Homework 1
Thanksgiving Dinner Binge

Due: Monday September 26th at 6:10 PM

Great programmers such as yourselves are known for their ability to eat a LOT of food; you do eat chips all day after all. Thanksgiving is coming up and your mom wants you to eat with the family. However, you DREAD the idea of getting up from your chair, leaving your comfortable dark room, and worst of all: socializing. Thus, you decide you are just going to write a VB program to simulate Thanksgiving's best part, the food.

Your program is going to simulate the process of eating from several thanksgiving dishes (turkey, mashed potatoes, cranberries, etc: name them what you want). Each of the **THREE** dishes will be a separate button on a form and you eat from it by simply clicking on the button. When you eat food it goes into your stomach; however, your stomach can only hold 15 (or 50: typo so you pick) portions/servings at any one time. To eat more food you must first click the digest button. This button removes all the food from your stomach and adds to the "total amount eaten" counter.

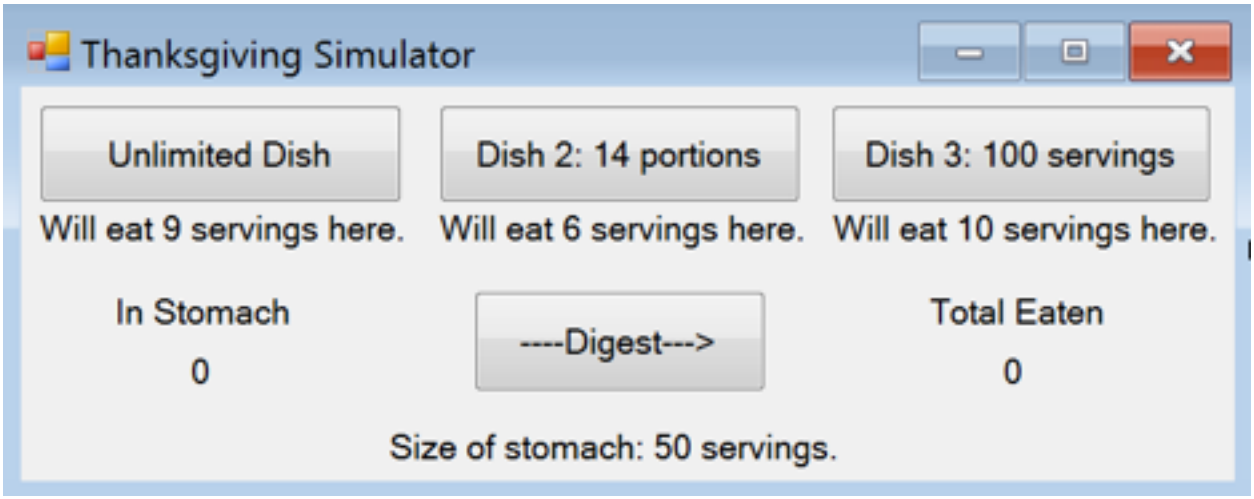
1. Dish 1: This is the easiest to code. It has an **unlimited** supply of food available. Every time you click you should eat **9** servings.
2. Dish 2 only has **40** servings in it. Each time you click you should eat **6** servings. Don't go negative, negative food don't exist! Also, make sure to take the remaining couple servings when there are less than 6 left (assuming you have room to eat them).
3. Spot three has **100** servings available but you shouldn't be greedy here. Take **10** servings the first time, then **9** servings, then **8**, and so on. Make sure that you don't start adding portions to the pile (eating negatives).

Whenever you are about to eat food make sure you have room to eat it and the dish still has food available!

This homework reviews the concepts of variables, constants, scope, conditional statements, and use of variables to keep track of state. Your “stomach” and your “total eaten score” should be implemented with global/module level variables. Each of the dishes’ food-pools can EITHER be implemented as **static** variables or as **global/module** level variables. Do not allow the food to go negative or create food out of nothing.

DO NOT store the numbers inside the labels themselves. Labels should only reflect the numbers that are stored in variables, not the other way around.

Here is a sample of what your form **could** look like:



You can make it fancier if you want with images or Thanksgiving colors. Do **not** make it garish or hard to read with bad color choices or messy/distracting images. You should update the numbers on the buttons to reflect how many servings are left. Also update the label under the 3rd dish to show how many would eat from that dish.

You ***must*** comment each and every line of your code to explain what its purpose is.

```
Dim intHomeworksCompleted As Integer = 3 'intCounter will count the
      number of homeworks I have completed this semester.
If intFriendHomeworks > intHomeworksCompleted Then
      'This code will execute if my friend has done more hws than I.
      …… blah blah more code and more comments
End If
```

Static Variable Example

Note: this is not required for the homework but is a **much** cleaner way to do it. As a general rule you should **avoid global variables <u>whenever</u> possible**.

If you want to use static variables for your homework to keep track of the house's candy pools this example will help you understand what a static variable is and how it differs from a normal variable.

Copy and paste the following into the button click sub procedure for a button named "btnExample".

```
Dim myNormalCounter As Integer = 100
myNormalCounter = myNormalCounter - 1
btnExample.Text = Convert.toString(myNormalCounter)
```

Try to click the button a few times. You will notice that the button will show 99 forever. Now try changing the "Dim" to either "Static Dim" or just "Static" and click the button.

```
Static myNormalCounter As Integer = 100
myNormalCounter = myNormalCounter - 1
btnExample.Text = Convert.toString(myNormalCounter)
```

Now it will count down from 99 one at a time. Why?

A normal variable has a lifetime (called it's scope). When the sub procedure that created the variable ends, the variable is destroyed. The next time the sub is called the variable is created again with its default value (in this case 100).

A static variable is just like a normal variable with one crucial change. It's lifetime is extended, it will never be reinitialized. The first time you call the sub the variable is created and initialized to the value specified. The next time you call the function, the variable still exists and will not be reset/reinitialized to the value specified.

It is important to see that static variables are similar to module level variables in that they are only created once (never reset). However they are different in that you can only access them inside of their scope, unlike module level variables which can be accessed anywhere. This is useful in that you can use the same name in different places to refer to different variables, just like local variables. Additionally you wont pollute the global space. If any of this confuses you, come to office hours!