# Short Question

The program below is expected to add and subtract two complex numbers. The program is using operator overloading feature of C++ to do that. Some of the lines of code are missing the program below, fill in those lines so that program can show an output below:

Result of addition: 4 + 2i

Result of subtraction: 2 + 6i

#include <iostream>

class ComplexNumber {

private:

int realPart;

int imaginaryPart;

public:

ComplexNumber(int real = 0, int imag = 0)

{

this->realPart = real;

this->imaginaryPart = imag;

}

ComplexNumber operator+(const ComplexNumber& other) {

// Fill in this function

}

ComplexNumber operator-(const ComplexNumber& other) {

// Fill in this function

}

void display() {

if (imaginaryPart < 0)

std::cout << realPart << " - " << -imaginaryPart << "i" << std::endl;

else

std::cout << realPart << " + " << imaginaryPart << "i" << std::endl;

}

};

int main() {

ComplexNumber num1(3, 4);

ComplexNumber num2(1, -2);

ComplexNumber result;

result = num1 + num2;

std::cout << "Result of addition: ";

result.display();

result = num1 - num2;

std::cout << "Result of subtraction: ";

result.display();

return 0;

}

**Answer:**

#include <iostream>

class ComplexNumber {

private:

int realPart;

int imaginaryPart;

public:

ComplexNumber(int real = 0, int imag = 0)

{

this->realPart = real;

this->imaginaryPart = imag;

}

// Operator overloading function for addition (+)

ComplexNumber operator+(const ComplexNumber& other) {

ComplexNumber result;

result.realPart = this->realPart + other.realPart;

result.imaginaryPart = this->imaginaryPart + other.imaginaryPart;

return result;

}

// Operator overloading function for subtraction (-)

ComplexNumber operator-(const ComplexNumber& other) {

ComplexNumber result;

result.realPart = this->realPart - other.realPart;

result.imaginaryPart = this->imaginaryPart - other.imaginaryPart;

return result;

}

void display() {

if (imaginaryPart < 0)

std::cout << realPart << " - " << -imaginaryPart << "i" << std::endl;

else

std::cout << realPart << " + " << imaginaryPart << "i" << std::endl;

}

};

int main() {

ComplexNumber num1(3, 4);

ComplexNumber num2(1, -2);

ComplexNumber result;

result = num1 + num2;

std::cout << "Result of addition: ";

result.display();

result = num1 - num2;

std::cout << "Result of subtraction: ";

result.display();

return 0;

}

# Essay Question

1. Write a C++ program that generates robot objects and displays the details of the robots in the designated areas. Your program should use Modular Programming approach to write code for this question. The program is expected to have a Robot class with the following details:

* private:
  + char\* name;
  + char\* location;
  + double weight;
  + double height;
  + double speed;
* public:
  + Robot()
    - Initialize data members to appropriate values.
  + Robot(const char \* name, const char\* location, double weight, double height, double speed)
    - Initialize values using parameterized constructor
  + ~Robot()
    - Deallocate memory
  + double getSpeed()
    - Get robot speed
  + void display()
    - Displays the robot. Check output for more info.
  + bool isValid()
    - Checks all the data members for appropriate values.

Besides the class, your program should also have 2 global functions in the header file:

* int fastest\_robot(Robot\* robot, const int num\_robots)
  + Checks for the fastest robot in the collection of robots. It should return the index of the fastest robot in the collection.
* void conrtolRooomReport(Robot\* robot, int num\_robots)
  + This should be your core function called from the main method. This function checks for all the valid robots and then display them using display method.
  + This function should also check for all the broken robots (robots that are not valid) and should display all of these robots with appropriate heading (check output for more info)
  + This function should also be responsible for calling the fastest\_robot function and display the details of the fastest robot.

#include "Robot.h"

int main()

{

const int num\_robots = 5;

Robot robot[num\_robots] = {

{ "KC1", "kitchen", 25.33, 4.55, 2.2, false },

{ "BR1", "bedroom", 5.22, 2.54, 2.2, false },

{ "Broken", "Bedroom", 10.12, 2.5, 0, false },

{ "KC2", "kitchen", 20.56, 5, 3.5, false },

{ "BR2", "bedroom", 25.32, 6.5, 3.1, false }

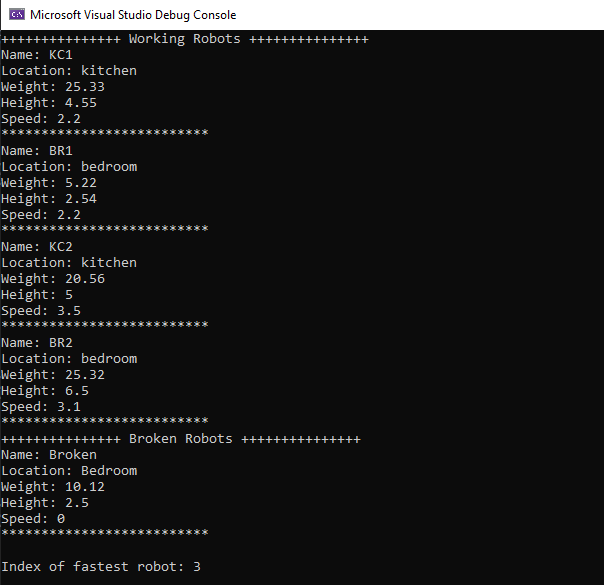
};

conrtolRooomReport(robot, 5);

return 0;

}

Below is the expected output of the program:



**Answer:**

Robot.h

#include <iostream>

#include <cstring>

class Robot

{

private:

char\* name;

char\* location;

double weight;

double height;

double speed;

public:

Robot();

Robot(const char \* name, const char \* location, double weight, double height, double speed);

~Robot();

bool isValid();

void display();

double getSpeed();

};

int fastest\_robot(Robot\* robots, const int num\_robots);

void conrtolRoomReport(Robot\* robots, int num\_robots);

Robot.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Robot.h"

Robot::Robot()

{

this->name = nullptr;

this->location = nullptr;

this->weight = 0.0;

this->height = 0.0;

this->speed = 0.0;

}

Robot::Robot(const char\* name, const char\* location, double weight, double height, double speed)

{

this->name = new char[strlen(name) + 1];

strcpy(this->name, name);

this->location = new char[strlen(location) + 1];

strcpy(this->location, location);

this->weight = weight;

this->height = height;

this->speed = speed;

}

Robot::~Robot()

{

delete[] this->name;

delete[] this->location;

}

bool Robot::isValid()

{

return (this->name != nullptr && this->location != nullptr && this->weight != 0 && this->height != 0 && this->speed != 0);

}

void Robot::display()

{

std::cout << "Name: " << this->name << std::endl;

std::cout << "Location: " << this->location << std::endl;

std::cout << "Weight: " << this->weight << std::endl;

std::cout << "Height: " << this->height << std::endl;

std::cout << "Speed: " << this->speed << std::endl;

std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << std::endl;

}

double Robot::getSpeed()

{

return this->speed;

}

int fastest\_robot(Robot\* robots, const int num\_robots)

{

int max = 0;

for (int i = 0; i < num\_robots; i++)

{

if (robots[i].getSpeed() > robots[max].getSpeed())

{

max = i;

}

}

return max;

}

void conrtolRoomReport(Robot\* robots, int num\_robots)

{

std::cout << "+++++++++++++++ Working Robots +++++++++++++++" << std::endl;

for (int i = 0; i < num\_robots; i++)

{

if (robots[i].isValid())

{

robots[i].display();

}

}

std::cout << "+++++++++++++++ Broken Robots +++++++++++++++" << std::endl;

for (int i = 0; i < num\_robots; i++)

{

if (!robots[i].isValid())

{

robots[i].display();

}

}

std::cout << "\nIndex of fastest robot: " << fastest\_robot(robots, num\_robots) << std::endl;

}

Source.cpp

#include "Robot.h"

int main()

{

const int num\_robots = 5;

Robot robots[num\_robots] = {

{ "KC1", "kitchen", 25.33, 4.55, 2.2 },

{ "BR1", "bedroom", 5.22, 2.54, 2.2 },

{ "Broken", "Bedroom", 10.12, 2.5, 0},

{ "KC2", "kitchen", 20.56, 5, 3.5},

{ "BR2", "bedroom", 25.32, 6.5, 3.1}

};

conrtolRoomReport(robots, num\_robots);

}