Đã bắt đầu vào lúc	Thứ tư, 27 Tháng chín 2023, 1:13 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ tư, 27 Tháng chín 2023, 1:51 PM
Thời gian thực hiện	38 phút 29 giây
Điểm	0,90/1,00
Điểm	**9,00** của 10,00 (**90**%)

Câu hỏi 1

Đúng một phần

Điểm 0,90 của 1,00

Sở thú kì diệu đang muốn lưu lại danh sách thông tin của các con vật. Lớp Animal sẽ là lớp biểu diễn một loài vật, trong sở thú đang c
Cat và Dog được kế thừa từ lớp Animal.

- **Class Animal** có 2 thuộc tính là **name** và **age**. Phương thức **clone()** dùng để tạo ra 1 bản sao của đối tượng
- **Class Cat** được kế thừa từ **Class Animal** và có thêm thuộc tính **hometown**, đồng thời ghi đè lại phương thức **clone** của **Class Anim**
- **Class Dog** được kế thừa **Class Animal** và có thêm thuộc tính **weight**, **height**, đồng thời ghi đè phương thức **clone** của **class Anima**

Sở thú quản lí danh sách các con vật bằng danh sách liên kết **ZooLinkedList** với phương thức **print()** để in ra thông tin của danh sách
node trong danh sách liên kết chứa thông tin về các con vật của sở thú kì diệu. **Class Node** có thuộc tính là **Animal* creature** lưu trữ
của con vật.

- Phương thức **Node(Animal* creature, Node* next=nullptr)** là copy constructor của **class Node**
- **ZooLinkedList(ZooLinkedList* other)** là copy constructor **class ZooLinkedList**

Lưu ý: Các phương thức trên đã được hiện thực, sinh viên không cần hiện thực lại.

Sinh viên cần hiện thực:

- **Phương thức info()** in ra thông tin ở các lớp Animal, Cat, Dog lần lượt theo các định dạng (có xuống dòng ở cuối):
  - Animal: "Unknown Creature: name, age"
  - Cat: "Cat: name, age, hometown"
  - Dog: "Dog: name, age, weight, height"
- Do mỗi con vật có cách thức để tạo ra khác nhau nên chúng ta phải có thêm **class AnimalFactory** (Tương tự như Factory pattern) v
thức **static create** nhận vào kind là kiểu của con vật (Cat, Dog,...). Ứng với mỗi con vật thì input từ màn hình (từ cin) sẽ ứng với thứ t
của constructor. Và nếu như kind khác với 2 kiểu đã biết là **Cat** và **Dog** thì sẽ tạo ra một đối tượng **Animal** nhận vào **name** và **age**. S
hãy hiện thực **phương thức create của class AnimalFactory**.
- Hiện thực **phương thức addAnimal(string kind)** để thực hiện thêm 1 Node mới vào đầu danh sách liên kết.
- **Overload toán tử +** để thực hiện merge 2 linked list, kết quả trả về là 1 ZooLinkedList mới được **deepcopy.**

Testcase Strategy:

- 15% trên tổng số test case sẽ chỉ kiểm tra **phương thức info().**
- 15% test case tiếp theo sẽ chỉ kiểm tra **phương thức info()** và **phương thức create của class AnimalFactory**.
- 25% test case tiếp theo sẽ chỉ kiểm tra **phương thức addAnimal của class ZooLinkedList**.
- 45% test case còn lại sẽ kiểm tra tất cả các phương thức.

For example:

Test	Input	Result
`Animal *creature = new Animal("Haha",10);` `creature->info();`		`Unknown Creature: Haha, 10`
`Animal *creature = new Cat("Felix",10,"VietNam");` `creature->info();`		`Cat: Felix, 10, VietNam`
`Animal *creature = AnimalFactory::create("Cat");` `creature->info();`	`Felix 1 USA`	`Cat: Felix, 1, USA`

`int n; cin >> n;`
`ZooLinkedList* zoo = new ZooLinkedList();`
`while(n--) {`
`  string kind; cin >>  kind;`
`  zoo->addAnimal(kind);`
`}`
`zoo->print();`
	`4`
	`Cat`
	`Felix 1 VietNam`
	`Mouse`
	`Micky 8`
	`Dog`
	`Albert 1 2 3`
	`Chicken`
	`Chip 2`
		`Unknown Creature: Chip, 2`
		`Dog: Albert, 1, 2, 3`
		`Unknown Creature: Micky, 8`
		`Cat: Felix, 1, VietNam`

`int n; cin >> n;`
`ZooLinkedList* zoo1 = new ZooLinkedList();`
`while(n--) {`
`  string kind; cin >>  kind;`
`  zoo1->addAnimal(kind);`
`}`
`cin >> n;`
`ZooLinkedList* zoo2 = new ZooLinkedList();`
`while(n--) {`
`  string kind; cin >>  kind;`
`  zoo2->addAnimal(kind);`
`}`
`ZooLinkedList *zoo3 = *zoo1 + *zoo2;`
`delete zoo1;`
`delete zoo2;`
`zoo3->print();`
	`2`
	`Cat`
	`Felix 1 VietNam`
	`Dog`
	`Albert 1 2 3`
	`2`
	`Dog`
	`Albert 1 2 3`
	`Cat`
	`Felix 1 VietNam`
		`Dog: Albert, 1, 2, 3`
		`Cat: Felix, 1, VietNam`
		`Cat: Felix, 1, VietNam`
		`Dog: Albert, 1, 2, 3`

Answer: (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
1 class Animal {
2 public:
```

```
public:
    string name;
    int age;
public:
    Animal(string name, int age): name(name), age(age) {};
    virtual Animal* clone() {
        return new Animal(this->name, this->age);
    }

    //TODO: Implement info() function
    virtual void info() {
        cout << "Unknown Creature: " << this->name << ", " << this->age << endl;
    }

};

class Cat : public Animal {
public:
    string hometown;
public:
    Cat(string name,  int age, string hometown) : Animal(name,age) {
        this->hometown = hometown;
    };
    Animal* clone() {
        return new Cat(this->name, this->age, this->hometown);
    }

    //TODO: Implement info() function
    void info() {
        cout << "Cat: " << this->name << ", " << this->age << ", " << this->hometown << endl
    }

};

class Dog: public Animal {
public:
    int weight;
    int height;
public:
    Dog(string name, int age, int weight, int height) : Animal(name,age) {
        this->weight = weight;
        this->height = height;
    };
    Animal* clone() {
        return new Dog(this->name, this->age, this->weight,this->height);
    };

    //TODO: Implement info() function
```

```
    //TODO: Implement info() function
    // Dog: "Dog: name, age, weight, height"
    void info() {
        cout << "Dog: " << this->name << ", " << this->age << ", " << this->weight << ", " <
    }

};

//Factory pattern
class AnimalFactory {
public:
    static Animal* create(string kind) {
        //TODO: Implement method to return Object due to kind of Animal
        if (kind == "Cat") {
            string name;
            int age;
            string hometown;
            cin >> name >> age >> hometown;
            return new Cat(name, age, hometown);
        }
        else if (kind == "Dog") {
            string name;
            int age;
            int weight,height ;
            cin >> name >> age >> weight >> height;
```

	Test	Input	Expected	Got
✔	`Animal *creature = new Animal("Haha",10);` `creature->info();`		`Unknown Creature: Haha, 10`	`Unknown Creature: Haha, 10`
✔	`Animal *creature = new Cat("Felix",10,"VietNam");` `creature->info();`		`Cat: Felix, 10, VietNam`	`Cat: Felix, 10, VietNam`
✔	`Animal *creature = AnimalFactory::create("Cat");` `creature->info();`	`Felix 1 USA`	`Cat: Felix, 1, USA`	`Cat: Felix, 1, USA`

	Test	Input	Expected	Got

✔	`int n; cin >> n;`
	`ZooLinkedList* zoo = new ZooLinkedList();`
	`while(n--) {`
	`  string kind; cin >>  kind;`
	`  zoo->addAnimal(kind);`
	`}`
	`zoo->print();`
		4
		Cat
		Felix 1
		VietNam
		Mouse
		Micky 8
		Dog
		Albert 1 2 3
		Chicken
		Chip 2
			Unknown Creature:
			Chip, 2
			Dog: Albert, 1, 2, 3
			Unknown Creature:
			Micky, 8
			Cat: Felix, 1,
			VietNam
				Unknown Creature:
				Chip, 2
				Dog: Albert, 1, 2, 3
				Unknown Creature:
				Micky, 8
				Cat: Felix, 1,
				VietNam

✔	`int n; cin >> n;`
	`ZooLinkedList* zoo1 = new ZooLinkedList();`
	`while(n--) {`
	`  string kind; cin >>  kind;`
	`  zoo1->addAnimal(kind);`
	`}`
	`cin >> n;`
	`ZooLinkedList* zoo2 = new ZooLinkedList();`
	`while(n--) {`
	`  string kind; cin >>  kind;`
	`  zoo2->addAnimal(kind);`
	`}`
	`ZooLinkedList *zoo3 = *zoo1 + *zoo2;`
	`delete zoo1;`
	`delete zoo2;`
	`zoo3->print();`
		2
		Cat
		Felix 1
		VietNam
		Dog
		Albert 1 2 3
		2
		Dog
		Albert 1 2 3
		Cat
		Felix 1
		VietNam
			Dog: Albert, 1, 2, 3
			Cat: Felix, 1,
			VietNam
			Cat: Felix, 1,
			VietNam
			Dog: Albert, 1, 2, 3
				Dog: Albert, 1, 2, 3
				Cat: Felix, 1,
				VietNam
				Cat: Felix, 1,
				VietNam
				Dog: Albert, 1, 2, 3

Testing was aborted due to error.

Show differences

Question author's solution (Cpp):

```cpp
class Animal {
public:
    string name;
    int age;
public:
    Animal(string name, int age): name(name), age(age) {};
    virtual Animal* clone() {
        return new Animal(this->name, this->age);
```

```
    }
    //TODO: Implement info() function
    virtual void info(){
        cout << "Unknown Creature: " << this -> name << ", " << this -> age << endl;
    }
};

class Cat : public Animal {
public:
    string hometown;
public:
    Cat(string name,  int age, string hometown) : Animal(name,age) {
        this->hometown = hometown;
    };
    Animal* clone() {
        return new Cat(this->name, this->age, this->hometown);
    }
    //TODO: Implement info() function
    void info(){
        cout << "Cat: " << this -> name << ", " << this -> age << ", " << this -> hometown
    }
};

class Dog: public Animal {
public:
    int weight;
    int height;
public:
    Dog(string name, int age, int weight, int height) : Animal(name,age) {
        this->weight = weight;
        this->height = height;
    };
    Animal* clone() {
        return new Dog(this->name, this->age, this->weight,this->height);
    };
    //TODO: Implement info() function
    void info(){
        cout << "Dog: " << this -> name << ", " << this -> age << ", " << this -> weight <<
    }
};

class AnimalFactory {
public:
    static Animal* create(string kind) {
        //TODO: Implement method to create Object due to kind of Animal
        if (kind == "Cat"){
            string name;
```

```
            int age;
            string hometown;
            cin >> name >> age >> hometown;
            return new Cat(name, age, hometown);
        } else if (kind == "Dog"){
            string name;
            int age, weight, height;
            cin >> name >> age >> weight >> height;
            return new Dog(name, age, weight, height);
        }
        string name;
        int age;
        cin >> name >> age;
        return new Animal(name, age);
    }
};

class Node {
public:
    Animal* creature;
    Node* next;
public:
    Node(): creature(nullptr), next(nullptr) {};
    Node(Animal *creature, Node* next=nullptr) {
        this->creature = creature;
        this->next = next;
    }
};

class ZooLinkedList {
public:
    Node *head;
public:
    ZooLinkedList(): head(nullptr){};
    ZooLinkedList(ZooLinkedList* other);
    void print();

    void addAnimal(string kind) {
        //TODO: Write function to add new Animal
        Animal *newAnimal = AnimalFactory::create(kind);
        Node *newNode = new Node(newAnimal, head);
        head = newNode;
    }
    ZooLinkedList* operator+ (ZooLinkedList& other) {
        //TODO: OVERLOADING OPERATOR + TO MERGE 2 LINKED LIST
        //return new Zoolinkedlist that merge 2 list
        ZooLinkedList* result = new ZooLinkedList();
```

```
        ZooLinkedList* result = new ZooLinkedList();
        Node* currNode = head;
        Node* lastNode = nullptr;

        // Sao chép danh sách liên kết thứ nhất vào danh sách kết quả
        while (currNode != nullptr) {
            Node* newNode = new Node(currNode->creature->clone());
            if (result->head == nullptr) {
                result->head = newNode;
                lastNode = newNode;
            }
            else {
                lastNode->next = newNode;
                lastNode = newNode;
            }
            currNode = currNode->next;
        }

        // Sao chép danh sách liên kết thứ hai vào danh sách kết quả
        currNode = other.head;
        while (currNode != nullptr) {
            Node* newNode = new Node(currNode->creature->clone());
            if (result->head == nullptr) {
                result->head = newNode;
                lastNode = newNode;
            }
            else {
                lastNode->next = newNode;
                lastNode = newNode;
            }
            currNode = currNode->next;
        }

        return result;
    }
};
```

Đúng một phần

Điểm cho bài nộp này: 0,90/1,00.