**Lab4: Polymorphism**

**Name: Proggaparomita Singha**

**Part1: 50 pts**

Complete the polymorphism project on which we have worked and replace the screenshots below with yours.

1. Sequence of constructor calls

Text

Description automatically generated with medium confidence

A picture containing text, plaque

Description automatically generated

1. Display the information of all the employees and the number of employees

A picture containing timeline

Description automatically generated

1. Display the sequence of the destructor call

Timeline

Description automatically generated with medium confidence

Timeline

Description automatically generated with low confidence

1. Fill out the self-scoring rubric below.

|  |  |  |
| --- | --- | --- |
| Screenshot | Points Assigned | Points Earned |
| Display the sequence of constructors | 10 | 10 |
| Display all the employees | 20 | 20 |
| Display the number of employees | 20 | 20 |
| Display the sequence of destructors | 10 | 10 |
| Total | 60 | 60 |

**Part2: 40 pts (5 pts each)**

1. Based on the outputs above, why do this of the Employee and the S\_Employee or H\_Employee have the same value? **When S\_Employee’s or H\_Employee’s constructor is called, the constructor also calls Employee’s constructor. Since Employee’s constructor is called while an instance of S\_Employee or H\_Employee is being made, the Employee becomes a member of the S\_Employee or H\_Employee object being made, and since the value was found using the keyword “this”, “this” returns the address of the object which is why they have the same address since they are a part of the same object.**
2. What is the size of Employee, S\_Employee, and H\_Employee, respectively? Add a screenshot.

Text

Description automatically generated

1. Are the sizes of S\_Employee and H\_employee the same? If so, why do you think so? **Yes, they are. H\_Employee has the 148 bytes inherited from Employee plus the 12 bytes from its double and integer attributes, yet S\_Employee has the 148 bytes plus its double attribute and although it should come to 156 bytes in total, instead it comes to 160 bytes. I think there might be some kind of padding going on that adds 4 bytes to S\_Employee’s class size to make sure that the bytes align nicely in the RAM which is why both S\_Employee’s and H\_Employee’s sizes are the same.**
2. Do neighboring objects occupy the contiguous memories? Explain your answer based on the outputs of your program. **They do not. For example, the difference between the address of the last H\_Employee object and the 2nd to last H\_Employee object is 168 bytes even though the size of H\_Employee is 160 bytes, and the difference in address between the first S\_Employee object and the 2nd S\_Employee object is 24,008 bytes even though the size of S\_Employee is 160 bytes. This is probably due to the fact that each object was created individually in the heap instead of all together like in one array.**
3. When you implement polymorphism in C++, why do you need to make the destructor of the base class virtual? **The destructor must be virtual in the base class in order to delete the member variables in the child class. If the destructor isn’t virtual, then when the object of the child class is deleted through the pointer of the same type as the parent class, only the member variables of the parent class will be deleted.**
4. (T/**F**) A child class must override the virtual function inherited from its parent class **False, this is not *necessary*. You only override if you want to customize the inherited virtual function, or if you inherit a pure virtual function and you don’t want the child class to become an abstract class.**
5. (T/**F**) You can use an object to call a virtual function in order to implement polymorphism in C++. **To implement polymorphism, the pointer of the same type as the parent class, has to call the virtual function.**
6. Why does polymorphism work only when public inheritance is used to define a child class? **Inherited member functions of the base class can’t be accessed through main() if the child is defined using private or protected inheritance. Polymorphism requires a pointer of the same type as the base class pointing to an object of the child class to be used to call inherited virtual member functions of the child class through main().**