



< Previous	✓	✓	✓	✓	✓	Next >
------------	---	---	---	---	---	--------

Problem 2

Bookmark this page

Problem 2 - Paying Debt Off in a Year

15.0/15.0 points (graded)

Now write a program that calculates the minimum **fixed** monthly payment needed in order pay off a credit card balance within 12 months. By a fixed monthly payment, we mean a single number which does not change each month, but instead is a constant amount that will be paid each month.

In this problem, we will *not* be dealing with a minimum monthly payment rate.

The following variables contain values as described below:

1. `balance` - the outstanding balance on the credit card
2. `annualInterestRate` - annual interest rate as a decimal

The program should print out one line: the lowest monthly payment that will pay off all debt in under 1 year, for example:

```
Lowest Payment: 180
```

Assume that the interest is compounded monthly according to the balance at the end of the month (after the payment for that month is made). The monthly payment must be a multiple of \$10 and is the same for all months. Notice that it is possible for the balance to become negative using this payment scheme, which is okay. A summary of the required math is found below:

```
Monthly interest rate = (Annual interest rate) / 12.0
Monthly unpaid balance = (Previous balance) - (Minimum fixed monthly payment)
Updated balance each month = (Monthly unpaid balance) + (Monthly interest rate x Monthly unpaid balance)
```

Test Cases to Test Your Code With. Be sure to test these on your own machine - and that you get the same output! - before running your code on this webpage!

Problem 2 Test Cases

Be sure to test these on your own machine - and that you get the same output! - before running your code on this webpage!

Test Cases:

```
1.
    Test Case 1:
    balance = 3329
    annualInterestRate = 0.2

    Result Your Code Should Generate:
    -----
    Lowest Payment: 310
```

```
2.
    Test Case 2:
    balance = 4773
    annualInterestRate = 0.2

    Result Your Code Should Generate:
    -----
    Lowest Payment: 440
```

```
3.
```

Test Case 3:
balance = 3926
annualInterestRate = 0.2

Result Your Code Should Generate:

Lowest Payment: 360

```
1 initBalance = balance
2 monthlyInterestRate = annualInterestRate / 12.0
3 month = 0
4 minPay = 10
5 def calculate(month, balance, minPay, monthlyInterestRate):
6     while month <12:
7         unpaidBalance = balance - minPay
8         balance = unpaidBalance + (monthlyInterestRate * unpaidBalance)
9         month += 1
10    return balance
11 while calculate(month, balance, minPay, monthlyInterestRate) > 0:
12     balance = initBalance
13     minPay +=10
14     month = 0
15     calculate(month, balance, minPay, monthlyInterestRate)
```

Press ESC then TAB or click outside of the code editor to exit

Unanswered

Hints

Hint: How to think about this problem?

- Start with \$10 payments per month and calculate whether the balance will be paid off in a year this way (be sure to take into account the interest accrued each month).
- If \$10 monthly payments are insufficient to pay off the debt within a year, increase the monthly payment by \$10 and repeat.

Hint: A way of structuring your code

- If you are struggling with how to structure your code, think about the following:
 - Given an initial balance, what code would compute the balance at the end of the year?
 - Now imagine that we try our initial balance with a monthly payment of \$10. If there is a balance remaining at the end of the year, how could we write code that would reset the balance to the initial balance, increase the payment by \$10, and try again (using the same code!) to compute the balance at the end of the year, to see if this new payment value is large enough.
- **I'm still confused!**

A good way to implement this problem will be to use a loop structure. You may want to refresh your understanding of **while** loops. Think hard about how the program will know when it has found a good minimum monthly payment value - when a good value is found, the loop can terminate.

- Be careful - you don't want to overwrite the original value of `balance`. You'll need to save that value somehow for later reference!

Reminder: Only hit "Check" once per submission. We are unable to give you more than 30 checks.

Important

Only hit "Check" once per submission. You only get 30 checks per problem.

** Our automatic grader may take a few minutes to respond with feedback. If you hit "Check" multiple times, you will lose a check for every press of the button.

** If you're unfamiliar with how our autograder works, first try out one of the infinite check problems in the Functions lecture sequence.

** Please be judicious with your checks, as we are unable to give you more than 30 checks. However this should be more than sufficient: if you do your code development in your local environment, and ensure you can pass our test cases, you should not require more than a few checks once you paste your working, tested code into our code box.

If you believe you have correct code but it is marked incorrect after clicking "Check"...

** After you submit your code, you can see every test case the graders runs on your code. They compare what your code outputs with what our answer code is supposed to output. Click the small link titled "See Full Output" below the Test Results header.

"Staff Debug: L397 Error" means your code has an infinite loop...

** Clicking Check may give you the error:

```
There was a problem running your solution (Staff debug: L379).
We couldn't run your solution (Staff debug: L397).
```


This means your code is taking too long or has an infinite loop. Test your code with more unique test cases, such as very large or very small values.

Do not define your own values

** For problems such as these, do not include `input` statements or define variables we told you would be given. Our automated testing will provide values for you - so write your code in the following box assuming those variables are already defined. The code you paste into the following box **should not** specify the values for the variables `balance` or `annualInterestRate`

Submit




You have used 1 of 30 attempts

 Your answers were previously saved. Click 'Submit' to grade them.

Problem 2 - Paying Debt Off in a Year

Hide Discussion

Topic: Problem Set 2 / Problem 2

Show all posts	by recent activity
 recurent relation into a general formula	4
Kindly asking for help to transform the recurent formula $b_n = b_{n-1} * a - c$ into a closed (likely arithmetic sequence) formula. a, c are c...	
 Confused with the math	2
I don't understand how in each case the lowest payment is calculated. For example in Test case 1, the balance is 3329, with an annu...	
 I have a working code (tested it in spyder and works fine for all examples) but get an error message when pasting it in. What do i need to change?	4
This is the error message I get: "TypeError: unorderable types: str() > int() *** ERROR: Expected to find a number in the line TypeErro...	

< Previous

Next >



[About](#)
[Affiliates](#)
[edX for Business](#)
[Open edX](#)
[Careers](#)
[News](#)

Legal

[Terms of Service & Honor Code](#)
[Privacy Policy](#)
[Accessibility Policy](#)
[Trademark Policy](#)
[Sitemap](#)

Connect

[Blog](#)
[Contact Us](#)
[Help Center](#)
[Media Kit](#)



© 2022 edX LLC. All rights reserved.
深圳市恒宇博科技有限公司 [粤ICP备17044299号-2](#)