**Zoo Tycoon**

Design Description

After reading through the project specifications, my initial plan is to create the Animal parent class, then create the 3 child classes Tiger, Penguin and Turtle and test their functionality before moving onto the Zoo class. As there is only going to be one Zoo object created for the game, I thought the Zoo class would be a good place to track a number of variables such as current day, bank balance and any bonusses, along with the current and maximum capacity of each type of animal. Then, after checking the functionality of the Zoo class and before I start looking at how to program the random events, I thought I would try to run-through a day in the Zoo without any random events happening. I feel that programming the random events will be one of the more challenging aspects of this project, so I would like to get all other modules of the game up and running before introducing the random event aspect to the game.

I plan to split the game flow i.e. the day at the zoo, into three separate modules so I can get each aspect up and running before moving onto the next part i.e. start\_day, zoo\_day and end\_day functions.

When programming the random event function, I first will test the function using cout messages to check that the function works before coding each random event. I plan to compile and test each random event after I write the code, then when it is functional, move on and write the next random event.

After I have written and compiled the various modules of my program, I plan to implement the test cases listed in the following table to detect any runtime issues with my code.

Test table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input Values** | **Driver Functions** | **Expected Outcomes** | **Observed Outcomes** |
| **START OF GAME** | | | | |
| User does not enter an integer when prompted to purchase each animal type at start of game | Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces | Int\_input\_val() function  Do-while loop prompts user to enter an integer | User is prompted to enter an integer until an integer is entered by the user | User is prompted to re-enter an integer until integer is entered. Chars, floats and input containing spaces are rejected |
| User enters a quantity other than 1 or 2 when prompted to purchase 1 or 2 animals of each type at start of game | Input is an integer other than 1 or 2 | Start\_game() function  While loop prompts user to enter 1 or 2 | Loop back to the question prompting user for correct input | Program loops back to the question prompting user to enter 1 or 2, as expected. |
| **ZOO DAY** | | | | |
| Check correct cost of animal is subtracted from bank balance when purchase animals at the start of the game | n/a – cost of each animal is deducted from the starting bank balance in the start\_game() function and displayed under “Current Stats” in the zoo\_day() function | Start\_ game () function  Manually calculate bank balance based on numbers of animals purchased and determine if bank balance matches what is calculated by the function | Bank balance matches what is calculated by the function | Bank balance matches what is calculated by the function, as expected |
| Check animals purchased at the start of the game are added to the zoo | n/a – the quantities of each animal purchased are set in the Zoo object in the start\_game() function and displayed under “Current Stats” in the zoo\_day() function | Start\_ game () function  Check the numbers of each animal listed in Current Stats match those that are purchased at the start of the game | The numbers of each animal listed in Current Stats match those that are purchased at the start of the game | The numbers of each animal listed in Current Stats match those that are purchased at the start of the game, as expected |
| Check feeding costs are calculated correctly for each animal type correctly and subtracted from the bank balance | n/a – the zoo\_day() function calculates the feeding cost for each animal and deducts the total cost from the bank balance then displays feeding cost for each animal and resultant bank balance | Zoo\_day() function  Manually calculate feeding cost based on the numbers of each animal and determine if bank balance matches what is calculated by the function | Feeding costs and bank balance match what is calculated by the function | Feeding costs and bank balance match what is calculated by the function, as expected |
| **RANDOM EVENTS** | | | | |
| Check bonus generated is correct | n/a – random\_event() function calculates a bonus based on the number of tigers and displays it on screen | Random\_event() function  If statements  Manually calculate bonus based on the number of tigers and determine if bonus matches what is calculated by the function | Bonus matches what is calculated by the function | Bonus matches what is calculated by the function, as expected |
| Check if an animal is sick it is removed from the population for that animal type | n/a – random\_event() function reduces the population of the animal type by 1 and sets the ‘dead’ animal back to a default age of 0. The numbers of each animal are displayed under “Current Stats” in the zoo\_day() function | Random\_event() function  While loop  If statements  For loops  Bool variables  Check the numbers of the sick animal type listed in Current Stats match is reduced by 1 | The numbers of the sick animal type listed in Current Stats match is reduced by 1 | The numbers of the sick animal type listed in Current Stats match is reduced by 1, as expected |
| Check animals cannot die if none of that animal type exist in the zoo | n/a – if none of the animal type selected to become sick exist in the zoo the random\_event () function will check the next animal type until it has checked all animal types | Random\_event() function  As a test I placed cout statements within the If statements and While loops to track the actions of the function | If none of the animal type selected exist, the function will check the next animal type. If there are no animals at the zoo the function will display a message to the user | The cout statements placed within the function display that the function behaves as described when none of the animal type selected exist. I removed the cout statements after this test |
| Check each animal type has its specified number of babies and the babies are added to the current population for that animal type | n/a – the random\_event() function increases the population of the animal type by 1, 5 or 10 depending on the animal type selected to have babies. The numbers of each animal are displayed under “Current Stats” in the zoo\_day() function | Random\_event() function  While loop  If statements  For loops  Bool variables  Check the correct number of babies for each animal type is added to the current population listed in Current Stats in the zoo\_day() function | The correct number of babies for each animal type is added to the current population listed in Current Stats in the zoo\_day() function | The correct number of babies for each animal type is added to the current population listed in Current Stats in the zoo\_day() function, as expected |
| Check animals younger than 3 days cannot have babies | n/a – if none of the animal type selected are old enough to have babies the random\_event () function will check the next animal type until it has checked all animal types | Random\_event() function  As a test I placed cout statements within the If statements and While loops to track the actions of the function | If none of the animal type selected are old enough to have babies, the function will check the next animal type. If there are no animals at the zoo old enough to have babies the function will display a message to the user | The cout statements placed within the function display that the function behaves as described when none of the animal type selected are old enough to have babies. I kept several of the cout statements in my code after this test as I felt they conveyed useful information to the user |
| Check animals cannot have babies if none of that animal type exist in the zoo | n/a – if none of the animal type selected to have babies exists in the zoo the random\_event () function will check the next animal type until it has checked all animal types | Random\_event() function  I placed cout statements within the If statements and While loops to track the actions of the function | If none of the animal type selected exist, the function will check the next animal type. If there are no animals at the zoo the function will display a message to the user | The cout statements placed within the function display that the function behaves as described when none of the animal type selected exist. I removed the cout statements after this test |
| After baby animals are born, check the array size doubles when the number of animals exceeds capacity | After animal type to have babies has been chosen, the end\_day () function checks the current size of the animal array and calls the resize function for that animal type if it is not big enough, animals in current array are copied across | I decided to test one of my resize functions,  Resize\_turtle(). To do this, I wrote a print\_info () function of the Animal class which prints out information on an Animal object. I then wrote a print\_turtle () function of the Zoo class which loops through all the turtles in the array and call it before and after my resize\_turtle() function in the random\_event() function | When size of current turtle array is not large enough to hold 10 new turtle babies, the array size doubles and current turtles are copied across | When adding 10 new turtle babies will exceed the size of the current array, the size of current turtle array doubles to make room for 10 new turtle babies and current turtles are copied across, as expected |
| **END OF DAY** | | | | |
| User does not enter an integer when prompted to purchase an animal at end of the day | Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces | Int\_input\_val() function  Do-while loop prompts user to enter an integer | User is prompted to enter an integer until an integer is entered by the user | User is prompted to re-enter an integer until integer is entered. Chars, floats and input containing spaces are rejected |
| User enters an integer other than 1 or 2 when asked whether they would like to purchase an animal at the end of the game | Input is an integer other than 1 or 2 | End\_day() function  While loop prompts user to enter 1 or 2 | Loop back to the question prompting user for correct input | Program loops back to the question prompting user to enter 1 or 2, as expected. |
| User enters a number other than 1, 2 or 3 representing the animal type when prompted to purchase an animal at the end of the day | Input is an integer other than 1, 2 or 3 | End\_day() function  While loop prompts user to enter 1, 2 or 3 | Loop back to the question prompting user for correct input | Program loops back to the question prompting user to enter 1 or 2, as expected. |
| Check correct payoff is being calculated for each animal | n/a – the end\_day() function calculates the payoff for each animal and adds it to the bank balance then displays resultant bank balance | End\_day() function  Manually calculate payoff based on the numbers of each animal and add to bank balance and determine if bank balance matches what is calculated by the function | Payoff and bank balance match what is calculated by the function | Payoff and bank balance match what is calculated by the function, as expected |
| Check correct cost of animal is subtracted from bank balance when purchase an animal | n/a – cost of the animal is deducted from the bank balance in the end\_game() function and resulting bank balance is displayed both after the purchase and in “Current Stats” by the zoo\_day() function | End\_ game () function  Manually calculate resulting bank balance after deducting the cost of the animal purchased and determine if bank balance matches what is calculated by the function | Bank balance matches what is calculated by the function | Bank balance matches what is calculated by the function, as expected |
| Check animal population increases after  purchasing an animal | n/a – the population of the animal purchased is increased by 1 and the new level is set in the Zoo object by the end\_day() function. The updated animal population is displayed under “Current Stats” in the zoo\_day() function | End\_ game () function  If statement depending on type of animal chosen  Check the population of the animal purchased has increased by 1 in the Current Stats | The population of the animal purchased has increased by 1 in the Current Stats | The population of the animal purchased has increased by 1 in the Current Stats, as expected |
| After purchasing an animal, check the array size doubles when the number of animals exceeds capacity | After choosing an animal, the end\_day () function checks the current size of the animal array and calls the resize function if it is not big enough, animals in current array are copied across | I decided to test one of my resize functions,  Resize\_turtle(). To do this, I wrote a print\_info () function of the Animal class which prints out information on an animal object. I then wrote a print\_turtle () function of the Zoo class which loops through all the turtles in the array and called it before and after my resize\_turtle() function in the random\_event() function | When size of current turtle array is not large enough to hold new turtle, the array size doubles and current turtles are copied into new array | When adding a new turtle will exceed the size of the current array, the size of current turtle array doubles to make room for the new turtle and current turtles are copied across, as expected |
| Check the game will not let you proceed if you don’t have enough money | Change the initial money for the zoo to $30,000 and keep buying tigers to force the bank balance to go under $0 | End\_game() function  If statements  Change value of money data member in default constructor of Zoo object to $30,000 | If the user does not have enough money to continue the game, an error message appears informing the user and the program ends | Bank balance is negative at the end of the day, pressed 1 to continue playing game and error message appeared saying I did not have enough money and game ended as expected |

Reflection

I found this project particularly challenging. There were many aspects I did not consider in my original design. For instance, I did not consider that when an animal’s array size is doubled I would need to copy across the current animals into the new larger array. Furthermore, I also did not consider that in the random events function, if an animal of a certain type was selected to get sick and none of that animal existed at the zoo then the animal could not be removed from the zoo. Before this occurred to me I was getting a negative number for the population of one of my animals.

I feel that my random events function itself is extremely long. Ideally, I should have split the 4 random event options into 4 separate functions which are called from the main random event function. I had a lot of debugging issues with this function and due to its size it made finding the issues more challenging. If I had split it up into separate functions, it would have been a lot easier to debug and it would be much simpler to understand what was happening in each scenario.