Homework

2

Functions at Work

Write a program that translates a letter grade into a numeric grade point. Instead of writing a monolithic program, use functions to solve the problem.

Here are the details:

> *Letter grades are **A**, **B**, **C**, **D**, and **F**, possibly followed by **+** or **–**. The numeric values are **4**, **3**, **2**, **1**, and **0**. In this scheme, there is no **F+**, **F–**, or **A+**. Adding a + to the grade increases the value by **0.3**; – decreases it by **0.3**.*

The `run()` function is already completed. **Do not change it**. Before the `run()` function, you'll find the **prototypes** for the functions called, as well as two **constants** that your program will use.

Writing the Stubs

Your job is to **implement** the functions that the program uses. Start by **writing the stubs**. A stub is a function "skeleton" that is syntactically correct, and which stands in for the finished function, so that the entire program can be compiled and run. You'll complete the mechanical details first, and worry about the logic of each function later.

1. **Copy** all of the prototypes from before `run()` to the bottom of the file.
1. Remove all the semicolons, and replace with a body `{ }`.
2. If the function return type is other than `void`:
   a. Create a `result` variable of the return type, **default initialized**
   b. Add a `return` statement that returns `result`

In the terminal, type `make` and press `ENTER`. Your program should compile and link with no errors. If it does not, then please post a message on Canvas and get help.

The *printTitle* Procedure

Here is what the program should look like when it runs:

```
sgilbert-H03-Grade Calculator
----------------------------------------
Enter your letter grade: C+
Grade value is [2.3]
```

The first two lines are produced by:

```
void printTitle();
```

The return type is **void**, so it is **a procedure**: it carries out an action. Nothing appears in the parameter list, so you need not pass any arguments.

Use **cout** along with the variables **STUDENT** and **ASSIGNMENT** to print the two lines inside **printTitle()**. Then, type **make run**, press **ENTER**, and make sure the output matches that shown here. Next, turn your attention to **getInput()**, which returns a **string**.

Implementing *getInput()*

Look at the program mockup in the previous section. The third line of output is produced by the function call on line 32 in **h03.cpp**.

```
string letterGrade = getInput();
```

Complete the **getInput()** function like this:

1. **Print the prompt** as shown in the mockup on the first line of your function. Remember to **leave a space after the colon** in the prompt. Do not add a newline after the prompt.
2. Call the **getline()** function to initialize the **result** variable. You can find more information on the **getline()** function in the Course Reader.
3. Return **result** as before.

Implementing letterToPoints()

The **`letterToPoints()`** function takes a **`stringIn`** (**`"A"`**) as input, returning the grade points (**`4.0`**) as a **`double`**. It may also return two "error" codes if the input is invalid: the constants **`INVALID_INPUT`** and **`INVALID_COMBINATION`**.

Processing (the second step in IPO) means turning input into output. The input is the **`letterGrade`** parameter, and the output is the **`result`** local variable.

1. Because strings can be compared using the **relational operators** in C++, use the condition **`letterGrade == "A"`**. When true, set the result to **`4.0`**.
2. After you've checked **`"A"`**, you want to check **`"A-"`**. If you already found the answer in Step 1, there's no reason to check again, which means you should use **sequential `if` statements** for the subsequent tests.
3. After you've processed all valid combinations, you should check for the three invalid ones: **`"A+"`**, **`"F+"`** and **`"F-"`**. Use the logical **`OR`** operator in your condition. This test should also be part of your sequence.
4. At the end of the sequence, use an **`else`** to set **`result`** to **`INVALID_INPUT`**.

We don't yet have the output part done, but you should use **`make`** and **`make run`** to catch any syntax errors at this point.

Implementing *printReport()*

The final, output function is **`printReport()`** which takes a **`double`** representing the grade points, and then prints the final line of output, like this:

```
Grade value is [2.3]
```

Here's what you should do:

1. Use **`cout`** to print the **literal text**: "`Grade value is [`".
2. If the **`points`** are **`0`** or greater, print them out, using one decimal point.
3. If the points are **`INVALID_INPUT`**, then print "`Invalid input`".
4. If the points are **`INVALID_COMBINATION`**, print "`Invalid combination`"
5. Finish by printing the "**`]`**" and a newline.

Use **`make test`** to check your work. If you have errors, correct them and check again. Make sure you use **`make submit`** to turn in the homework before the deadline.