**Greedy**

[**tl;dr**](http://docs.cs50.net/problems/greedy/greedy.html#tl-dr)

Implementa un programa que calcule el mínimo número de monedas requeridas para dar el cambio a un usuario.

$ ./greedy

O hai! How much change is owed?

0.41

4

[**Background**](http://docs.cs50.net/problems/greedy/greedy.html#background)

Cuando se utiliza un aparato como este, las probabilidades son de que se quiera minimizar el número de monedas que se dispensan a cada comprador, para no tener que presionar las palancas más veces que las necesarias. Afortunadamente, las ciencias de la computación han dado a los cajeros de todas partes, formas de minimizar el número de monedas a través de: greedy algorithms.

De acuerdo al National Institute of Standards and Technology (NIST), un algoritmo greedy es uno que "siempre toma la solución mejor, inmediata, o local al encontrar una respuesta. Los Algoritmos Greedy buscan la solución más general o globalmente óptima para algunos problemas de optimización, pero podrían encontrar una solución menos que óptima para instancias de otros problemas."

¿Qué significa todo eso? Bien, supongamos que un cajero le debe cambio a un cliente y en ese cinturón de cajero hay palancas que dispensan quarters, dimes, nickels, y pennies. Resolver este “problema” requiere presionar una o más veces una o más palancas. Piensa en un cajero "greedy" como uno que quiere sacar con cada presión, el pedazo más grande posible de este problema. Por ejemplo, si a un comprador se le deben 41¢, el más grande primero (i.e., mejor immediata, o local) pedazo que puede ser tomado es 25¢. (Ese pedazo es el "mejor" en tanto nos acerca a 0¢ más rápido de lo que cualquier otra moneda lo haría.) Note que un pedazo de ese tamaño corta lo que era un problema de 41¢ hasta ser un problema de solo 16¢ ya que 41 - 25 = 16. Ese es el remanente, un problema similar pero más pequeño. No es necesario mencionar que otro pedazo de 25¢ sería muy grande (asumiéndo que el cajero prefiere no perder dinero), y así nuestro cajero greedy se moverá a sacar un pedazo de tamaño de 10¢, dejándolo con un problema de 6¢. A este punto, greed llama por un pedazo de 5¢ seguido de un pedazo de 1¢, y en ese punto el problema está resuelto. El cliente recibe un quarter, un dime, un nickel, y un penny: cuatro monedas en total.

Resulta que este enfoque greedy (i.e., algoritmo) no es solo localmente óptimo sino también globalmente óptimo para la moneda de EEUU (y también para la Unión Europea). Es decir, siempre y cuando un cajero tenga suficientes monedas esta aproximación de mayor a menor siempre cederá la menor cantidad de monedas posibles. ¿Que tan pocas? ¡Bueno, tu dínoslo!

[**Especificación**](http://docs.cs50.net/problems/greedy/greedy.html#specification)

* Escriba, en un archivo llamado greedy.c en tu directorio ~/workspace/pset1/ un programa que primero pregunte al usuario cuánto cambio se debe y entonces regrese el mínimo de monedas con las que el cambio dicho se pueda hacer.
* Use get\_float de la Librería CS50 para obtener la entrada del usuario y printf de la librería Standard I/O para sacar su respuesta. Asuma que las únicas monedas disponibles son: quarters (25¢), dimes (10¢), nickels (5¢), y pennies (1¢).
  + Le pedimos que utilice get\_float para poder manejar dólares y centavos, aunque sin signo de dólar. En otras palabras, si aun cliente se le deben $9.75 (como sería l caso dónde un periódico cuesta 25¢ pero el comprador paga con un billete de $10), asuma que la entrada de su programa será 9.75 y no $9.75 o 975. Sin embargo, si a un cliente se le debe $9 exactamente, asuma que la entrada de su programa será 9.00 o solamente 9 pero, de nuevo, no $9 o 900. Por supuesto por la naturaleza de los valores de punto flotante, su programa probablemente funcionará con entradas como 9.0 así como 9.000 ; usted no necesuta preocuparse de revisar si la entrada del usuario está "formateada" como el dinero debería ser.
* Usted no necesita intentar revisar si la entrada de un usuario es muy larga para caber en un float. Usando solamente get\_float se asegurará de que la entrada del usuario sea de hecho un valor de punto flotante pero no que sea un no negativo.
* Si el usuario falla al proveer un valor no negativo, su programa deberá re solicitor al usuario un monto válido una y otra vez hasta que el usuario cumpla.
* Incidentalmente, como Podemos automatizar algunas pruebas para su Código, le pedimos que la última línea de salida de su programa sea solo el mínimo número de monedas posible: un integer seguido de \n.

[**Walkthrough**](http://docs.cs50.net/problems/greedy/greedy.html#walkthrough)

[**Us**](http://docs.cs50.net/problems/greedy/greedy.html#usage)**o**

Su programa deberá comportarse igual que el ejemplo de abajo. Asumiéndo que el texto subrayado es lo que el usuario ha escrito.

$ ./greedy

O hai! How much change is owed?

0.41

4

$ ./greedy

O hai! How much change is owed?

-0.41

How much change is owed?

-0.41

How much change is owed?

foo

Retry: 0.41

4

[**Testing**](http://docs.cs50.net/problems/greedy/greedy.html#testing)

[**check50**](http://docs.cs50.net/problems/greedy/greedy.html#code-check50-code)

check50 cs50/2017/x/greedy

[**Staff Solution**](http://docs.cs50.net/problems/greedy/greedy.html#staff-solution)

~cs50/pset1/greedy

[**Sugerencias**](http://docs.cs50.net/problems/greedy/greedy.html#hints)

* Por el punto final de arriba en la especificación, no olvide poner un character de nueva linea al final de la impresión de salida!
* Tenga cuidado con la inherente imprecisión de los valores de punto flotante. Por ejemplo, 0.1 no puede ser representado exactamente como un float. Intente imprimiendo su valor a, digamos 55 lugares decimales, con código como el siguiente:
* float f = 0.1;

printf("%.55f\n", f);

Y entonces, antes de hacer cualquier cambio, usted probablemente quiera convertir la entrada del usuario enteramente a centavos (i.e., de un float a int) para evitar pequeños errores que de lo contrario podrían sumarse! Por supuesto no solo pases la entrada delusuario de un float a un int! Después de todo, a cuantos centavos equivale un dólar?

* Y tenga cuidado de no redondear sus centavos (al más cercano penny); no "truncque" sus centavos!