

1 Presentation

Due Date: February 3, 2022 no later than 11:59pm.

Purpose: This first project is designed to teach you how to devise, implement, and submit solutions to programming problems. As you will be expected to complete projects not only this semester, but throughout your studies, one of our primary goal is to familiarize you with the submission process. This task will also ensure that you understand the basic concepts we have been studying thus far.

Skills: For this project, you will need:

- To exhibit an understanding of basic IDE features,
- To analyze a simple problem and determine how a program could solve it,
- To implement your solution,
- To understand how to meaningfully test and debug your program.

Knowledge: This assignment will familiarize you with the following important skills:

- How to share a program,
- How to interpret a series of basic instructions,
- How to convert between units of measures using a program.

2 Tasks

2.1 Challenge

In short: In this project, you are asked to create a program that asks the user to enter an amount of money they wish to withdraw, and then displays an assortment of banknotes corresponding to that amount.

In more details: In your IDE, create a project named “Project01”. Write your name and the date in a delimited comment *at the very beginning* of your code.

In the Main method:

- Declare a variable and store your Augusta University username in it.
- Then, write a statement that will make your program display on the screen the content of that variable, followed by “ would like to know how much you want to withdraw:”.
- Then, write a statement that will store the value entered by the user. The value should be stored in a data type that supports high precision, since we are dealing with currency. (It is OK if the program crashes if the user doesn’t enter a number at the prompt).
- Then, write statements (more than one can be needed) to split the user-provided amount of money into an equivalent value in \$50, \$20, \$10, \$5 and \$1 banknotes. Your program should prioritize using the fewest total number of banknotes, i.e. do not display 10 \$1 bank notes instead of 1 \$10 banknote.
- Finally, write a statement that will display “This corresponds to”, followed by the results of your computations.

Example: Assuming your username is “jdoe”, an execution could give something *like*:

```
jdoe would like to know how much you want to withdraw:
150 ↵
This corresponds to 3 $50 banknotes, 0 $20 banknotes, 0 $10 banknotes, 0 $5
↵ banknotes, and 0 $1 banknotes.
```

Note that “150” was entered by the user, *not* by the programmer: your program should work with any monetary value! Also note that the program displays “0 \$20 banknotes” (with an “s”) because we do not know in advance how many banknotes of each type will result from the conversion. Another example could be:

```
jdoe would like to know how much you want to withdraw:
36_ ←_
This corresponds to 0 $50 banknotes, 1 $20 banknotes, 1 $10 banknotes, 2 $5
↪ banknotes, and 1 $1 banknotes.
```

Bonuses: Extra points will be given if your program

- Can handle decimal values, and additionally displays information about the remainder,
- Uses format specifiers to display more nicely the currency,
- Displays the information more nicely.

Taken together, those improvements would result in an execution *like*:

```
jdoe would like to know how much you want to withdraw:
45.91 ←_
This corresponds to
- 0 $50.00 banknotes,
- 2 $20.00 banknotes,
- 0 $10.00 banknotes,
- 1 $5.00 banknotes,
- 0 $1.00 banknotes,
and the remainder is 91 cents.
```

2.2 Submission

Once your project is done, zip the folder into a file “lname_fname.zip”, where “lname” is your last name and “fname” is your first name. Upload this zip on D2L before Thursday, February 3, at no later than 11:59 PM in the “Project 1” assignment submission folder.

3 Additional Considerations

3.1 Criteria for Success

A good, finished project, should include *all* of the following:

- Something was submitted, on D2L
- The archive can be downloaded, extracted, and the solution in it can be opened using any IDE
- The archive and solution have the correct names
- Your name and the date are in a delimited comment at the beginning of the file
- The program compile and can be executed
- All the variables have appropriate types, and are set to the correct value, or can be set to expected values by the user
- The messages are correctly displayed
- The conversion is correct, including with possible rounding or truncating
- Your code is commented.

3.2 Advice and Recommendations

- Read the problem statement over and over; make sure you did not overlook anything.
- Make sure your project compiles without errors or warnings and can be executed as expected. Errors if the user gives bad input are fine!
- A partially completed project is better than nothing. If the user input doesn't work, or if the conversion is a bit off, add a comment describing your difficulties: that will show that you are aware of the limitations of your program.

- Make sure you submitted the right files by re-downloading them (possibly on a different computer) and making sure you can still compile and execute your program.
- *Do not* ask other classmates, the undergraduate assistants, or tutors for help. This work is supposed to be *your own*, and should reflect *your own* understanding of the previous labs. Copying-and-pasting code from the internet will hurt your grade and your understanding of this class. Any cheating will be easily detected and punished according to the documents mentioned in the syllabus.
- You can ask me for help and feedback. Please upload your project according to the instructions, and send me an email asking to go over it if you want me to check that you submitted the right file, that your code compiles, etc. You should ask early, and not wait until the last minute.
- No project help will be given during lab: I want to make sure I can assist the students working on the current lab.