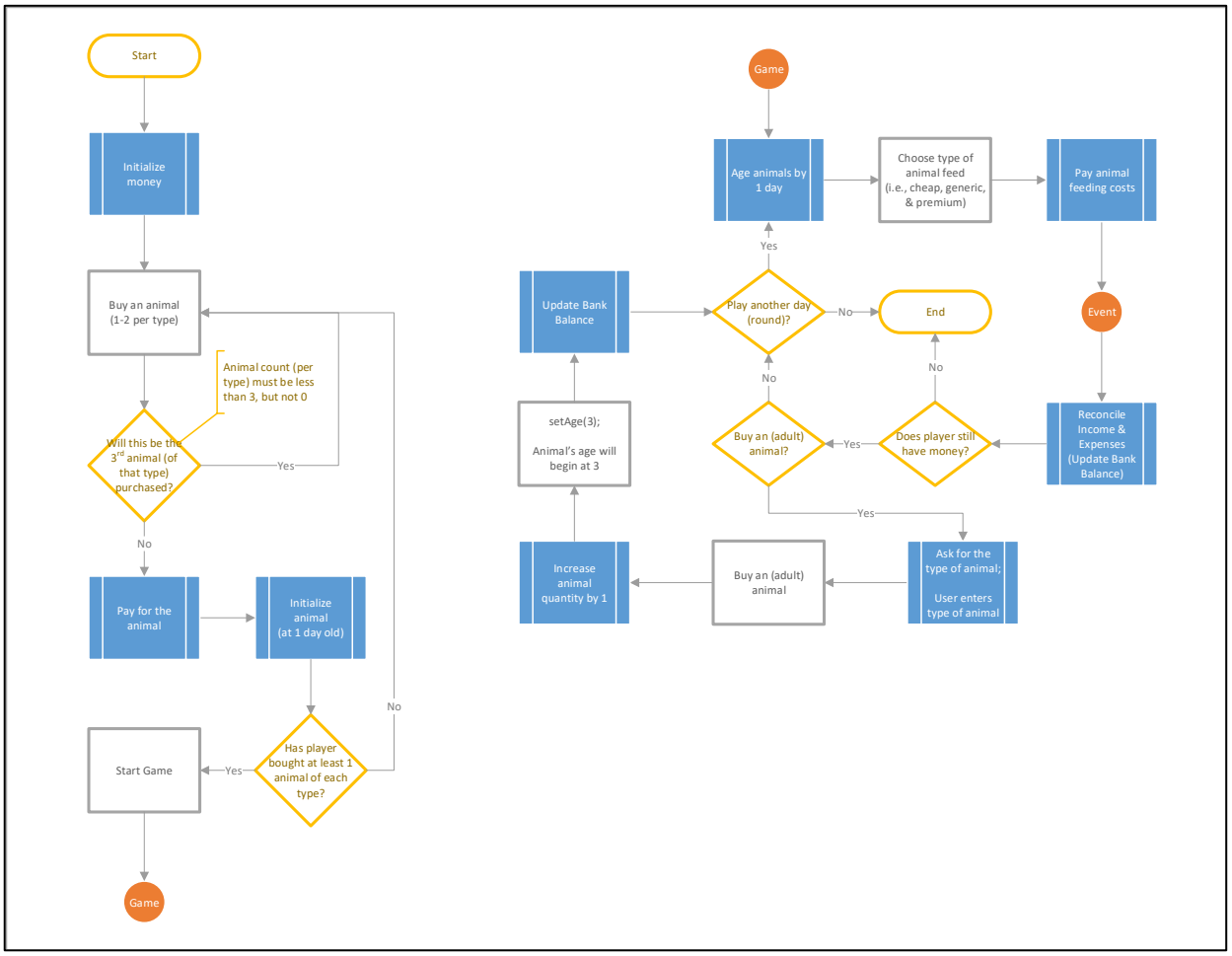
Ryan DiRezze
CS 162 400
Project 2

PROJECT 2 (DESIGN + REFLECTION)

Project 2 (Zoo Tycoon) design, test cases, and reflection

DESIGN / FLOWCHART

Project 2 flowchart (PDF) is included within this assignment submission's .zip file



TEST CASES

Program test tables, including: test plan, expected output, and actual output. See the .zip file for this in Excel.

Test Case	Input Value(s)	Driver Functions
Incorrect menu selection (special chars)	Input = %, &, *, (,), #, $	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Incorrect menu selection (positive integers)	Input >= 0	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Incorrect menu selection (negative integers)	Input < 0	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Incorrect menu selection (wrong letter)	Input = (letters != 'A', 'B', 'a', or 'b')	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Correct menu selection to begin program (uppercase char)	Input = 'A'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Correct menu selection to quit program (uppercase char)	Input = 'B'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Correct menu selection to begin program (lowercase char)	Input = 'a'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Correct menu selection to quit program (lowercase char)	Input = 'b'	menu() if(tolower(input) == 'a') else if(tolower(input) == 'b')
Wrong input (special char) for choosing initial starting animals	Input = #, $, %, ^, &, *	Zoo() if(tolower(input) == 'a') else if(tolower(input) == 'b') else if(tolower(input) == 'c')
Wrong input (positive integer) for choosing initial starting animals	Input >= 0	Zoo() if(tolower(input) == 'a') else if(tolower(input) == 'b') else if(tolower(input) == 'c')

REFLECTION

This assignment reflection will cover the following content: (1) Design changes while developing and implementing the assignment's program, (2) problems that were experienced and how those problems were overcome, and (3) what was learned from this assignment.

From my initial flowchart (design) draft to my final draft, the primary design changes related to the modularization of the program's code with functions. When I started to write code, I had created a very basic flowchart, and thus, I was not able to derive much value from my very rough draft of a flowchart. This assignment's complexity brought the importance of a well-done program design, which can be done with a flowchart, to light. And I found that a solid foundation, which can be found in a quality program design, or flowchart, is priceless when implementing a clean and modularized program successfully.

As I began to work on the program, I recognized that a variety of different operations kept appearing within the program, which possessed minor differences from one instance to the next. For example, rather than implementing various functions within main() in order to meet the requirements of completing a "round", or a day, as it's called with respect to the program. What I chose to do was make one function that performed all required transactions to pass a round. By implementing and using a single function, "payRound()", my main() file's implementation of the Zoo Tycoon game was very clean.

The example that I just described was one example of a situation where I started to implement and use functions within main() to pass a round, but I changed my design shortly after beginning. The targeted design was one function, "playRound()", which would run the user through an entire round of Zoo Tycoon. As far as problems go, I found that it was difficult to keep track of what each function and line of code was doing before I condensed a round of Zoo Tycoon into one function call.

After recognizing the problem, I utilized a flowchart to visually depict the program in an easy to digest way, which helped me properly break down "chunks" of the program into separate process steps and decision points. I learned that taking the additional time up-front pays significant dividends when you're in the middle of programming and defining and implementing code. Although a program design has been required by previous assignments, I didn't quite realize the significance of a well-done modular design, which is easily created when drafting a program's flowchart. Going forward, I now recognize the immense value of creating a clean and modular design, as my messy start to this assignment was salvaged after I spent some quality time refining the flowchart while capturing key "chunks" of code, which were implemented as functions.