Assignment #3 – More OOP in C++/Inheritance
Due: Sunday, 05/08/16, 11:59pm

Grading: EVERY assignment in this course is graded by demoing your work for 10 minutes with a TA. You are required to meet with a TA within two weeks from the due date to demo, and you receive an automatic 50 point deduction for failure to do so. If you miss a scheduled appointment, you will be penalized 10 points for rescheduling within 1 day (24 hours), 25 points within 7 days (1 week), and 50 points for anything outside of a week. Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program☺

(90 pts) **Program Implementation: Problem Statement**

You are the proud owner of a virtual zoo that has places to house tigers, polar bears, and penguins! You will write a Zoo Tycoon game using classes and inheritance. Zoo Tycoon is a game where the user creates a zoo that has exhibits with a number of different animals in each. Your zoo animals can be Tigers, Polar Bears, and Penguins. You start with an initial amount of money in your zoo bank, and from there you can begin to operate.

Specific Animal traits:

- **Age**
  - **>=3, adult <3, baby**
- **Cost**
  - Tigers cost $10,000, Polar Bears cost $5,000 and penguins cost $1,000.
- **Babies**
  - Tigers have 1 baby, Polar Bears have 2 babies, and penguins have 3 babies.
- **Average Food Cost**
  - You can get this average food cost from the user or set it as a constant for an animal.
  - Tigers have a feeding cost of 5 times the average food cost per animal, Polar Bears 3 times the average food cost per animal, and Penguins have an average animal food cost per animal. For example: average food cost $10.
- **Payoff**
  - Tigers average 10% of their cost plus a bonus in payoff per animal, Polar Bears and penguins (like other animals) average 5% of their cost in payoff per animal.

Game Flow:

You begin your zoo with a specified amount of money, e.g. 100,000 dollars. You can buy either one or two of each animal to start your zoo. Each time you buy an animal, the cost of that animal is subtracted from the bank. Each animal bought starts with an age value of 3.

Each turn is a “day”. At the beginning of the day, each animal increases 1 day older, and the user pays the feeding cost of each animal. This is required, so that the animals don’t die. After that money is subtracted from the bank, a randomized event takes place.

Random Events:

1) a sickness occurs in the zoo,
    - ask the user which animal will die,
    - subtract the cost of the animal from your budget.
2) a boom in zoo attendance occurs,
    - generate a random bonus for the tigers, 250-500 dollars
    - add the payoff of each animal back into your budget as a reward
3) a baby animal is born,
    - ask the user for the animal having a baby
    - if there is an animal old enough to be a parent, age > 3
        - add number of babies born to that specific animal type to the exhibit
    - else, reprompt the user for an animal.
    - begin baby at age 0,
4) nothing happens.

After the random event, calculate your payoffs for the day based on normal visits for penguins and polar bears, and possibly a bonus from a boom in attendance to see the tigers!

Last, ask the player if they want to buy an adult animal, continue to the next day, or end the game. If they chose to buy an animal, subtract that cost and add the animal to the appropriate exhibit.

When you add money for profit or subtract money for sickness, the babies, or animals with an age value less than 3 “days”, have a value of 2 times their normal payoff or cost.

Requirements

- Each exhibit is a dynamic array of each type of animal.
- You must have classes for a **zoo, animal, tiger, penguin**, and **polar bear**.
- **You Must Implement Inheritance:** Your tiger, penguin, and polar bear must inherit some traits and behaviors from the animal class, while there may be others unique to the specific animal, such as a bonus for tigers.
- Every time you buy an animal, at the beginning of the game or during play time, subtract that animals cost from the bank.
- Subtract the feeding cost (which depends on how many animals you own and what kind) every morning.
- Have a random event take place during each turn and perform those bank transactions correctly.

- Animals that aren't "born" in the zoo start at age 3 and this age is incremented every day. Animals younger than 3 (animals born in the exhibit) cost or earn the zoo 2 times their regular amount.
- Make sure that the zookeeper doesn't go bankrupt and continues to play.
- Reprompt user at the end of each day (at the end of the game play for loop) to keep playing until they lose or they wish to end the game.
- No memory leaks

Extra Credit (15 pts):

New Animal: (5 pts)
Add a new animal class to the game with all the required elements for age, cost, babies, average food cost, and payoff. Alternatively, allow the user to dynamically create a new animal during runtime, prompting them for each trait.

Status messages: (5 pts)
Use your knowledge of ifstream to output a random message from a text file for each event.
Example (During an attendance boom):
"Today is National Tiger Day! Tigers generate money today! You made: 360 extra dollars"

Different Feed Types: (5 pts)
Allow the user to pick between three different types of feed.
Generic: Base case, behaves normally.
Premium: Twice as expensive for all animals, sickness becomes half as likely to occur.
Cheap: Half as expensive for all animals, sickness becomes twice as likely to occur.

(10 pts) **Program Style/Comments**
In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!
http://classes.engr.oregonstate.edu/eecs/spring2016/cs162-001/162_style_guideline.pdf

```
/*********************************************************
** Program: zoo_driver.cpp
** Author: Your Name
** Date: 05/06/2016
** Description:
** Input:
** Output:
*********************************************************/
```

Electronically submit your C++ program (**.h, .cpp, and Makefile files**, not your executable!!!) and your test files as a tarred archive by the assignment due date, using TEACH.

You must tar these files together using the following command:

```
tar –cvf assign3.tar file1 file2 file3 …
```

****NOTE:** The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See: http://engineering.oregonstate.edu/computing/fileaccess/