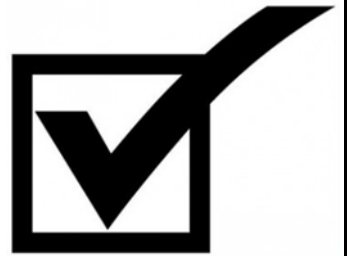


**BUSINESS
PROFESSIONALS**
of **AMERICA**
Giving Purpose to Potential



C++ Programming

(335)

REGIONAL 2025

PRODUCTION:

Speeding Ticket

_____ (530 points)

Test Time: 90 minutes

GENERAL GUIDELINES:

Failure to adhere to any of the following rules will result in disqualification:

1. Member must hand in this test booklet and all printouts if any. Failure to do so will result in disqualification.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder.
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.

*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

The graders will *not* alter your source code. Submissions that do *not* contain source code will *not* be graded.

Assumptions to make when taking this assessment:

- The goal of this assessment is to create a C++ console app that will allow the user to enter data to generate a speeding ticket and to print the ticket information to the console.
- No global variables are to be used in this project
- No numbers greater than 500 will be entered in the program

Development Standards:

- Your Code must use a consistent variable naming convention.
- All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any). Readability is a goal of good code.

Commenting for Source Code Review (see the rubric):

- Certain sections of your code will be graded. These gradable blocks of code can range from creating data structures, method algorithms, exception handling, and class construction.
- The grading rubric contains a section called Source Code Review: in this section are listed a description all the graded programming concepts.
- Each gradable item must have a comment listed at its beginning, and the comment must be prefixed with the comment flag. The flag helps the graders easily locate the code to increase the effectiveness of grading.
- The flag will always use this naming convention: **SC#** (NOTE: the # symbol will be replaced with sequential numbering, i.e. **SC1, SC2, SC3**, etc.
- No explanation in the comment with the flag is required, only the comment flag; however, any information placed in the comment could help the grader better understand and avoid any costly errors.
- The comment flag needs to be place in close proximity to the block of code it represents.
- If a comment flag is not present, you will not receive credit.
- In this example the Source Code Review has a gradable section of code for printing to the console (Remember these are non-related examples):
 - SC12: **WriteLine** method in the **Form1** class is printing the correct object _____ 10 pts
 - The user will place the code above the method call:

```
//SC12 printing a sarcastic comment  
Console.WriteLine("Get ready for some fun!");
```

C++ Regional

You will be developing a console application inside of C++. The goal of this application is to allow the user to enter in speeding ticket information. Your program will need to capture all the important data from the user as well as handle any exceptions or data entry errors. The program will then print out the speeding ticket along with the ticket cost.

Input/Output

STEP 1: When the program begins, the program will go line by line gathering data for each of the different prompts. The image below shows all the prompts for data at one time but remember the program will ask them one at a time as it steps through them.

```
Enter first name: Jose
Enter last name: Montana
Enter car make: Chevy
Enter car model: Tahoe
License plate number (1 to 8 characters): 123456
Enter speed (non-negative integer): 35
Enter speed limit (non-negative integer): 20
Is the infraction in a school zone (true or false)? true
is the infraction in a construction zone (true or false)? false
Ticket entry created.
```

STEP 2: After all the data has been entered the program will print the ticket details to the console and ask the user if they would like to enter another ticket violation.

```
Name: Jose Montana
Car Make: Chevy
Car Model: Tahoe
License Plate: 123456
Speed: 35 mph
Speed Limit: 20 mph
School Zone: true
Construction Zone: false
Ticket Cost: $600.00

Would you like to enter another ticket? (yes or no):
```

Step 3: You will enter the information again for a second violation selecting true for both school zone and construction zone infractions.

```
Would you like to enter another ticket? (yes or no): Yes
Enter first name: Mary Ann
Enter last name: Smith
Enter car make: Jeep
Enter car model: Wrangler
License plate number (1 to 8 characters): Off Road
Enter speed (non-negative integer): 30
Enter speed limit (non-negative integer): 20
Is the infraction in a school zone (true or false)? true
Is the infraction in a construction zone (true or false)? true
Ticket entry created.

Name: Mary Ann Smith
Car Make: Jeep
Car Model: Wrangler
License Plate: Off Road
Speed: 30 mph
Speed Limit: 20 mph
School Zone: true
Construction Zone: true
Ticket Cost: $1200.00

Would you like to enter another ticket? (yes or no):
```

Error Handling

Your program will be required to have the ability to handle any data entry errors. The errors will occur anytime the user is attempting to enter an incorrect value for something very specific. The scenarios that should handle data errors are listed below.

STEP 1 String Entries: the program will not allow users to enter in blank entries for non numerical data points. This will include the first name, last name, car make, car model, and license plate number. The example below only shows a blank entry for the first name but it will be the same for all of these. String entries must allow multiple word entries (allow spaces).

```
Enter first name:
First name cannot be blank. Please try again.
Enter first name:
```

STEP 2 Number Data Errors: the program will need to handle error entries for the numerical data points. This will be the speed limit and the speed of the driver. These entries need to be positive integers only. No characters, symbols, decimal place values, or negative values will be allowed. The image below shows an example of data entry errors and how the program needs to respond.

```
Enter speed (non-negative integer): 22.2
Invalid input. Please enter a valid non-negative integer for speed.
Enter speed (non-negative integer): rrrr
Invalid input. Please enter a valid non-negative integer for speed.
Enter speed (non-negative integer): %%%
Invalid input. Please enter a valid non-negative integer for speed.
Enter speed (non-negative integer): 34
Enter speed limit (non-negative integer):
```

STEP 3 Boolean Data Errors: the program will need to handle error entries for the Boolean entries. This will be the school zone and construction zone. These entries need to be “true” or “false” The image below shows an example of data entry errors and how the program needs to respond.

```
Is the infraction in a school zone (true or false)? True
Invalid input. Please enter 'true' or 'false' for school zone status.
Is the infraction in a school zone (true or false)? true
Is the infraction in a construction zone (true or false)? f
Invalid input. Please enter 'true' or 'false' for construction zone status.
Is the infraction in a construction zone (true or false)? false
Ticket entry created.
```

Output

Your program will need to print all of the ticket data on separate lines. The only data values that will be combined will be the first and last names. Each line will need to have a data label, colon, and space then the data, i.e.:

 Name: Jose Montana

Numeric data will need appropriate unit measurements, i.e.:

 Speed Limit: 20 mph

Ticket cost needs to be limited to two decimal places.

Requirements/Explanations

1. You must create a project called “**Regional_CPlusPlus**” in the C++ Regional Student folder.
2. The main function must be entered in a file named “**Regional_CPlusPlus.cpp**”.
3. Your program will need the following required variables (can use more as needed):
 - a. Strings: firstName, lastName, make, model, licensePlate, anotherTicket
 - b. Integers: speed, speedLimit
 - c. Booleans: inSchoolZone, inConstructionZone
4. Your program is required to have at least one function named getInput:
 - a. The function will not return any data and will be passed all the ticket information by reference.
 - b. All of the input for the ticket information will be validated upon input
 - i. Strings must not be blank, but should allow spaces
 - ii. Integers must be positive non-zero numbers
 - iii. Boolean values must be entered by full word in lower case (true and false)
5. Listed below are the calculations you will perform for the speeding tickets.
 - a. **Cost** = $(speed - speedLimit) * 20$
 - b. **schoolZone**: if it is true that they were in a school zone, then the Initial Cost will be doubled.
 - c. **constructionZone**: if it is true that they were in a construction zone, then the Initial Cost will be tripled.
 - d. NOTE: it is possible to get a School Zone and a Construction Zone violation.
6. Output will need to be on separate lines.
 - a. First and last names will be combined on one line
 - b. Speed and speed limit will need listed as mile per hour (MPH)
 - c. Cost will need to be displayed in US dollars with two decimal places

RUBRIC

Solution and Project (NOTE: UC represents uppercase and LC represents lowercase)		
The project file is present on the flash drive in a single folder with your contest ID		10 points
Program Execution (If the program does not execute, then the remaining items in this section receive a score of zero)		
I/O STEP 1a: When the program starts, the user is prompted for each value on separate lines. User prompts closely match those in the example.		20 points
I/O STEP 1b: "Ticket Entry Created." is display after the last value is properly entered		10 points
I/O STEP 2a: Ticket information is printed on separate lines and matches the example		10 points
I/O STEP 2b: User asked if they would like to enter another ticket (see example)		10 points
I/O Step 3: Program allows the user to enter in multiple tickets		20 points
Err STEP 1a: The program will not allow the user to enter in blank data into the fields that ask for string entries (first name, last name, license plate, car make, car model)		20 points
Err Step 1b: The user is allowed to enter multiple words into a string variable. (i.e. Mary Ann should be a valid entry for first name)		10 points
Err STEP 2a: The program does not allow the user to enter in alphanumeric characters for numerical fields (speed and speed limit)		20 points
Err STEP 2b: The program does not allow the user to enter numbers less than 1 for numerical fields (speed and speed limit).		10 points
Err STEP 2c: The program does not allow the user to enter numbers with decimals for numerical fields (speed and speed limit).		10 points
Err STEP 3: The program forces the user to enter the words "true" or "false" for Boolean fields (school and construction zones).		10 points
STEP 4: program allows the user to begin entering in a second ticket with no errors		10 points
STEP 5: user will be able to print the two speeding tickets that were entered. Format of output must match the sample given. Fields that are out of order will earn NO credit.		20 points
10mph Ticket Calculation False-False: ticket value for a 10mph speed infraction equals \$200 (both zones are false)		20 points
10mph Ticket Calculation School Zone True: ticket value for a 10mph speed infraction in a school zone equals \$400 (construction zone is false)		20 points
10mph Ticket Calculation Construction Zone True: ticket value for a 10mph speed infraction in a construction zone equals \$600 (school zone is false)		20 points
10mph Ticket Calculation Both Zones True: ticket value for a 10mph speed infraction in a construction zone equals \$1200		20 points
Subtotal		/270 Points

Source Code Review No credit will be given for missing flagged comments Code segments must work to receive full credit		
A comment containing the contestant number is present at the top of the Form1.cs file		10 points
SC1: <i>Data Input Comment Present</i>		5 points
SC1a: First Name- Validated for empty and allows spaces		10 points
SC1b: Last Name- Validated for empty and allows spaces		10 points
SC1c: Vehicle Make- Validated for empty and allows spaces		10 points
SC1d: Vehicle Model- Validated for empty and allows spaces		10 points
SC1e: License Plate- Validated for empty and allows spaces. Does not allow more than 8 characters		10 points
SC1f: Speed- Positive non-zero integer, no alphanumeric		10 points
SC1g: Speed Limit- Positive non-zero integer, no alphanumeric		10 points
SC1h: School Zone – Boolean input as full word (true or false)		10 points
SC1i: Construction Zone – Boolean input as full word (true or false)		10 points
SC2: <i>Output Comment Present</i>		5 points
SC2 a-h: Output matches example		80 points
SC2 i: Cost is displayed with only two decimal places		10 points
SC3: <i>Enter another ticket Comment Present</i>		5 points
SC3 a: Program allows multiple tickets to be entered		10 points
SC3 b: Input data is converted to lower case		20 points
Subtotal		/235 Points
Total Points		/505 Points

```
/* Student ID: for graders */

#include <iostream>
#include <string>
#include <cctype> // For isdigit
#include <iomanip> // For std::fixed and std::setprecision
#include <algorithm> // For std::transform
using namespace std;

// Function to handle input validation
void getInput(
    string& firstName,
    string& lastName,
    string& make,
    string& model,
    string& licensePlate,
    int& speed,
    int& speedLimit,
    bool& inSchoolZone,
    bool& inConstructionZone
) {
    string tempInput;
    bool validInput;
    /*      SC1      */
    /*      a-i      */
    // Validate and prompt for first name
    validInput = false;
    while (!validInput) {
        cout << "Enter first name: ";
        getline(cin, firstName);
        validInput = !firstName.empty();
        if (!validInput) {
            cout << "First name cannot be blank. Please try again." << endl;
        }
    }

    // Validate and prompt for last name
    validInput = false;
    while (!validInput) {
        cout << "Enter last name: ";
        getline(cin, lastName);
        validInput = !lastName.empty();
        if (!validInput) {
            cout << "Last name cannot be blank. Please try again." << endl;
        }
    }
}
```

```
    }

    // Validate and prompt for make of the car
    validInput = false;
    while (!validInput) {
        cout << "Enter car make: ";
        getline(cin, make);
        validInput = !make.empty();
        if (!validInput) {
            cout << "Make of the car cannot be blank. Please try again." << endl;
        }
    }

    // Validate and prompt for model of the car
    validInput = false;
    while (!validInput) {
        cout << "Enter car model: ";
        getline(cin, model);
        validInput = !model.empty();
        if (!validInput) {
            cout << "Model of the car cannot be blank. Please try again." << endl;
        }
    }

    // Validate and prompt for license plate number
    validInput = false;
    while (!validInput) {
        cout << "License plate number (1 to 8 characters): ";
        getline(cin, licensePlate);
        validInput = !licensePlate.empty() && licensePlate.length() <= 8;
        if (!validInput) {
            cout << "License plate number must be between 1 and 8 characters. Please try
again." << endl;
        }
    }

    // Validate and prompt for speed
    validInput = false;
    while (!validInput) {
        cout << "Enter speed (non-negative integer): ";
        getline(cin, tempInput);
        bool isInteger = true;
        validInput = !tempInput.empty() && !(tempInput[0] == '-' && tempInput.size() ==
1);
```

```
// Check if input is a valid integer
if (validInput) {
    for (char c : tempInput) {
        if (!isdigit(c)) {
            isInteger = false;
            break;
        }
    }
    validInput = isInteger;
}

if (validInput) {
    speed = stoi(tempInput);
    if (speed >= 0) {
        validInput = true;
    }
    else {
        validInput = false;
    }
}

if (!validInput) {
    cout << "Invalid input. Please enter a valid non-negative integer for speed." <<
endl;
}

// Validate and prompt for speed limit
validInput = false;
while (!validInput) {
    cout << "Enter speed limit (non-negative integer): ";
    getline(cin, tempInput);
    bool isInteger = true;
    validInput = !tempInput.empty() && !(tempInput[0] == '-' && tempInput.size() ==
1);

    // Check if input is a valid integer
    if (validInput) {
        for (char c : tempInput) {
            if (!isdigit(c)) {
                isInteger = false;
                break;
            }
        }
        validInput = isInteger;
    }
}
```

```
    }

    if (validInput) {
        speedLimit = stoi(tempInput);
        if (speedLimit >= 0) {
            validInput = true;
        }
        else {
            validInput = false;
        }
    }

    if (!validInput) {
        cout << "Invalid input. Please enter a valid non-negative integer for speed limit." <<
endl;
    }
}

// Validate and prompt for school zone status
validInput = false;
while (!validInput) {
    cout << "Is the infraction in a school zone (true or false)? ";
    getline(cin, tempInput);
    if (tempInput == "true") {
        inSchoolZone = true;
        validInput = true;
    }
    else if (tempInput == "false") {
        inSchoolZone = false;
        validInput = true;
    }
    else {
        cout << "Invalid input. Please enter 'true' or 'false' for school zone status." << endl;
    }
}

// Validate and prompt for construction zone status
validInput = false;
while (!validInput) {
    cout << "is the infraction in a construction zone (true or false)? ";
    getline(cin, tempInput);
    if (tempInput == "true") {
        inConstructionZone = true;
        validInput = true;
    }
}
```

```
        else if (tempInput == "false") {
            inConstructionZone = false;
            validInput = true;
        }
        else {
            cout << "Invalid input. Please enter 'true' or 'false' for construction zone status."
<< endl;
        }
    }
    cout << "Ticket entry created." << endl;
}
```

```
int main() {
    // Variables to hold user input
    string firstName, lastName, make, model, licensePlate;
    int speed, speedLimit;
    double fine = 0.0;
    bool inSchoolZone, inConstructionZone;
    string anotherTicket;

    do
    {
        // Get and validate inputs
        getInput(firstName, lastName, make, model, licensePlate, speed, speedLimit,
inSchoolZone, inConstructionZone);

        // Calculate the fine based on the inputs
        int difference = speed - speedLimit;
        if (difference > 0) {
            fine = difference * 20.0;
        }
        else {
            fine = 0.0;
        }

        // Apply multipliers for school zone and construction zone infractions
        if (inSchoolZone) {
            fine *= 2.0; // Double for school zone
        }
        if (inConstructionZone) {
            fine *= 3.0; // Triple for construction zone
        }

        /*    SC2    */
        /*    a-i    */
    }
```

```
// Display the fine and user information with two decimal points
cout << fixed << setprecision(2); // Set precision for currency formatting

cout << "\nName: " << firstName << " " << lastName << endl;
cout << "Car Make: " << make << endl;
cout << "Car Model: " << model << endl;
cout << "License Plate: " << licensePlate << endl;
cout << "Speed: " << speed << " mph" << endl;
cout << "Speed Limit: " << speedLimit << " mph" << endl;
cout << "School Zone: " << (inSchoolZone ? "true" : "false") << endl;
cout << "Construction Zone: " << (inConstructionZone ? "true" : "false") << endl;
cout << "Ticket Cost: $" << fine << endl;

/*    SC3    */
// Ask the user if they want to enter another ticket
cout << "\nWould you like to enter another ticket? (yes or no): ";
getline(cin, anotherTicket);

// Convert input to lowercase
transform(anotherTicket.begin(), anotherTicket.end(), anotherTicket.begin(),
::tolower);

} while (anotherTicket == "yes");

return 0;
}
```