Bootcamp Exercise 1: Control and Flow

Due Date

- at least 1 commit by 1:00 PM today
- another commit by 6:00 PM today
- final commits must submitted to your repo by Thursday, September 24th

Exercise 1) Write a for loop statements so that it runs from 1:9 and prints the following output to your screen:

```
*
```

Exercise 2) Modify your for loop so that it prints 10 asterisks, with each asterisk separated by exactly one ampersand sign (&), with no spaces or new line characters.

Exercise 3) by hand, figure out the initial values of these variables and values at the the start and end of each iteration of the loop

```
dogs <- 10;
for (i in 1:5){
    dogs <- dogs + 1;
}
###
meatloaf <- 0;
for (i in 5:9){
    meatloaf <- meatloaf - i + 1;
    cat(meatloaf)
}
###
bubbles <- 12;
for (i in -1:-4){
    bubbles <- i;
}
```

Exercise 4) modify this code so that it will print out a message during presidential as well as congressional election years

```
###you can use the if statement with the modulus operator to conditionally perform ope
years <- c( 2015, 2016, 2018, 2020, 2021)
for(ii in 1:length(years)){
    if(years[ii] %% 2 == 0){
        cat(years[ii], 'Hooray, congressional elections!', sep = '\t', fill = T)
    }

}
```

Exercise 5) More fun with loops. Here are the bank accounts from seven randomly selected UCLA grad students

```
bankAccounts <- c(10, 9.2, 5.6, 3.7, 8.8, 0.5);

Now look at the error message the following lines of code produce. Can you think of a

interestRate <- 0.0125;
for (i in 1:length(bankAccounts)) {
    compounded[i] <- interestRate*bankAccounts[i] + bankAccounts[i]; }

HINT: variables must be initialized before you can perform operations on them
HINT 2: look at the rep() function and see if you can use that to initialize a variabl
```

Exercise 6) Go back to the compounded interest example. Suppose we now want to compound the interest annually, but across a period of 5 years. The for loop we discussed earlier only compounds for a single year. Try this:

```
bankAccounts <- c(10, 9.2, 5.6); #define bank accounts here
interestRate <- 0.0525;
house <- c(4.8, 3.8, 5.7); #deduct
food<- c(3.5, 4.3, 5.0);    #deduct
fun <- c(7.8, 2.1, 10.5);  #deduct
#and incomes (through TAships) of
income <- c(21, 21, 21); #add this

for (j in 1:5) {
    for (i in 1:length(bankAccounts)) {
        #step 1 modify bankAccounts so that amounts reflect income and expenses
        #step 2 get calculate interest and add to accounts from step 1
        #you can actually use the line you have already written if you
        #modify amounts in bankAccounts directly in step 1
    }
```

Exercise 7) Three students have estimated annual expenditures for food, housing, and fun of: (in thousands of dollars)

```
house <- c(4.8, 3.8, 5.7);
food<- c(3.5, 4.3, 5.0);
fun <- c(7.8, 2.1, 10.5);

#and incomes (through TAships) of

income <- c(21, 21, 21);
```

Modify the 5-year interest-compounding code from #5 and #6 so that it runs from 2015-2020 and so that in odd numbered years students 1 and 3 get trust fund disbursements of $5000. (hint the modulus function %% will be helpful)

Exercise 8) use a while loop to sum all numbers from 1:17. You will need to use a counter variable (like index seen in class).

Exercise 9) write a function takes a number, and prints ‘small’ if number less than or equal to -1; ‘medium’ if between -1 and + 1’big’ if

greater than or equal to + 1