

1. Визначте функцію "sayName ()" для "Mother class ", яка виводить на екран "I'm a mother".

```
void _____::sayName() {  
    _____ << "I'm a mother";  
}
```

2. Введіть відсутнє ключове слово, щоб успадкувати "Daughter" від "Mother".

```
class Daughter : _____ Mother  
{  
};
```

3. Введіть символ, що відсутній, щоб успадкувати похідний клас від базового класу.

```
class Derived _ public Base  
{  
};
```

4. Публічні члени класу доступні ...

- ☐ ... для всіх
- ☐ ... лише для функцій-членів похідного класу
- ☐ ... лише для функцій-членів класу

5. Що означає protected?

- ☐ члени доступні для всіх
- ☐ члени доступні похідним членам класу
- ☐ члени приватні

6. Визначте конструктор і деструктор для класу "Mother".

```
_____::Mother() {  
    cout << "constructor" << endl;  
}  
Mother::~_____() {  
    cout << "destructor" << endl;  
}
```

7. Оголосіть protected елементу Base class та отримайте до нього доступ з функції "foo" похідного класу

```
_____ Base {  
    _____ :  
    int baseVar;  
};  
class Derived : _____ Base {  
public:  
    void foo() {  
        baseVar = 12;  
    }  
};
```

8. Оголосіть клас "B" з власним конструктором і клас "D" зі своїм конструктором, де "D" спадкує "B".

```
_____ B {  
public:  
    _____ () {  
        cout << "B's constructor";  
    }  
};  
class D : _____ B {  
    _____ () {  
        cout << "D's constructor";  
    }  
};
```

9. Якщо клас "D" успадковує клас "B", коли створюється об'єкт класу "D"

- ... конструктор " D " викликається перед конструктором " B "
- ... конструктор " B " викликається перед конструктором " D "
- ... викликається лише конструктор " D "

10. Якщо клас "D" походить від класу "B", коли об'єкт "D" знищується ...

- ... деструктор базового класу викликається перед деструктором " D "
- ... деструктор класу "D" викликається перед деструктором його бази