1. Nếu bỏ qua các vấn đề về bộ nhớ và thời gian chạy, liệu có tình huống nào mà vòng lặp là lựa chọn duy nhất để giải quyết mà không thể dùng đệ quy?  
   Không. Tất cả các giải pháp dùng vòng lặp đều có thể được cài bằng đệ quy, mặc dù dùng đệ quy có thể tốn quá nhiều bộ nhớ.

Không có

1. Có tình huống nào mà đệ quy là lựa chọn duy nhất (không thể dùng vòng lặp để cài đặt thuật toán)?  
   Không, cách trình biên dịch xử lý các hàm đệ quy chính là một cách dùng vòng lặp để thay thế tất cả các kiểu đệ quy. Các bạn sẽ học về phương pháp này ở môn Cấu trúc dữ liệu và giải thuật.không có

Không có

1. Nên dùng chiến lược nào hơn, đệ quy hay vòng lặp?

Tùy tình huống, hãy chọn cách nào đơn giản, dễ hiểu, và hiệu quả.

Tuỳ từng trường hợp

1. Cần đề phòng những gì khi viết hàm đệ quy?

Tránh bị lặp vô hạn

1. Không chạy chương trình, hãy tính xem output của lời gọi hàm ex233(6) là gì.

void ex233(int n) {  
    if (n <= 0) return;  
    cout << n << " ";  
    ex233(n-2);  
    ex233(n-3);  
    cout << n << " ";  
}

6 5 4 3 2 1

1. Hàm đệ quy sau có vấn đề gì?

long factorial(int n) {

    long temp = factorial(n-1) \* n;  
    return (n <= 1) ? 0 : temp;

}

1. Cho hàm đệ quy sau:

int mystery(int a, int b) {  
   if (b == 0) return 0;  
   if (b % 2 == 0) return mystery(a+a, b/2);  
   return mystery(a+a, b/2) + a;  
}

Lời gọi hàm mystery(2,25) và mystery(3,11) cho kết quả gì? Thực ra mystery(a,b) tính cái gì với a,b>0?  
Câu hỏi tương tự, nhưng sửa code của mystery thay + bằng \* và thay return 0 bằng return 1.