

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 có 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 Animal**
- **Class Dog** được kế thừa **Class Animal** và có thêm thuộc tính **weight**, **height**, đồng thời ghi đè phương thức **clone** của **class Animal**

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) với phương 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 tự 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**. Sinh viên 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
<pre>Animal *creature = new Animal("Haha",10); creature->info();</pre>		Unknown Creature: Haha, 10
<pre>Animal *creature = new Cat("Felix",10,"VietNam"); creature->info();</pre>		Cat: Felix, 10, VietNam
<pre>Animal *creature = AnimalFactory::create("Cat"); creature->info();</pre>	Felix 1 USA	Cat: Felix, 1, USA
<pre>int n; cin >> n; ZooLinkedList* zoo = new ZooLinkedList(); while(n--) { string kind; cin >> kind; zoo->addAnimal(kind); } zoo->print();</pre>	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
<pre>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();</pre>	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:
```

```

1  ~~~~~
2  public:
3      string name;
4      int age;
5  public:
6      Animal(string name, int age): name(name), age(age) {};
7  virtual Animal* clone() {
8      return new Animal(this->name, this->age);
9  }
10
11     //TODO: Implement info() function
12 virtual void info() {
13     cout << "Unknown Creature: " << this->name << ", " << this->age << endl;
14 }
15
16 };
17
18 class Cat : public Animal {
19 public:
20     string hometown;
21 public:
22     Cat(string name, int age, string hometown) : Animal(name,age) {
23         this->hometown = hometown;
24     };
25     Animal* clone() {
26         return new Cat(this->name, this->age, this->hometown);
27     }
28
29     //TODO: Implement info() function
30 void info() {
31     cout << "Cat: " << this->name << ", " << this->age << ", " << this->hometown << endl;
32 }
33
34 };
35
36 class Dog: public Animal {
37 public:
38     int weight;
39     int height;
40 public:
41     Dog(string name, int age, int weight, int height) : Animal(name,age) {
42         this->weight = weight;
43         this->height = height;
44     };
45     Animal* clone() {
46         return new Dog(this->name, this->age, this->weight, this->height);
47     };
48
49     //TODO: Implement info() function

```

```

49 //TODO: Implement info() function
50 // Dog: "Dog: name, age, weight, height"
51 void info() {
52     cout << "Dog: " << this->name << ", " << this->age << ", " << this->weight << ", " << this->height << endl;
53 }
54
55 };
56
57 //Factory pattern
58 class AnimalFactory {
59 public:
60     static Animal* create(string kind) {
61         //TODO: Implement method to return Object due to kind of Animal
62         if (kind == "Cat") {
63             string name;
64             int age;
65             string hometown;
66             cin >> name >> age >> hometown;
67             return new Cat(name, age, hometown);
68         }
69         else if (kind == "Dog") {
70             string name;
71             int age;
72             int weight,height ;
73             cin >> name >> age >> weight >> height;
74             return new Dog(name, age, weight, height);
75         }
76     }
77 }

```

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

Testing was aborted due to error.

Show differences

Question author's solution (Cpp):

```
1 class Animal {
2 public:
3     string name;
4     int age;
5 public:
6     Animal(string name, int age): name(name), age(age) {};
7     virtual Animal* clone() {
8         return new Animal(this->name, this->age);
```

```

9      }
10     //TODO: Implement info() function
11     virtual void info(){
12         cout << "Unknown Creature: " << this -> name << ", " << this -> age << endl;
13     }
14 };
15
16 class Cat : public Animal {
17 public:
18     string hometown;
19 public:
20     Cat(string name, int age, string hometown) : Animal(name,age) {
21         this->hometown = hometown;
22     };
23     Animal* clone() {
24         return new Cat(this->name, this->age, this->hometown);
25     }
26     //TODO: Implement info() function
27     void info(){
28         cout << "Cat: " << this -> name << ", " << this -> age << ", " << this -> hometown
29     }
30 };
31
32 class Dog: public Animal {
33 public:
34     int weight;
35     int height;
36 public:
37     Dog(string name, int age, int weight, int height) : Animal(name,age) {
38         this->weight = weight;
39         this->height = height;
40     };
41     Animal* clone() {
42         return new Dog(this->name, this->age, this->weight, this->height);
43     };
44     //TODO: Implement info() function
45     void info(){
46         cout << "Dog: " << this -> name << ", " << this -> age << ", " << this -> weight <<
47     }
48 };
49
50 class AnimalFactory {
51 public:
52     static Animal* create(string kind) {
53         //TODO: Implement method to create Object due to kind of Animal
54         if (kind == "Cat"){
55             string name:

```

```

55         string name;
56         int age;
57         string hometown;
58         cin >> name >> age >> hometown;
59         return new Cat(name, age, hometown);
60     } else if (kind == "Dog"){
61         string name;
62         int age, weight, height;
63         cin >> name >> age >> weight >> height;
64         return new Dog(name, age, weight, height);
65     }
66     string name;
67     int age;
68     cin >> name >> age;
69     return new Animal(name, age);
70 }
71 };
72
73 class Node {
74 public:
75     Animal* creature;
76     Node* next;
77 public:
78     Node(): creature(nullptr), next(nullptr) {};
79     Node(Animal *creature, Node* next=nullptr) {
80         this->creature = creature;
81         this->next = next;
82     }
83 };
84
85 class ZooLinkedList {
86 public:
87     Node *head;
88 public:
89     ZooLinkedList(): head(nullptr){};
90     ZooLinkedList(ZooLinkedList* other);
91     void print();
92
93     void addAnimal(string kind) {
94         //TODO: Write function to add new Animal
95         Animal *newAnimal = AnimalFactory::create(kind);
96         Node *newNode = new Node(newAnimal, head);
97         head = newNode;
98     }
99     ZooLinkedList* operator+ (ZooLinkedList& other) {
100         //TODO: OVERLOADING OPERATOR + TO MERGE 2 LINKED LIST
101         //return new ZooLinkedList that merge 2 list
102         ZooLinkedList* result = new ZooLinkedList();

```



```

102     ZOOLinkedList* result = new ZOOLinkedList();
103     Node* currNode = head;
104     Node* lastNode = nullptr;
105
106     // Sao chép danh sách liên kết thứ nhất vào danh sách kết quả
107     while (currNode != nullptr) {
108         Node* newNode = new Node(currNode->creature->clone());
109         if (result->head == nullptr) {
110             result->head = newNode;
111             lastNode = newNode;
112         }
113         else {
114             lastNode->next = newNode;
115             lastNode = newNode;
116         }
117         currNode = currNode->next;
118     }
119
120     // Sao chép danh sách liên kết thứ hai vào danh sách kết quả
121     currNode = other.head;
122     while (currNode != nullptr) {
123         Node* newNode = new Node(currNode->creature->clone());
124         if (result->head == nullptr) {
125             result->head = newNode;
126             lastNode = newNode;
127         }
128         else {
129             lastNode->next = newNode;
130             lastNode = newNode;
131         }
132         currNode = currNode->next;
133     }
134
135     return result;
136 }
137 };

```

Đúng một phần

Điểm cho bài nộp này: 0,90/1,00.

