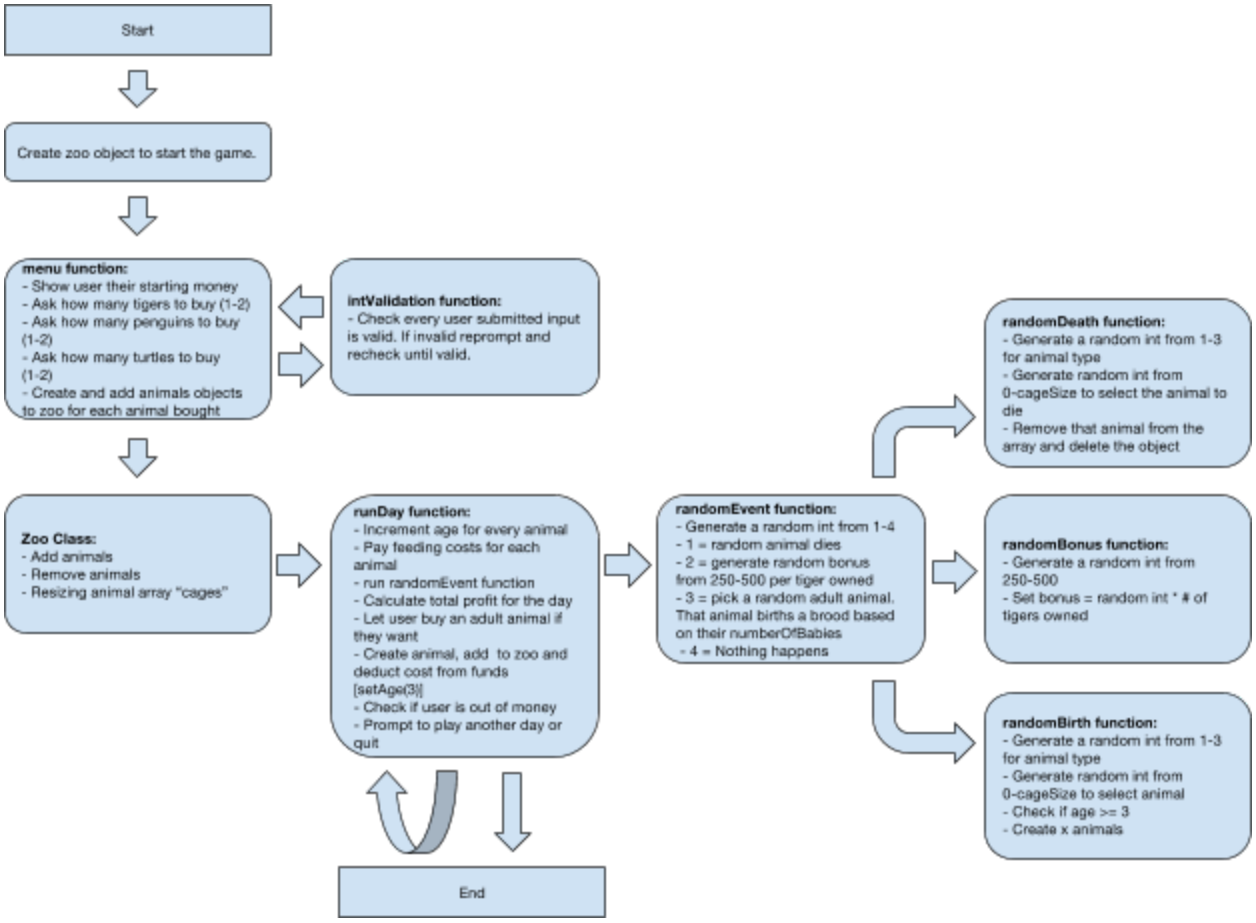
Joel Huffman
2/3/2018
Project 2 Reflection

Design:



Test Tables:

Test Case	Input	Expected Result	Observed Result
Game over from overspending	Buying tigers with < $10,000 in funds	You have run out of money. Game Over	You have run out of money. Game Over
Random boom for tigers	Random bonus (250-500) with 2 tigers	You get a bonus of ~$375 per tiger today. Netting you $750 for your 2 tigers	You get a bonus of $448 per tiger today. Netting you $2240 for your 5 tigers
Buying one of each animal initally	1 1 1	You have $48900 remaining	You have $48900 remaining
Buying two of each animal initially	2 2	You have $47800 remaining	You have $47800 remaining

Joel Huffman
2/3/2018
Project 2 Reflection

	2		
Buying animals at every opportunity	Always purchasing an animal	No crashes	No crashes

Reflection:

This project has been one of the hardest assignments for me for this class. While my overall program design from the flowchart above remained pretty stable, I had to continually make small changes to features and functions I already had tested and working because they wouldn't work with the next function I was working on.

This included switching from one 2D dynamic array of pointers to animals to house all three animal types (one animal type for each column in the array). This proved to be an issue once dynamically resizing the animal cages came into the picture. I wanted each individual cage to be able to be resized individually as need be, rather than all three simultaneously when only one cage really needs it. This required me to switch to three separate dynamic arrays of pointers to animals.

I originally created generalized functions for add animals, removing animals and doubling cage size that could be used for all three animal types. These functions ended up being really unwieldy because they required so many references and pointers to each animal's information. I ended up creating separate functions, specific for each animal, all with the same functionality. This was not an elegant solution, but I had to get the program functional in the limited timeframe.

I spent a lot of my debugging time working with memory leaks and ensuring all the animal object pointers declared with "new" were properly deallocated with "delete". This was a huge hassle and I think working with vectors would have removed much of the headache from this assignment.

I plan on coming back to this assignment when I have some free time to take another crack at incorporating the multiple animal functions together and simplifying the overall layout.