**1,Elective Booking system**

1. Write a class called elective course which has course code, capacity, reserved.
   1. Have a member functions which displays the status of the elective course and the percentage of students registered.

The report should be in the form: "Course [id] : [registered]/[capacity] ([percentage]%) seats allotted"

**#include** <iostream>

**class** elective {

**public**:

elective(**int** code, **int** capacity, **int** reserved);

**void printStatus**();

**private**:

**int** code;

**int** capacity;

**int** reserved;

};

**void** elective::printStatus()

{

// print report here

}

elective::elective(**int** code, **int** capacity, **int** reserved)

{

// Save data to members

}

**int main**() {

**int** reserved = 0,

capacity = 0;

std::cout << "Provide elective capacity: ";

std::cin >> capacity;

std::cout << "Provide number of reserved seats: ";

std::cin >> reserved;

elective **booking**(1, capacity, reserved);

booking.printStatus();

**return** 0;

}

"Course [id] : [registered]/[capacity] ([percentage]%) seats allotted"

Example input

1

10

50

Example output

Elective 1 : 50/100 (50%) seats allotted

Example input

1

180

200

Example output

Elective 1 : 200/180 (111%) seats allotted

1. To this class do the following operations
   1. Add capacity
   2. Reduce capacity

We might also want to be able to add new capacity or reduce them. Provide a way for the user to do this via a simple interface.

The command "add [n]" will try to add n seats to the capacity of the elective.

The command "reduce [n]" will try to reduce n seats from the capacity of the elective.

If an operation fails for any reason, the program will issue the message "Cannot perform this operation"

The command "quit" will stop execution of the program.

#**include** <iostream>

**class** elective {

**public**:

elective(**int** code, **int** capacity, **int** reserved);

**void printStatus**();

**bool addSeats**(**int** number\_seats);

**bool cancelseats**(**int** number\_seats);

**private**:

**int** code;

**int** capacity;

**int** reserved;

};

// ...

elective::elective(**int** code, **int** capacity, **int** reserved)

{

// Save data to members according to limits

}

**bool** elective::addseats(**int** number\_seats)

{

// try to add reservations and return 'true' on success

// keep the limits in mind

**return** false;

}

**bool** elective::cancelseats(**int** number\_seats);

{

// try to cancel reservations and return 'true' on success

// keep the limits in mind

**return** false;

}

**int main**() {

**int** reserved = 0,

capacity = 0;

std::cout << "Provide elective capacity: ";

std::cin >> capacity;

std::cout << "Provide number of reserved seats: ";

std::cin >> reserved;

elective **booking**(1, capacity, reserved);

std::string command = "";

**while** (command != "quit")

{

booking.printStatus();

std::cout << "What would you like to do?: "

std::cin.getline(command);

// handle the command

}

**return** 0;

}

Example input

1

100

50

add 4

cancel 200

quit

User prompts were omitted in the output

Example output

Elective 1 : 50/100 (50%) seats reserved

Elective 1 : 54/100 (54%) seats reserved

Cannot perform this operation

Example input

180

200

add 1

quit

User prompts were omitted in the output

Example output

Elective 1 : 189/180 (105%) seats reserved

Cannot perform this operation