14. Is the following valid code? If so, what does it print?
```
cout << (int *) "Home of the jolly bytes";
```

15. Write a code fragment that dynamically allocates a structure of the type described in Question 8 and then reads a value for the `kind` member of the structure.

16. Listing 4.6 illustrates a problem created by following numeric input with line-oriented string input. How would replacing this:
```
cin.getline(address,80);
```
with this:
```
cin >> address;
```
affect the working of this program?

17. Declare a `vector` object of 10 `string` objects and an `array` object of 10 `string` objects. Show the necessary header files and don't use `using`. Do use a `const` for the number of strings.

Programming Exercises

1. Write a C++ program that requests and displays information as shown in the following example of output:
```
What is your first name? Betty Sue
What is your last name? Yewe
What letter grade do you deserve? B
What is your age? 22
Name: Yewe, Betty Sue
Grade: C
Age: 22
```
Note that the program should be able to accept first names that comprise more than one word. Also note that the program adjusts the grade downward—that is, up one letter. Assume that the user requests an A, a B, or a C so that you don't have to worry about the gap between a D and an F.

2. Rewrite Listing 4.4, using the C++ `string` class instead of `char` arrays.

3. Write a program that asks the user to enter his or her first name and then last name, and that then constructs, stores, and displays a third string, consisting of the user's last name followed by a comma, a space, and first name. Use `char` arrays and functions from the `cstring` header file. A sample run could look like this:
```
Enter your first name: Flip
Enter your last name: Fleming
Here's the information in a single string: Fleming, Flip
```

4. Write a program that asks the user to enter his or her first name and then last name, and that then constructs, stores, and displays a third string consisting of the user's last name followed by a comma, a space, and first name. Use `string` objects and methods from the `string` header file. A sample run could look like this:

```
Enter your first name: Flip
Enter your last name: Fleming
Here's the information in a single string: Fleming, Flip
```

5. The `CandyBar` structure contains three members. The first member holds the brand name of a candy bar. The second member holds the weight (which may have a fractional part) of the candy bar, and the third member holds the number of calories (an integer value) in the candy bar. Write a program that declares such a structure and creates a `CandyBar` variable called `snack`, initializing its members to `"Mocha Munch"`, `2.3`, and `350`, respectively. The initialization should be part of the declaration for `snack`. Finally, the program should display the contents of the `snack` variable.

6. The `CandyBar` structure contains three members, as described in Programming Exercise 5. Write a program that creates an array of three `CandyBar` structures, initializes them to values of your choice, and then displays the contents of each structure.

7. William Wingate runs a pizza-analysis service. For each pizza, he needs to record the following information:
   - The name of the pizza company, which can consist of more than one word
   - The diameter of the pizza
   - The weight of the pizza

   Devise a structure that can hold this information and write a program that uses a structure variable of that type. The program should ask the user to enter each of the preceding items of information, and then the program should display that information. Use `cin` (or its methods) and `cout`.

8. Do Programming Exercise 7 but use `new` to allocate a structure instead of declaring a structure variable. Also have the program request the pizza diameter before it requests the pizza company name.

9. Do Programming Exercise 6, but instead of declaring an array of three `CandyBar` structures, use `new` to allocate the array dynamically.

10. Write a program that requests the user to enter three times for the 40-yd dash (or 40-meter, if you prefer) and then displays the times and the average. Use an `array` object to hold the data. (Use a built-in array if `array` is not available.)