Homework

10

A read Procedure

The Pascal programming language used its predefined procedures **Read** and **Write** to do input and output. Even though there's probably not much demand for translating old Pascal code into C++, overloading, default arguments, and reference parameters make writing your own take on Pascal's **Read** procedure pretty easy.

Here are the several overloaded versions of the **read** function to write:

```
1. if (read("How old are you? ", age)) ...
2. if (read("What is your gpa? ", gpa, true)) ...
3. if (read("What is your full name? ", name, true)) ...
4. while (read(ch, '.')) cout << ch; ...
```

1. Prompts and reads into the variable age. If there are no read errors, it returns **true**. If there are errors, it consumes the offending token, clears the input stream and returns **false**. Additional input remains in the stream.
2. Exactly as the previous example, except any additional input on the line after the input value is consumed and discarded, but only if the input was successful.
3. If the variable is a **string**, as here, then the optional **bool** parameter means to use **getline()** rather than formatted input. It should default to false. Return the value of the that parameter.
4. This version reads one character into **ch**. If it matches the sentinel argument, return **false**, otherwise, return **true**.

A Few Hints

- Versions 1, 2 and 3 use **default arguments** for the optional last parameter.
- For version 1 and 2, the parameter indicates **whether the entire line** should be consumed or only the specific input. For version 3, the parameter indicates whether to use **`getline()`** or **`>>`.**
- Versions 1 and 2 (**`int`** and **`double`**) return **`false`** if the input is not well formed. (Check with **`cin.fail()`**). For the returned value, return **`false`** if input fails, **`true`** otherwise.
- For version 3 accepts **`string`** variables (whose input cannot fail). Return the value of the last parameter.
- Version 4 (which only works with **`char`** variables) and does not have a default parameter, not a prompt, returns **`false`** when it reads **the sentinel**. Read all spaces using **`cin.get()`** or with **`noskipws`** and **`>>`**.
- Consume additional end-of-line input using **`cin.ignore(1024, '\n');`**
- Reset the input flags by using **`cin.clear();`** Consume an invalid token using **`cin >> junk`** where **`junk`** is a **`string`**.

Use **`make test`** to test your code, **`make stest`** or **`make run`** to run any student tests. Once your score is OK, use **`make submit`** to turn it in.

If you get stuck, ask for help on Piazza, or come by my office hours (early!!!).