C++ Primer Plus, 5^{th} *Edition* by Stephen Prata
Chapter 4: Compound Types
Review Questions

1. How would you declare each of the following?
   a. `actors` is an array of 30 `char`
   See the following code:
   ```
   char array[30];
   ```
   b. `betsie` is an array of 100 `short`
   See the following code:
   ```
   short betsie[100];
   ```
   c. `chuck` is an array of 13 `float`
   See the following code:
   ```
   float chuck[13];
   ```
   d. `dipsea` is an array of 64 `long double`
   See the following code:
   ```
   long double dipsea[64];
   ```

2. Declare an array of five `ints` and initialize it to the first five odd positive integers.
   See the following code:
   ```
   int sea[5] = {1,3,5,7,9};
   ```

3. Write a statement that assigns the sum of the first and last elements of the array in Question 2 to the variable `even`.
   See the following code:
   ```
   int even = sea[0] + sea[4];
   ```

4. Write a statement that displays the value of the second element in the `float` array `ideas`.
   See the following code:
   ```
   cout << ideas[1] << endl;
   ```

5. Declare an array of `char` and initialize it to the string `"cheeseburger"`.
   See the following code:
   ```
   char land[] = "cheeseburger";
   ```

6. Devise a structure declaration that describes a fish. The structure should include the kind, the weight in whole ounces, and the length in fractional inches.
   See the following code:
   ```
   struct fish
   {
       char kind[20];
       int weight;
       double length;
   };
   ```

7. Declare a variable of the type defined in Question 6 and initialize it.
See the following code:

```
struct fish goldfish = {"goldfish", 1, 1.5};
```

8. Use `enum` to define a type called `Response` with the possible values `Yes`, `No`, and `Maybe`. `Yes` should be `1`, `No` should be `0`, and `Maybe` should be `2`.
See the following code:

```
enum response {No, Yes, Maybe};
```

9. Suppose `ted` is a `double` variable. Declare a pointer that points to `ted` and use the pointer to display `ted`'s value.
See the following code:

```
double * p = &ted;
cout << *p << endl;
```

10. Suppose `treacle` is an array of 10 `floats`. Declare a pointer that points to the first element of `treacle` and use the pointer to display the first and last elements of the array.
See the following code:

```
float * f = treacle;
cout << "first element: " << *f << endl
     << "second element: " << *(f + 9) << endl;
```

11. Write a code fragment that asks the user to enter a positive integer and then creates a dynamic array of that many `ints`.
See the following code:

```
cout << "enter a positive integer: ";
int n;
cin >> n;
int fuzzy[n];
```

12. Is the following valid code? If so, what does it print?

```
cout << (int *) "Home of the jolly bytes";
```

Yes, this is valid code. When you feed a string into the `cout` *object, you are actually giving* `cout` *the memory address of the first character in the string. Since the type cast works for a pointer which points to the address of the first character, it must also work in the same way as the address of the first character. The code prints the memory location of the first character in the string.*

13. Write a code fragment that dynamically allocates a structure of the type described in Question 6 and then reads a value for the kind member of the structure.
See the following code:

```
fish * pt = new fish;
cout << "type of fish? ";
cin.get(pt->kind, 19);
```

14. 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?

    `cin` only accepts input after the return key is hit and reads the first token of data available. Thus, whatever the user typed, only the first word (or token) would be stored as the address. Additionally, there would be no error checking if the input was over 79 characters.