Chase Verbout
CS202 Spring 2022
Program 1

Efficiency Write-Up

A. **Discuss the effectiveness of your design and classes**
    a. The solution I came up with is effective in completing the given task. Utilizing a single hierarchy system stemming from the Animal class. My specialized animals, cheetah and dolphin were derived once more into node versions of the respective classes and the gorilla animal was not because it is used by a vector later. The contest class also incorporates a single hierarchy system that ensures the respective versions of races have the same initial info and functions.
B. **Discuss the validity of your approach**
    a. Logical for what we have been taught, sure. Making the 3 animals a part of a hierarchy was a valid approach. The same goes for the contests. However, with the instructed data structures I don't think it makes logical sense to use 3 different versions for different races. But we needed to fit them in somewhere and the only place they seem to fit is there. I think a more valid approach would have been to use the CLL for each one and incorporate generic programming into the mix.
C. **Analyze your results in terms of object oriented programming methodologies**
    a. My resulting program follows the layout of OOP well. There are no structs and there are no public data structures. Everything is encapsulated well and my makeshift client interface is able to interact with the functions given as well as the returned outputs. None of my functions are void return calls and my classes implement copy constructors/assignment operators when working with dynamic memory. The same goes for initialization lists when needed.
D. **What major changes did you have to make in your design and explain why**
    a. My first thought with this project was to have an Animal node that could point to any of the specialized animals. After a week of class we learned that hierarchy just does not work like that. My response was to make the animals have node functionality but I was informed that this could have complications later I don't expect so I changed once more into derived node classes which did work well and is a good practice within OOP.
E. **Describe the efficiency of the approach and the efficiency of the resulting code**
    a. The code is efficient for what I asked of it. The CLL works very well for a competition like this that I've made to operate in rounds. It can circle through and get each animal's turn recursively. The ARR of LLL works just fine and is efficient in it's own right but for what I need it to do in a competition it doesn't make as much sense. I made it into a hash table to make the dispersion of dolphins even but I need to access every dolphin every time so using an ARR of LLL is a weird structure since I do not utilize the log n runtime it provides on most uses. The vector is the clear winner as far as readability and simplicity of code goes.

F. How well did the data structure perform for the assigned application?

a. As stated above the CLL works great for an unorganized set of data that needs to be dynamic in size. The ARR of LLL is not really efficient specifically for this program, it can be used correctly, but we gain nothing by using it in this instance. It actually takes up more space than the CLL because of the ARR that is created upon initialization each time so why use an ARR LLL if we don't need direct access to a specific animal. I could have implemented a remove by name but the amount of time cut that short.

G. Would a different data structure work better? Which one and why...

a. The ARR aspect was unnecessary - could have just been a LLL that adds at the front. We really just need a structure that lets us access each animal quickly and grows dynamically. Unless we need search functionality we don't need the ARR that takes up more memory.

H. Were there classes that had clear responsibilities?

a. Yes, the nodes emphasize this well and so do the respective animals and contests. However, it does feel repetitive as I've said and that is a result of not knowing how templates work as well as my own interpretation of the instructions.

GDB Write-Up

A. Discuss the effectiveness of the debugger

a. GDB was great. It actually did save me several times. Valgrind doesn't always give the most clear errors but it does let you know which functions the problem is occurring from. With just the function name or line I'm able to make a break and follow the function trail. One example from this program was through making copy constructors and destructors I was receiving conditional jumps as well as invalid reads and writes from my dynamic classes. So I used GDB to better visualize what my program is calling during these functions as well as to verify that my initialization lists and kickstarted constructors were being called as I intended.

B. Discuss what problems it helped you solve

a. A more specific example of what GDB helped me figure out was in working with my nodes and data structures. I found that my copy constructor was causing errors in compilation. I was able to make sure that when one node was copy constructed it was pointing to a different position in memory. This was not the case initially but see that the address was the same when printing both nodes in the command line of GDB. From this it was an easy fix.

C. Discuss features that you would like to learn about so that you could use them the next time you program

a. I would really appreciate the ability to reset my program after I've run through it rather than exiting and reloading it. Just a small thing but it feels much less fluid. I also wonder about being able to undo a step or next call.