CISC 5300—Programming in C++
Fall, 2018

Programming Project # 1: Counting Coins
Due Date: Wednesday 19 September 2018

Write a program that prompts the user to enter some number of

- pennies (1-cent coins)
- nickels (5-cent coins)
- dimes (10-cent coins)
- quarters (25-cent coins)
- half dollars (50-cent coins)

Query the user separately for the number of each size coin. Your program should then print out the total value of the coins (in cents)

Here are a few sample runs:

```
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 23
  Nickels? 17
  Dimes? 14
  Quarters? 7
  Half dollars? 3
The value of all your coins is 573 cents.
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 1
  Nickels? 2
  Dimes? 0
  Quarters? 3
  Half dollars? 2
The value of all your coins is 186 cents.
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 5
  Nickels? 1
  Dimes? 3
  Quarters? 2
  Half dollars? 1
The value of all your coins is 140 cents.
```

```
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 5
  Nickels? 0
  Dimes? 0
  Quarters? 0
  Half dollars? 0
The value of all your coins is 5 cents.
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 5
  Nickels? 0
  Dimes? 0
  Quarters? 0
  Half dollars? 2
The value of all your coins is 105 cents.
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 0
  Nickels? 0
  Dimes? 0
  Quarters? 0
  Half dollars? 0
The value of all your coins is 0 cents.
erdos@dsm:proj1$ proj1
Enter the number of coins you have for each denomination.
  Pennies? 23
  Nickels? 17
  Dimes? 17
  Quarters? 7
  Half dollars? 3
The value of all your coins is 603 cents.
```

A few considerations and hints:

1. You are to do this using only the techniques found in Chapters 1 through 3 of the text. In other words, you don't need to read ahead to figure out how to do this. There *are* less verbose ways to solve this problem using a loop and vectors, but we're not ready for that yet.

2. I have made an executable version of this program for you to try out. The executable is available as the file

   `~agw/class/programming-c++/share/proj1/proj1-agw`

   on the Departmental Linux machines. This means you can execute it by logging in on one of these machines, doing the shell command

```
cd ~agw/class/programming-c++/share/proj1
```

to put you into the proper directory, after which you can run the shell command `proj1-agw` as many times as necessary. For that matter, you can copy the `proj1-agw` executable into your working directory, and run it from there.

3. Use (appropriately-named) variables to store the number of half-dollars, quarters, dimes, nickels, and pennies.

4. You should do this project, in a subdirectory `proj1` of your `~/private/programming-c++` directory. You can create this via the shell command

   `mkdir␣~/private/programming-c++/proj1`

   (You shouldn't need the `-p` flag, since you previously created `~/private/programming-c++` when you did Project 0.) To guarantee that any files you create for this project (e.g., the C++ source code file `proj1.cc` and the executable program `proj1`) live in this directory, you will need to issue the shell command

   `cd␣~/private/programming-c++/proj1`

   before you start creating the file `proj1.cc`, i.e., before you run "`emacs proj1.cc`".

5. Remember to include the standard libraries by putting

   ```
   #include <bjarne/std_lib_facilities.h>
   ```

   near the beginning of your program. *Unless stated otherwise, this will be standard operating procedure from now on.*

6. You should test your program with the seven sets of input values shown above. If you get the same answers as seen above, your program is good enough. For example, you needn't worry about erroneous input data (such as a negative number of coins).

7. As with Project 0, use the `photo` program so you can have something to turn in. *Don't do this until the program is working and you're ready to submit your solution!* The commands you should issue are as follows:

   ```
   photo
   cat proj1.cc
   g++ -std=c++17 -o proj1 proj1.cc
   proj1
   proj1
   proj1
   proj1
   proj1
   proj1
   proj1
   exit
   ```

The input values you should use when running `proj1` are the same as were shown above. Note that you need to run `proj1` seven times, once for each data set. *Do **not** use any other sets of input values.*

This will create a file, named `typescript`, in the current directory, containing a listing of `proj1.cc`, evidence that it compiles with no errors or warnings, and a sample execution. This `typescript` file may have "funny" characters, some of which may be removed by issuing the shell command

```
photo-clean typescript > proj1.out
```

You should make sure that your file `proj1.out` really contains what you think it contains. (It might be empty, or it might contain out-of-date material.) You can do this by looking at the file (via the shell command "`more proj1.out`") or by using the shell command "`ls -lt`" to ensure that `proj1.out` is nonempty and that it is newer than the `typescript` file before proceeding further. You may need to delete a pre-existing `proj1.out` file should this not be the case.

You can now turn in `proj1.out` in one of two ways:

(a) You can simply send me an email, letting me know that you're done with the project, and that I can get the clean typescript file myself. For this to work, your file must be named[1]

```
~/private/programming-c++/proj1/proj1.out
```

or I won't be able to find it. If you follow the instructions given above (i.e., make sure that you're in the `~/private/programming-c++/proj1` directory before you start your work and that the clean typescript is named `proj1.out` within that directory), then this will happen automatically.

(b) You can email it to me, by issuing the shell command

```
mail -s "Project 1" agw < proj1.out
```

Good luck!

[1] As always, the tilde denotes your home directory.