Program 3

Due May 9, 2021 by 11:59pm **Points** 90 **Submitting** a file upload
File Types tar and cpp **Available** Apr 24, 2021 at 12am - May 14, 2021 at 11:59pm 21 days

This assignment was locked May 14, 2021 at 11:59pm.

Zoo Tycoon

Motivation

The goal of this assignment is to start working with inheritance in C++. As you implement your program you'll also expand your knowledge and familiarity of object-oriented programming.

Problem Statement

For this assignment, you will implement a small game where the player acts as the owner of the city's zoo, which has exhibits of lemurs, tigers, and black bears. As the owner, it's the player's job both to ensure the welfare of the animals and to generate as much profit as possible. They accomplish these goals by investing in animals, feeding them, caring for them when they are sick, and raising their babies when they are born.

Each species of animal has these specific traits:

- **Age**
  - For all species, an animal is a baby if it is less than 5 months old.
  - For all species, an animal is an adult if it is at least 4 years old.
  - Note: For our purposes the age range between 5 months and 4 years can be considered as adolescence. The animal is no longer a baby but is not yet capable of reproducing.
- **Cost**
  - Lemurs cost $700 each.
  - Tigers cost $15,000 each.
  - Black Bears cost $6,000 each.
- **Babies**
  - Lemurs have 1 baby at a time.
  - Tigers have 3 babies at a time.
  - Black Bears have 2 babies at a time.
- **Food cost**
  - The base daily food cost varies on a month-to-month basis (see below).
  - Tigers have a monthly food cost of 5 times the base food cost.
  - Black Bears have a monthly food cost of 3 times the base food cost.

    - Lemurs have a monthly food cost equal to the base food cost.
- **Revenue**
    - Each animal generates monthly revenue equal to a percentage of the initial cost of one of its species.
        - All animals (except lemurs) generate 10% of the cost of one of their species (i.e. each tiger generates $1500 each month, and each black bear generates $600 each month).
        - Each lemur generates 20% of the cost of a lemur (i.e. each lemur generates $140 each month).

Game Flow

The game starts with the owner having no animals and $100,000 in the bank, and it proceeds one month at a time. You can think of each month as a turn for the player. During a single month, several things happen.

1. The age of each animal currently in the zoo increases by 1 month (note that you can track time in days or weeks if you prefer).
2. A special event occurs. The special event is chosen at random from among the following options:
    - One randomly chosen animal gets sick. In order to care for the sick animal, the owner must pay an amount equal to half the initial cost of an animal of the same species as the sick animal (e.g. a sick tiger costs half of $15,000, i.e. $7,500). If the owner has enough money to cover this cost, it is subtracted from their bank account. If they do not have enough money, then the sick animal dies and is removed from the zoo.
    - A randomly chosen adult animal gives birth to the appropriate number of babies for its species (a non-adult can't have babies). This can only occur if the zoo owns two or more of that particular species. For example, black bear babies cannot be born unless the zoo owns at least two adult bears. For all species, each baby starts with age 0 and is added into the zoo.
    - No special event occurs during this month.
3. The owner receives monthly revenue for each animal, as specified above.
4. The owner may buy up to two adult animals of a single species. The owner may only buy one species per month, but they do not have to buy any animals if they don't want to. Each animal that the owner buys is exactly 4 years old. When the owner buys an animal, the cost of the animal is subtracted from the owner's bank account.
5. The owner must pay the upcoming feeding cost for each animal in the zoo (including any they just bought). The cost of food for each animal is calculated using the base cost of food. This starts out as $80. Each month, the base cost changes to a random value between 80% and 120% of the base cost from the month before. Once the cost of food for each animal is calculated, this amount is subtracted from the owner's bank account.

In addition to the specifications above, your game must have these features:

- Each animal less than 5 months old (a baby) generates twice the amount of revenue as an adult animal. These babies (who are less than 5 months old) also cost twice as much as an adult if they get sick.
- If the player runs out of money at any point, the game ends with the zoo going bankrupt.

Other Program Requirements

- You must have a class for each of the following things: zoo, animal, lemur, tiger, and bear.
- You must use inheritance: the classes for lemur, tiger, and bear must inherit some traits and behaviors from the animal class, though each of them may also have unique traits or behaviors as necessary.
- Within your zoo, the exhibit of each species of animal must be represented as a dynamically-allocated array of objects of the appropriate class.
- Your program should implement the game flow described above. The player may play one month at a time until they choose to quit the game. At the beginning of each month, you should let the player know how much money they have in the bank and how many adults (>= 4 years old) and babies (< 5 months old) they have of each species.
- Your program is not allowed to have any memory leaks. It is recommended that you use Valgrind to occasionally test your program as you develop it (even when everything seems to be working).
- The Big 3 must be implemented as appropriate.
- Your program must be factored into interface, implementation, and application. Specifically, you should have one header file and one implementation file for each class, and you should have a single application file containing your `main()` function. You should also include a Makefile that specifies compilation for your program.
- No use of the <algorithms> and <vector> libraries
- Lack of correct coding style will incur an automatic 10% deduction. You must follow the spirit of the assignment.

Extra Credit

In addition to the requirements above, you may earn extra credit as follows:

- (5 points) When the game begins, allow the user to make a one-time selection to choose the quality of feed that the Zoo will purchase:
  - **Regular** - This is the normal food, and it behaves as described above.
  - **Premium** - This food costs twice as much as regular food for all animals, but it reduces the probability of sickness by half.

- **Cheap** - This food costs half as much as regular food for all animals, but it doubles the probability that an animal will get sick.

Implementation Details

For this assignment you have additional freedom to design your own implementation. There are certain baseline requirements as specified in this document. However, the exact member functions and member variables are up to you.

PLEASE... take the time to design your program on paper. Use scratch paper with flow charts or pseudo-code. In particular, think of how your program will flow. What will the constructors do? How will you pass objects and variables between different classes? The old adage "Measure twice, cut once" could certainly be applied to CS162 as "Design twice, program once".

Taking the time to do a thorough program design can save you hours of frustrating debugging time!

Programming Style/Comments

In your implementation, make sure that you include a file header for every .cpp or .h file. Also ensure that you use proper indentation/spacing and include comments! Below is an example header to include. Be sure to review the style guidelines for this class and try to follow them. I.e. don't align everything on the left or put everything on one line!

Style guidelines: **cpp_style_guideline.pdf**
(https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)

```
/*********************************************************
** Program: zoo_tycoon.cpp
** Author: Your Name
** Date: xx/xx/2021
** Description:
** Input:
** Output:
*********************************************************/
```

When you compile your code, it is acceptable to use C++11 functionality in your program. In order to support this, change your Makefile to include the proper flag.
For example, consider the following approach (note the inclusion of **-std=c++11**):

```
g++ -std=c++11 <other flags and parameters>
```

In order to submit your homework assignment, you must create a TAR archive that contains your .h, .cpp, and Makefile files. This tar file will be submitted to Canvas. In order to create the tar file, use the following command:

```
tar –cvf assign3.tar <list of all the .h and .cpp files> Makefile
```

Grading

You are required to **meet with a TA** within two weeks of the assignment due date to demo and receive a grade. You can schedule a demo with a TA online using the link provided on the TAs page. You must use your OSU email to schedule (for FERPA reasons), or your appointment will be cancelled.

Programming assignments that do not compile and run on the OSU ENGR servers will receive a grade of zero, with no exceptions.

Program #3 Rubric

Criteria	Ratings	Pts
Program Header / Good indentation & Use of whitespace / Function Documentation At a minimum, header should contain author's name (2pts) and a description of the program. (2pts) Conditional blocks of code should always be indented. (3pts) Every function contains it's own initial block comment (or multiple lines of comments prior to the function definition) that provides the reader with an explanation of the function's purpose (3 pts). -1 pt for each function that is missing a header (up to 5 point penalty).		9 pts
Classes & Inheritance Implements classes for Zoo, Animal, Lemur, Tiger, and Black Bear (5 pts). Lemur, Tiger, and Black Bear classes are derived from the Animal class (8 pts).		13 pts
Dynamic Memory Usage Collections of each species are held in dynamically-allocated arrays in the Zoo class		9 pts
Animal Purchases / Food Expenses / Special Events Player can buy new animals, and the cost is correctly deducted from their bank account (9 pts). The game correctly calculates the monthly cost of food and deducts it from the player's bank account (9 pts). The game correctly generates special events (sickness or birth) (5 pts).		23 pts
Animal Sickness / Birth Sick animals either result in the correct amount being deducted from the cost of the player's bank account, or they die if the player doesn't have enough money (9 pts). The birth of babies is correctly implemented, with only adult animals giving birth and babies starting with an age of 0 (9 pts) Babies of a species can only be born if the zoo owns 2+ adults of that same species. (4 pts).		22 pts
Monthly Revenue / Bonus Revenue Monthly revenue is correctly added to the player's bank account based on the number and species of animals in the zoo (14 pts).		14 pts
Extra Credit - Special Food Game allows the zoo owner to select premium, regular, or cheap food. This choice should influence the probability of sickness as described in the specifications (5 pts).		0 pts
Auto Deduction Automatic Deductions: -9 memory leaks (use valgrind and sliding scale for how many/bad it is), -9 no makefile or tar, -9 use of globals or libraries not introduced in class, -9 didn't separate files, -2 program crashes during testing (e.g. segmentation fault or similar abrupt halts)		0 pts
		Total Points: 90