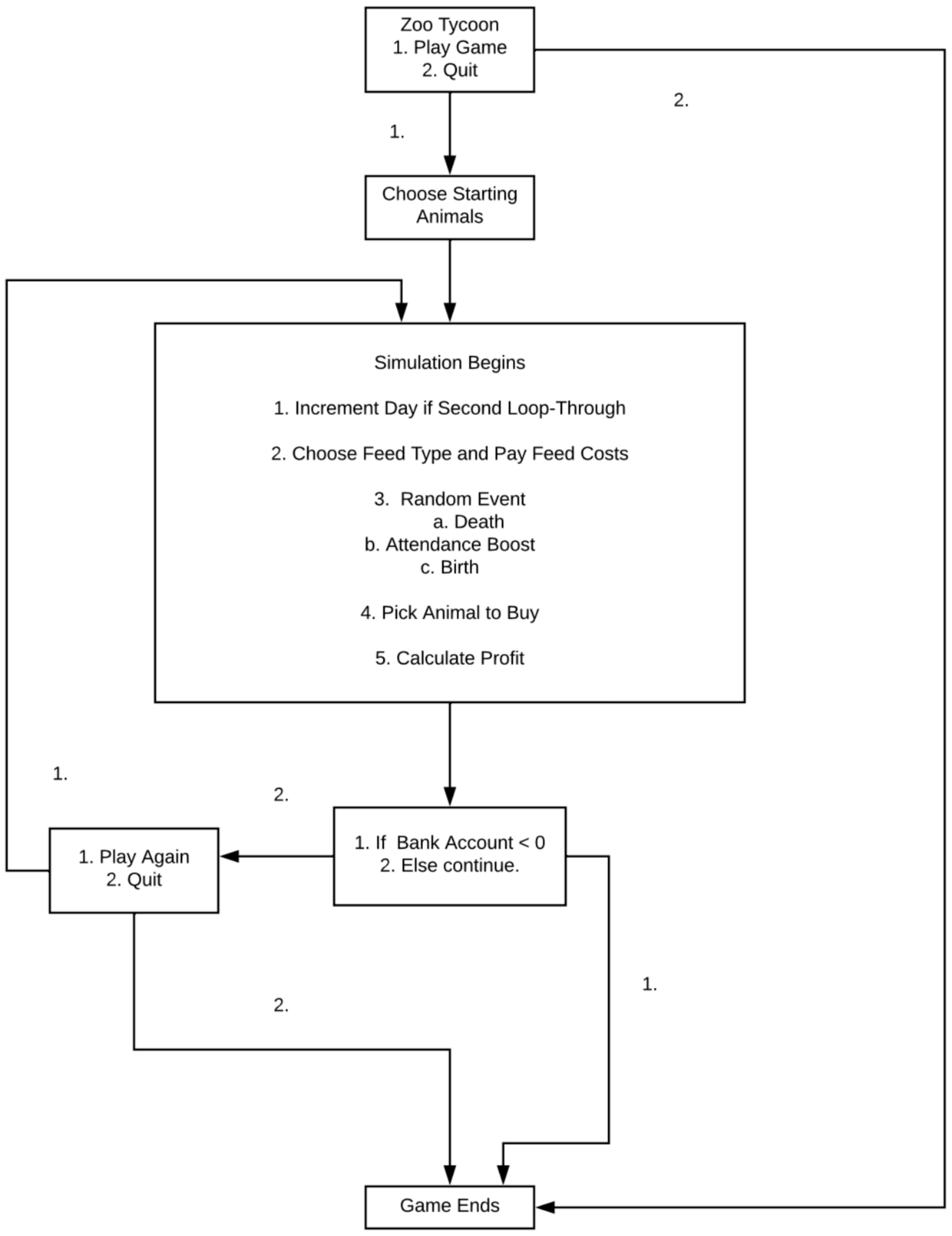
Devin Gendron
Project 2 – Zoo Tycoon

Flowchart:



PseudoCode:

```
Main()

        //start menu function
        //switch statements to play game or quit

startMenu()

        //print statements
        //use inputvalidation func to receive proper return
        //return value

endGameMenu()

        //print statements
        //use inputvalidation func to receive proper return
        //return value

promptUserMenu()

        //using inputval(); code, retrieve feedtype
        //using inputval(); code, retrieve animalchoice

zooClass()      -take all input from menus

        //zooMenu (starting menu)
        //startZoo (pick starting animals)
        //pick feed type
        //add and subtract functions for animals
        //zoo sim function
                //increment day
                //pay feeding costs
                //random event
                        //animal death
                        //attendance boost
                        //animal birth
                //calculate profit
                //buy animal
```

```
simulation()
    //increment day
    //pay feeding costs
    //random event
        //animal death
        //attendance boost
        //animal birth
    //calculate profit
    //buy animal

addanimal()
    //check for cost or birth
        //check if cage is full
            //double if full
        //if first entry
            //add animal
        //if spot is null
            //add animal
    //else
        //say not enough money

increment day()
    //increment day
    //increase age of all animals

feedchosen()
    //pick feed type

pay feeding costs()
    //calculates feed costs
    //subtract from bank

random event()
    //randomizer
        //death function
        //attendance function
        //birth function

animal death()
    //remove random animal from animal array

attendance boost()
    //randomizer for bonus amount
```

```
animal birth()
    //randomly pick animal type
        //add animal function

calculate profit()
    //calc profit from all animals and bonuses.
    //add to bank account

buy animal()
    //choose which type of animal
        //add animal function
```

Test Plan:
zooMenu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

endGameMenu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

inputValidation();

Test Case	Expected Outcomes	Observed Outcomes
Input Letters	Incorrect entry counter incremented. Restarts loop.	Restarts loop for proper entry
Input symbols	Incorrect entry counter incremented. Restarts loop.	Restarts loop for proper entry
Input too low	badEntry bool sets to true, restarts loop until correct input.	Restarts loop for proper entry
Input too high	badEntry bool sets to true, restarts loop until correct input.	Restarts loop for proper entry
Input empty string	badEntry bool sets to true, restarts loop until correct input.	Restarts loop for proper entry
Correct entry	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

feedChosen();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Incorrect entry counter incremented. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Incorrect entry counter incremented. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

buyAnimal();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Incorrect entry counter incremented. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Incorrect entry counter incremented. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement.	Returns value to main to continue with program

startZoo();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns "Incorrect entry…"	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns "Incorrect entry…"	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement.	Returns value to main to continue with program

addAnimal(1);

Test Case	Expected Outcomes	Observed Outcomes
1 = First buys	Allow user to enter first animals	First animals chosen

addAnimal(2);

Test Case	Expected Outcomes	Observed Outcomes
2 = animal birth	Random animal bred	Random animal bred

addAnimal(3);

Test Case	Expected Outcomes	Observed Outcomes
3 = adult animal bought	Adult animal added to zoo	Adult animal added to the zoo

feedChosen(1);

Test Case	Expected Outcomes	Observed Outcomes
1 = cheap	Food half as cheap, but animals twice as likely to die	Food has proper cost and animals twice as likely to die

feedChosen(2);

Test Case	Expected Outcomes	Observed Outcomes
2 = generic	Food same as base cost	Food has proper cost, no change in likelihood of death

feedChosen(3);

Test Case	Expected Outcomes	Observed Outcomes
3 = premium	Food twice as expensive and animals twice as likely to live	Food has proper cost and animals twice as likely to live

Zoo simulation();

Test Case	Expected Outcomes	Observed Outcomes
Sim runs	Loops and asks user proper prompts and menus	All menus and prompts properly executed

Reflection:

When I first read the Project 2 assignment requirements, I was a little intimidated. The project seemed to be quite intensive and require a lot of technical work. My initial reaction was to use vectors to control the different types of animals, however with the requirement to use arrays, I knew I was going to have a bit of difficulty coding that portion of the assignment. So, I began designing.

In my design notebook, I drew up some rough flowcharts for how I wanted my project to run and began the pseudo code for how I wanted each function to operate. The design was coming together nicely and puzzling my functions in accordance with my flowchart and the assignment requirements was easy. I then began constructing my test plan. Some test plans were able to be carried over due to my repeating input validation and menu functions, however others needed to be created.

It was at this point that I began coding my design. I did not have to make many design changes while coding, as I pre-planned almost everything. The biggest problem I encountered were the dynamic arrays for each animal. I was struggling getting the arrays to work properly even when my program would compile. I was getting seg fault after seg fault after seg fault. I spent a lot of time in office hours the first half of the fourth week speaking with the TA's and discussing how I could work this section of my code to get it running. Eventually after all the advice from the TA's and a lot of research, I got my program running properly! At this point, the rest of my program had no major issues. Some tweaking here, some tweaking there, and it was done! I had included the extra credit design for differing feed types into my program, so I then included that and got it running properly within the program.

Reflecting on this project, it was quite difficult – even if it was just the portion that controlled the creation, filling, deleting, and doubling of the arrays. However, once this was portion was accomplished, the rest of the program wasn't so intimidating. I felt as if the Zoo Tycoon project was a really great learning experience. Having us use arrays instead of vectors really forced us to think outside the box and utilize dynamic memory differently than we would have with vectors. What I really learned during this assignment was mainly a more in depth understanding of dynamic pointers and arrays. I felt very shaky on more complicated application of pointers coming into this class, but I’ve been more and more comfortable after each assignment is submitted. I’m very excited at my development as a programmer and can’t wait to take on the next challenge.