///////////////////////////////////////////////////////////////////////////

Reflection

Workshop - #2

Full Name : MOHAMED ASHRAF BHAROT

Student ID#: 139539225

Email : mamabharot@myseneca.ca

Section : ZEE

Authenticity Declaration:

I declare this submission is the result of my own work and has not been.

shared with any other student or 3rd party content provider. This submitted

piece of work is entirely of my own creation.

///////////////////////////////////////////////////////////////////////////

1. Given the following C statements:

float price1 = 17.96f, price3 = 17.98f;

double price2 = 17.96, price4 = 17.98;

printf("1. 17.96 = %f (as cents:%d)\n", price1, (int)(price1 \* 100));

printf("2. 17.96 = %lf (as cents:%d)\n", price2, (int)(price2 \* 100));

printf("3. 17.98 = %f (as cents:%d)\n", price3, (int)(price3 \* 100));

printf("4. 17.98 = %lf (as cents:%d)\n", price4, (int)(price4 \* 100));

Execute the above sample code on your machine. Briefly explain why price1 and the corresponding converted cent value does not properly represent the intended value. Why do you think all the other samples work as expected?

* The result of price 1 and the corresponding converted cent value does not represent the intended value like other samples work that is because float type has been declared in price1.Float provides limited precision value as compared to double data type and also when multiplies by 100 it gives an integral value. So, price 1 does not provides intended value. While the others price2, price3 and price4 are declared as double data type, because of which it gives a more precised value.

2. Execute the below code on your machine:

int money = 7245;

printf("1. $%.2lf\n", money / 100.0);

printf("2. $%.2lf\n", money / (double)100);

printf("3. $%.2lf\n", (double)money / 100);

Explain why these three examples work as intended. Apply terms like **cast**, **promotion**, **coercion**, **implicit**, and **explicit** in your answer.

* In the first example there has been a promotion of data type to double from integer by using 100.0. Because of which the division operation is performed which results in double type, so it works as intended.
* In the second example the result is accurate because of using casting which converts one data type into another. By using the double cast operator, we intend to get this result with the help of explicit casting.
* The third example also works on the same principal of type casting as in second example. It converts the integer data type to double to give intended result.

3. Execute the following code on your machine:

int result, money = 7245;

result = money / 100;

printf("1. $%.2lf\n", money / 100);

printf("2. $%d\n", money / 100);

printf("3. $%.2lf\n", result);

printf("4. $%d\n", result);

Apply terms like **cast**, **promotion**, **truncation**, **coercion**, **implicit**, and **explicit** in answering

following questions

1. a ) What is the value of the variable **result**? Explain why.
2. b) What values are displayed for samples 1, 2, 3, and 4?
3. c) What can you conclude about the printf function that causes samples 1 and 3 not to work as
4. expected?

* a.) The value of the variable result will be 72. The division operation in the line result = money/100, is normal division between two integers because of which it will discard the fractional part of the result and thus, the result will contain truncated value.
* b.) The values displayed for samples 1,2,3 and 4 are
  + For sample 1: $ 0.00
  + For sample 2: $72
  + For sample 3: $0.00
  + For sample 4: $72

c.) The format specifier used in sample 1 & sample 3 is %.2lf because of which printf expects a double type of argument. However, the actual argument is an integer. In this case, there is an implicit conversion from integer to double which leads to an unexpected result.