Program Design & Testing Document for Program 3+
Lyell C Read

Problem Statement

This problem/assignment is asking me to construct a C++ program that:

- Allows the user to play Zoo Tycoon
- Uses classes to implement that game as follows:
    - Using a parent class that has the elements:
        - Age
        - Cost
        - Babies
        - Food Cost
        - Revenue
    - Using a class for each animal that has specific traits and values for the above elements, as it inherits from that parent.
- The user must start with 100,000, be able to purchase animals.
- The user can buy up to two adults of one species. Each turn can only see the user buy one species. Adults start at 3 years.
- It costs a random amount to feed each animal per day.
- The user must be the victim of a special event (described in guide)
- Zoo goes bankrupt when the user runs out of cash
- Classes must be for sea otter, sloth, monkey (inherit from animal) and zoo class.
- Within Zoo, dynamically allocated array of all species of one kind makes up an exhibit.
- No memory leaks
- Use Makefile for compilation.

I assume:

- NOTHING! I will check every input, so nothing is assumed.

This I will achieve by:

- A program that uses classes to provide the user a fine game of Zoo Tycoon
- Deallocating memory when done if it was dynamically allocated.

Understanding the Problem

As described in the requirements, the problem asks me to create a program that can play a nice game of Zoo Tycoon using classes and inheritance. The game will be neatly organize, and follow the typical rules of the game listed in README.md, on GH.

Pseudo Code (Simplified)

Parent Class: animal:

Age
Cost
Babies
Food cost Multiplier
Revenue

Functions:

Increase Age	pre:none; post: increased age
Get Age	pre:none; post: age returned
Get Cost	pre:none; post: cost returned
Babies	pre:none; post: babies maintenance complete
Get Food Cost	pre:none; post: food cost returned
Get Revenue	pre:none; post: revenue returned

Class: Zoo

Current capital
Sloth exhibit array of Sloths
Monkey exhibit array of Monkeys
Sea Otter exhibit array of S.O's

Functions:

Run_Game	pre:none; post: Game complete
Check_For_Broke	pre:none; post:returns if player is broke
Resize Exhibit	pre:exhibit is defined; needs to be resized.; post:resized array of exhibit to exhibit+1
Print Data	pre:none; post:data printed
Buy Animals	pre:none; post:purchase animals menu and choice
Play_Turn	pre:none; post:turn played
Incur Costs	pre:none; post:all player costs are incurred
Have Babies	pre:none; post:babies process is completed
Generate Profit	pre:none; post:profit is calculated
Special Event	pre:none; post:special event manager
Feed Animals	pre:none; post:animals fed

Class Sloth : Animal

Babies at a time = 3
Food cost multiplier = 1
Cost = 2000
Revenue = .05*cost = 250

Class Sea otter: Animal
    Babies at a time = 2
    Food cost multiplier = 2
    Cost = 5000
    Revenue = .05*cost = 250

Class Monkey: Animal
    Babies at a time = 1
    Food cost multiplier = 4
    Cost = 15000
    Revenue = .1*cost = 1500

Main
    Zoo mz;
    mz.Run_Game

Zoo::Run_Game
    While !Broke:
        Play_Turn
        Incur Costs
        Have Babies
        Generate Profit
        Special Event
        Feed Animals

Zoo::Play Turn
    Print Data
    Buy Animals

Data Verification.

Checking only occurs on user input of a number from the list of options on a menu {1: buy; 2: Skip turn; 3: Quit} and buy menu {1: Sloth; 2: S.O; 3: Monkey}. Input listed as {menu1, menu2}. * indicates failure before second input.

Ask for value from other player	**What Should Happen**	**Does This Happen**
{“”,*}	Error - please enter again	
{1,””}	Error - please enter again	

{12, *}	Out of range. Enter again	
{1,1}	Sloth Purchase	
{2,*}	Turn Skipped	
{3,*}	Quit	
{1,15}	Out of Range	
{1,””}	Error - please enter again	
{“r”,*}	Error - please enter again	
{1,”ersadsfjasdf”}	Error - please enter again	