OBJECT-ORIENTED PROGRAMMING
Winter Semester Academic Year 2021-2022 Exercise 3

The goal of this exercise is to practice the use of the new and delete operators, copy constructor functions, the 'this' pointer, inheritance, and virtual functions. We have a community of creatures: good creatures and bad creatures. The creatures live next to each other in specific positions. The creatures possess the following characteristics: • Each creature has a name.
• Each creature has a lifespan that it is expected to live. Initially, when born, it has a lifespan of L, with L > 0 (L is common for all creatures).
• When a creature's lifespan reaches zero, it becomes a zombie. The following actions have no effect on creatures in the zombie state.
• A creature can be cloned in the position of another creature. In this case, the other creature is destroyed, and its position is taken by a new creature created as a copy of the first one.
• Each creature can either be blessed or beaten. In the first case, its lifespan increases by one unit. In the second case, it decreases by one unit. Good creatures, when blessed and sufficiently resilient, are cloned by taking the next position in the community (the next position after the last one wraps around to the first). They are considered "resilient" if their current lifespan is greater than a threshold (good threshold) common for all good creatures. Bad creatures, when blessed and sufficiently resilient, are cloned by occupying all their consecutive next positions in the community that are occupied by creatures in a zombie state (except if the bad creature being blessed is at the end). They are considered "resilient" if their current lifespan is greater than a threshold (bad threshold) common for all bad creatures. At each moment, a creature is randomly chosen, and a random action is applied to it, either blessing or beating. This process is repeated M times. It is possible for the same creature to be selected more than once. Implement the above in C++. For implementation, use the following classes.

**Creature Class:**
This class stores:
- The name of the creature
- The expected lifespan of the creature

Its behavior is characterized by:
- Initially, its lifespan is equal to L, it is not in a zombie state, it belongs to a specific position in a community, and it takes its name.
- It is checked if it is a zombie by verifying if its lifespan is equal to 0 (is a zombie).
- It is checked if it's a good creature or not (is a good), deferring the check to derived classes.
- It can be cloned, replacing another creature, destroying it, and creating a copy to take its position in the community.
- It can be beaten, reducing its lifespan by one unit if it's not in a zombie state.
- It can be blessed, increasing its lifespan by one unit if it's not in a zombie state.

**Good Creature Class:**
This class specializes the creature class with the following:

- When blessed, in addition to its default behavior as a creature, if it's sturdy, it is cloned to the next position.
- The "is a good" check returns true.

**Bad Creature Class:**
This class specializes the creature class with the following:
- When blessed, in addition to its default behavior as a creature, if it's sturdy, it is cloned to all consecutive positions occupied by creatures in zombie state, starting from its own position (unless it's at the end).
- The "is a good" check returns false.

**Creature Society Class:**
This class stores:
- The number of creature positions it has
- An indicator of the initial position (sequence) of the community

Its behavior includes:
- Initially creating N creature positions occupied by N creatures randomly determined as good or bad.
- Beating position i, affecting the creature in that position.
- Blessing position i, affecting the creature in that position.
- Cloning the creature in position i to the next position (clone next).
- Cloning the creature in position i to all consecutive positions from i onward that are occupied by creatures in zombie state, except if i corresponds to the last position (clone zombies).
- Returning the number of good creatures in the community (number of good).
- Returning the number of creatures in the community that are in a zombie state (number of zombies).

**Implement a Main Function:**
Implement a main function that:
- Initially creates a community with N creatures.
- Then, randomly selects a position and applies a random action (blessing or beating) to the creature in that position. Repeat this process M times.
- Afterwards, if the community consists only of good creatures, print the message "Good Dominates in the World!". If the community consists of creatures in a zombie state, print the message "This is a dead society". Otherwise, print the message "Try again to improve the world".

**General Observations:**
The values N, M, L, good thrsh, and bad thrsh should be provided from the command line when calling the program, in that order.
Relative messages should be printed from the functions bless, beat, clone, as well as from the construction and destruction functions.
The names of the creatures should be retrieved from an array of names for good creatures and an array of names for bad creatures, creating unique names by appending the number corresponding to the position the creature occupies in the community.

For implementation, you can define additional data types and enhance the above with data or functions where needed. Depending on your approach, you must define appropriate member functions and data members. Alternatively (but not mandatory) for the above, you can consider:
- The community contains an indicator of a sequence of "positions" it contains. A position is defined as a data structure containing a pointer to the creature that occupies it, a pointer to the community it belongs to, and the index of the position in the community's sequence. Each creature has a pointer to the position that contains it, providing access to the community and the position's index.
- The community contains a sequence of pointers to creatures, and creatures contain a pointer to the community they belong to. Then, the position's index of a creature is obtained by traversing the elements of the sequence (until finding the pointer to the examined creature).