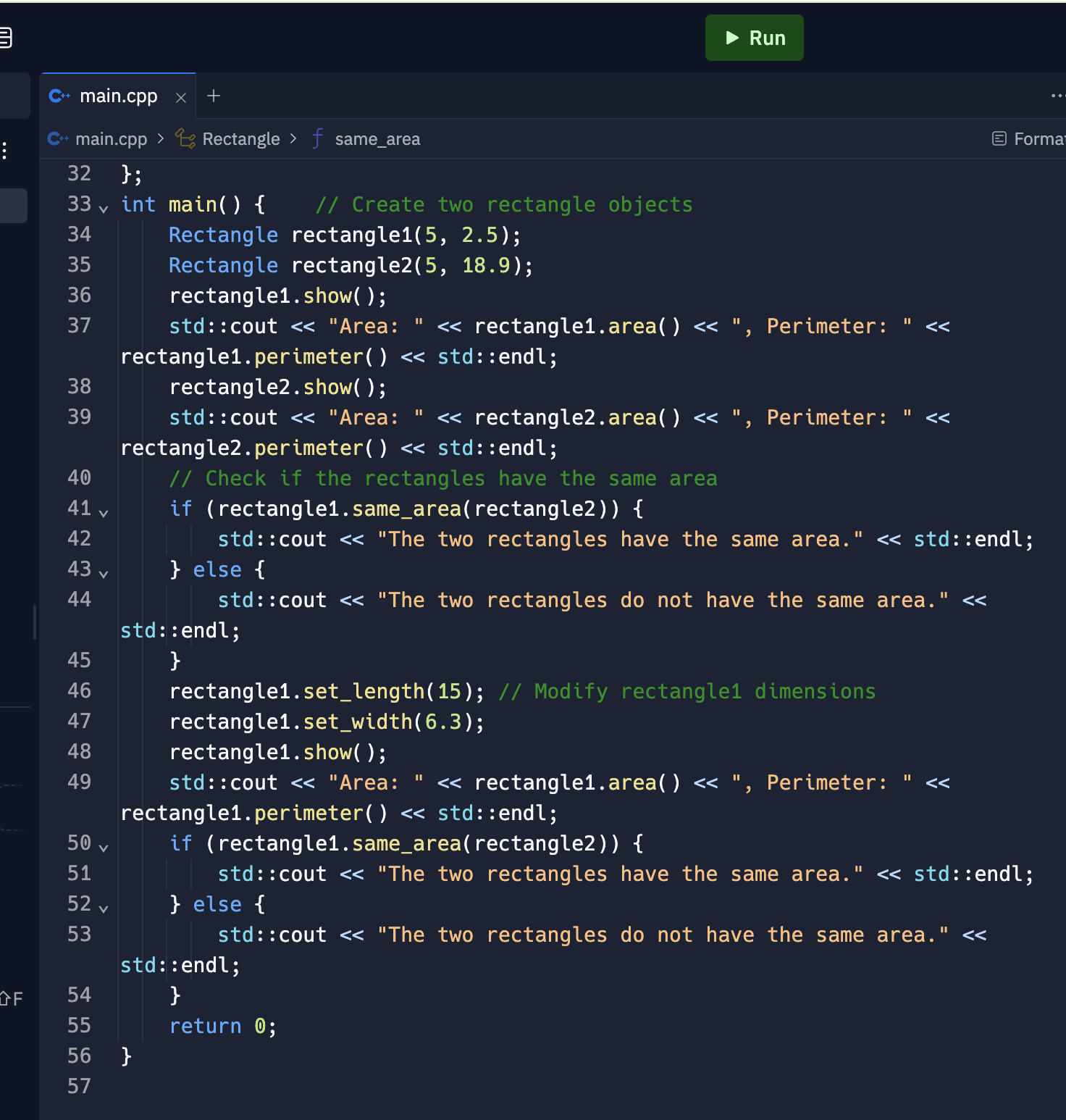
std::cout << "The two rectangles do not have the same area." << std::endl;

}

return 0;

}





**#Solution-4:**

#include <iostream>

#include <string>

#include <vector>

class MusicIns {

private:

std::vector<std::string> stringInstruments;

std::vector<std::string> windInstruments;

std::vector<std::string> percInstruments;

public:

// Initializing string instruments

void string() {

stringInstruments = {"Veena", "guitar", "sitar", "sarod", "mandolin"};

}

// Initializing wind instruments

void wind() {

windInstruments = {"Flute", "clarinet", "saxophone", "nadaswaram", "piccolo"};

}

// Initializing percussion instruments

void perc() {

percInstruments = {"Table", "mridangam", "bongos", "drums", "tambour"};

}

// Displaying instrumental arrays

void show() {

std::cout << "String Instruments:\n";

for (const auto& instrument : stringInstruments) {

std::cout << "- " << instrument << "\n";

}

std::cout << "\nWind Instruments:\n";

for (const auto& instrument : windInstruments) {

std::cout << "- " << instrument << "\n";

}

std::cout << "\nPercussion Instruments:\n";

for (const auto& instrument : percInstruments) {

std::cout << "- " << instrument << "\n";

}

}

};

int main() {

MusicIns music;

music.string();

music.wind();

music.perc();

std::cout << "Lists of the instrument arrays:\n";

music.show();

return 0;

}

