We expect that you will complete mini projects ***individually***. This means that you ***cannot***:

- Share code with friends.
- Find code on the internet.
- Write code together with a partner.
- Submit ***any*** code that was not written entirely by you.

This does leave room for limited collaboration. The following things ***are acceptable:***

- Talking to a partner about strategies you are using to code this project.
- Asking for help ***with code that you have written***.
- Offering to help a friend ***with code that they have written*** as long as you do not give them your code, or copy their code.
- Asking instructors for help.

Mini Project #1 – Finance Calculator

You are working for a bank, and you need to write a function to help that bank do some financial calculations. The bank wants you to create a **single** function that can accomplish multiple calculations. This mini-project requires you to create a MATLAB function that can perform various financial calculations on two numbers, and returns a single output variable with the result. You should name your function **`netIDfinances`**, where netID is your netID (as in aaa111).

Inputs

Your function should take three inputs in the following order:

`account` – The first number in the calculation you want to perform.

`adjustment` – The second number in the calculation you want to perform

`indicator` – A variable, that has a data type of your choice[1], that the user will use to determine which operation to perform. **You should note in a comment how you are implementing the indicator variable.**

Outputs

Your function should have one output, `result`. Your function should be able to perform the following operations:

- **A transfer**. This means that you are transferring the amount of money indicated by `adjustment` to the account with a current balance indicated by `account`. The `result` output should be the sum of these two numbers.
- **A withdrawal**. This means you are taking out the amount of money indicated by `adjustment` from the account with the current balance indicated by `account`. The `result` output should be the difference between the two numbers. *You do not need to worry about the account being overdrawn (having less than 0 balance).*
- **Interest**. This means that the account with a balance indicated by `account` has either an interest payment or accruement with the interest rate indicated by `adjustment`. The `result` output should be the amount to be paid or accrued based on the balance and interest rate.
- **Interest payment**. This should do the same calculation as Interest, but `result` should be the balance in the account after the payment has been made rather than the amount of the payment.

- **Interest accruement**. This should do the same calculation as Interest, but `result` should be the balance in the account after the interest amount has been accrued into the account.

Note that every time this function is called, it should only perform ONE of these operations. However, the user should be able to choose from any of the operations with each call using the indicator input.

If a user calls your function with an indicator value that is not associated with anything, please return a string with a message indicating this as your output instead of the result of a calculation. **You can choose what that message is.**

For example, if you were to use a string for your indicator[1] input and wanted to transfer money, the call of:

```
result = aaa111calculator(1000,10, 'addSomeMoney');
```

Might store the value of 1010 in the variable `result`.

In another example, using a numerical variable for the indicator[1], and associating 10 with the interest payment operation, a call of:

```
result = aaa111calculator(125, 0.01, 10);
```

Would store the value of 123.76 in the variable `result`.

Notes:

[1]It is your choice to determine how the indicator variable works, and there are quite a few ways to approach it and set up the associated conditional statements. Note that the above two examples are for two differently written functions: One that uses strings for the indicator, and one that uses numbers. If you are interested in strings, you should look into using the `strcmp` function. **You may not use the exact indicator names or values I have used in these examples.**

(Note for students with prior programming experience, or students looking for an extra challenge. Try coding interest payment and interest accruement recursively. If you do this correctly, your entire program will only one + operation, one - operation, and one * operation! Recursive programs are programs that call themselves)