CS 135 - Computer Science I : Spring 2023
Programming Assignment 02a (`pa02a.cpp`)

Limitations

Please refrain from using advanced C++ code techniques in this assignment. Limit commands to those discussed in the current and previous chapters, labs, and lectures:

- **C++ Command Limit: Do not use commands beyond Chapter 02.**

Students are NOT allowed to collaborate on this programming assignment. However, you can ask a teaching assistant or tutor for help. Do not share code. If you get help or discuss this assignment with another instructor, include their name(s) in the header comments.

Unless instructed to do so, do not use the following C++ commands:

- `exit()`
- `continue`
- `break` (unless it's in a `switch` statement)
- `goto`

All functions must have only ONE `return` statement, which must be located at the end of the function. This rule includes `main()`.
(see https://dbrodersen.net/codeRubric_cs135 for a submission checklist.)

Your code must adhere to the course style guide to get credit for this programming assignment.
(see https://dbrodersen.net/comments for code formatting standards and how to comment your code.)

- Your output must match the provided sample interactions, if any.
- Never use TAB characters in source code or program output.

Please submit questions regarding this assignment at least 24 to 48 hours before it is due to allow time for a response that you can implement.

Programming Assignment: Students may NOT collaborate with each other

To complete this assignment you must first...

- Read the text through Chapter 02 – up to Type Conversion, and completed the lab assignment.
- Complete the previous chapter labs and programming assignments
- Study the sample solutions for the previous assignments

The due date for this assignment is posted in Canvas.

Section: 1 Problem Description

You are considering adding solar renewable energy to your home through the use of roof-mounted solar panels. You want to determine the surface area required for the minimum number of solar panels needed to determine if they can fit on your roof!

Write a program that prompts the user to input the annual power consumption for their home in kilowatt-hours (kWh). Compute the number of solar panels required and their total surface area in square feet. Display the minimum number of panels required and the solar panel surface area in square feet. See Sample Interaction below.

> Given: The average output of a solar panel is 300 watts (W) per hour, a typical panel is 1.6m long by 1m wide, and there are 2.54 cm in one inch.

Input	Outputs
Annual energy consumption (kWh)	• Number of panels required • Total area of solar panels (ft^2)

Programming Assignment: Students may NOT collaborate with each other

Section: 2 Algorithm

1. **Determine your annual electric consumption:** The first step is to determine your home's annual electric consumption in kilowatt-hours (kWh). You can find this information on your energy bills or by using a smart meter.
2. **Calculate the number of solar panels needed:** Once you know your annual electric consumption, you can use the following rule of thumb formula to approximate the number of solar panels needed:

$$\textit{Number of solar panels} = \frac{\textit{annual electric consumption (kWh)}}{\textit{average output of a solar panel (Watts)}}$$

   For example, if your annual electric consumption is 10,000 kWh and the average output of a solar panel is 300 watts per hour, you would need approximately 33.3 solar panels (10,000 / 300 = 33.3), or 33 whole panels.
3. **Calculate the surface area of the solar panels:** The surface area of the solar panel is the total area that is covered by the solar cells. The surface area for one panel can be computed from its dimensions. Once you know the number of solar panels needed and the surface area of each panel, you can determine the total surface area of the solar panels needed for your home.

Section: 3 Input specification

The only input to this program is the annual energy consumption in kWh, which may have decimal digits.

Section: 4 Output specification

Your program outputs the whole number of solar panels required and the total solar panel surface area required in square feet. Your output formatting should match the sample interaction shown below.

Programming Assignment: Students may NOT collaborate with each other

Section: 5 Sample Interaction

Here is a sample interaction for your program once it is compiled and running correctly.

Text in red is command invocation. Highlighted blue text represents user input:

Sample Interaction

```
$ ./pa02a
Please enter your annual power consumption in kWh: 10000.0

Number of solar panels required: 33
Solar panel surface area required: 568.334 square feet
$
```

Helpful commands for this assignment

Bellagio Linux Console (Terminal)

```
$ mkdir -p ~/cs135/pa02a/
$ cd ~/cs135/pa02a/
$ nano pa02a.cpp
$ geany pa02a.cpp
$ cpplint pa02a.cpp
$ pclint pa02a.cpp
$ cppcheck pa02a.cpp
$ aspell -c pa02a.cpp
$ astyle pa02a.cpp
$ g++ $CXXFLAGS pa02a.cpp -o pa02a
$ ./pa02a
$ man linux command
$ cppman CPP command
```

Where:

`mkdir -p`
: creates a directory and its path

`nano`
: ASCII text editor available on Bellagio for editing program files

`geany`
: IDE for working on programs (works best with x2go)

`cpplint`
: Open Source Linter (static code analyzer) by Google

`pclint`
: Commercial Linter by Gimp Suit Software

`cppcheck --language=c++ --std=c++14`
: Performs a static check for bugs and undefined behavior

`aspell -c`
: spell checker

`astyle`
: artistic style code indenter / formatter / beautifier

`g++ $CXXFLAGS`
: C++ compiler. The option **-o** will allow you to name the executable file

`man` Immediate online help on a Linux command

`cppman`
: Immediate online help on a C++ command

Programming Assignment: Students may NOT collaborate with each other

Turn in pa02a

To receive credit for your work:

- Files must be correctly named
- Files must be correctly submitted before the due date and time
- The program must compile on the bellagio server.
- The program must not crash when launched or while running
- A correctly formatted header comment must be included (see: https://dbrodersen.net/comments)
- Code must be properly commented (see: https://dbrodersen.net/comments)
- Code must be properly formatted, such as proper indentation, etc (see video examples and sample solutions)
- TABs must not be used in the source code or output of the program

Note: *It's really important that you implement your assignments exactly as written. No extra bells and whistles, please. I've written a shell script that:*

- *uses cpplint and pclint along with custom commands that go through the mechanics of your code to make sure syntax, variable names, functions, procedures, and spelling are all correct.*
- *runs the assignment using the sample data exemplified in the assignment description and compares your program screen and file output with the expected output*
- *runs the assignment a second time using data other than that provided in the sample solution*

In this way, I am able to determine if the logic and calculations in your program are correct. Lastly, I look at the code myself, scanning for code styling, proper commenting, and efficiency.

When you include extra features, the script flags them and deducts points.

Programming Assignment: Students may NOT collaborate with each other

Labs and programming assignments are NOT turned in using Canvas. Use the command below while logged into *Bellagio* to turn in your assignment. Cut and paste will not work with this command. You must type it in yourself. Note: you must be in the directory where pa02a.cpp exists for this command to work.

Bellagio Linux Console (Terminal)

```
$ turnin -c cs135-dbrodersen -p pa02a -v pa02a.cpp

— results of a successful submission —
Submitted files:
pa02a-username
pa02a-username/pa02a.cpp
$
```

The Bellagio **turnin** command submits assignments to be graded. The drop box for the assignment closes promptly at the due date/time. If the project is enabled and you want to resubmit your assignment, you may do so by re-running the `turnin` command.
Note: New submissions overwrite any previous submissions.

Syntax: `turnin -c` *course-name* `-p` *project-id* `-v` *file name(s)* separated by spaces.

Option	Purpose
`-c`	**Course**. Sets the submission course to which we'll submit our assignments. *Required.*
`-p`	**Project**. Sets the project to which we'll submit our assignments. The project id is the name of the file without the extension.
`-v`	**Verbose**. Prints a list of submitted files once they have been submitted.
`-h`	**Help**. Print a help message.
`-l`	**List**. Prints a list of projects, along with whether or not they are enabled and shows which project is the default project.

Programming Assignment: Students may NOT collaborate with each other