**Lý thuyết:**

**Câu 1: Khi phân tích mối quan hệ lớp điểm và lớp hình tròn, ta có thể xác định:**

Lớp hình tròn kết thừa lớp điểm vì hình tròn được cấu tạo từ các điểm. Mà khi nhắc đến kế thừa ta sẽ nghĩ đến những thuộc tính mà lớp cha có, còn lớp con thì không tự có mà nó chỉ có khi được kế thừa từ lớp cha. Quay lại với hình tròn và điểm, ta thấy hình tròn không thể tự sinh ra mà nó được “kế thừa” từ vố số điểm nên ta coi hình tròn là một lớp con và điểm là một lớp cha. (A)

**Câu 2: Tính chất kế thừa dạng private là:**

Thành phần protected, public của lớp cha sẽ trở thành private của lớp con vì khi test chương trình ta thấy vậy. (C)

**Câu 3: Tính chất kế thừa dạng protected là:**

Thành phần protected, public ở lớp cha tương ứng trở thành thành phần protected ở lớp con. (A)

**Câu 4: Tính chất kế thừa dạng public là**:

Thành phần public ở lớp cha trở thành public ở lớp con và thành phần protected ở lớp cha trở thành protected ở lớp con. (C)

**Câu 5: Câu lệnh khai báo tại dòng số 5 cho biết:**

class Diem { (1)

private:

int x,y; (2)

public: (3)

…..

};(4)

class Hinhtron: public Diem (5) {

private: r:real;(6)

};

Khai báo class Hinhtron kế thừa public class Diem. (A)

**Câu 6: Cho lớp B kế thừa lớp A, để truy cập vào thành phần private x, y của lớp A thì ta chọn kiểu kế thừa:**

Không thể kế thừa vì private làm cho các thuộc tính x, y tại lớp A chỉ có thể truy cập trong phạm vi nội bộ lớp. (A)

**Câu 7: Cho lớp A kế thừa lớp B. Cả hai lớp đều có phương thức Xuat(). Nếu khai báo đối tượng object thuộc lớp A khi gọi đến phương thức Xuat(object.Xuat()) là gọi đến phương thức của lớp nào.**

Xảy ra lỗi vì phương thức Xuat() bên ngoài không được chỉ ra từ đối tượng cụ thể nào. (C)

**Câu 8: Trong kế thừa nhiều mức có cho phép:**

Cho phép trùng tên và phương thức. (A)

**Câu 9: Ví dụ nào chứng tỏ tính kế thừa**

Xe đạp và xe đạp đua (B)

Điểm và hình tròn (D)

**Câu 10: Trong kế thừa có thể;**

Kế thừa tất cả các phương thức và thuộc tính khai báo trong phần protected, public bao gồm hàm tạo và hàm hủy (B)

Nếu hàm tạo và hàm hủy ở lớp cha và lớp con đều có thì sẽ ưu tiên chạy ở lớp con

**Câu 11: Cho đoạn chương trình sau:**

class Trai\_cay {

private:

char mauvo[5];

char mauhat[3];

char hinhdang[30];

public:

void Hienthi();

};

Cần xây dựng class Trai\_dua\_hau. Quan hệ giữa class Trai\_cay và Trai\_dua\_hau là:

Trai\_cay là lớp cha của Trai\_dua\_hau. (A)