**Support turn3**

1. **Lý thuyết cơ bản**

Kiểm thử Hộp Trắng (còn gọi là Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing hoặc Structural Testing)

* + Trong kỹ thuật này, mục đích là duyệt qua tất cả các câu lệnh ít nhất một lần. Do đó, mỗi dòng mã đều được kiểm tra => thích hợp với **unit testing**, do kiểm thử integration và system thì khối lượng công việc sẽ vô cùng lớn (xem [Link](https://viblo.asia/p/ky-thuat-kiem-thu-hop-trang-white-box-testing-maGK7MpOlj2))
  + Người kiểm thử phải có kiến thức nhất định về việc mã hoá, cấu trúc bên trong của chức năng, biết lâp trình phần mềm.
  + Việc kiểm thử được tiến hành dựa vào việc kiểm xem giải thuật, mã lệnh đã làm có đúng không.
  + Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code; khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ.

=> **Đây là 1 trong lí do quan trọng làm mình viết unit test theo phương**   **pháp thủ công, chính là để phục vụ cho white box testing**

Câu hỏi đặt ra là tại sao chỉ có 6 test cho phần unit testing, bởi mình đã cân nhắc rất kỹ, các hàm còn lại đều cần đầu ra từ hàm khác, rất khó để test độc lập, nếu cố gắng nó lại thành integration testing, các bạn đọc source code cũng có thể thấy điều dó

* + <https://www.geeksforgeeks.org/basis-path-testing-in-software-testing/>

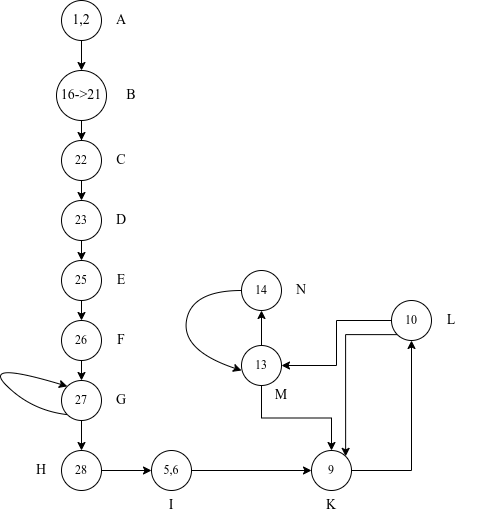
=> Tham khảo thêm link trên đây để hiểu hơn về phần 3

\* Công thức tính độ phức tạp: V(G) = E - N + 2\*P

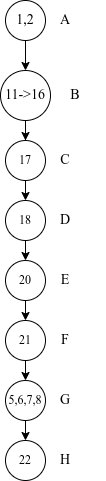
* + *E* = số cạnh của đồ thị.
  + *N* = số nút của đồ thị.
  + *P* = số lượng các [thành phần được kết nối](https://en.wikipedia.org/wiki/Connected_component_(graph_theory)) .

1. **Support**
2. Công việc cần thực hiện
   * Tạo biểu đồ luồng (bám sát vào code để hiểu đồ thị luồng)
   * Xác định độ phức tạp theo chu kỳ của đồ thị luồng
   * Xác định tập cơ bản của các đường đi tuyến tính độc lập
   * Chuẩn bị các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn.
   * So sánh kết quả thực tế của từng trường hợp thử nghiệm với kết quả mong đợi.
3. Xây dựng cho từng test

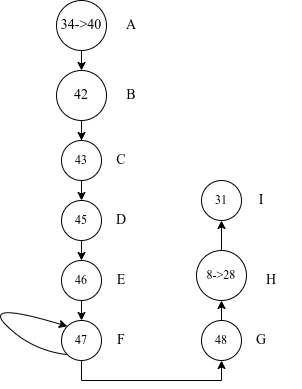
* **Test1:**
  + Biểu đồ luồng (Các số trong hình tròn biểu thị vị trí dòng code)



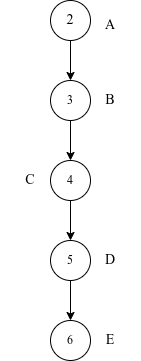
* + Độ phức tạp: V(G) = E - N + 2\*P = 16 – 13 + 2 = 5
  + Xác định tập cơ bản của các đường đi tuyến tính độc lập
    - Path1: A -> B -> C -> D -> E -> F -> G -> G
    - Path2: A -> B -> C -> D -> E -> F -> G -> G -> H -> I -> K -> L -> M -> N
    - Path3: A -> B -> C -> D -> E -> F -> G -> G -> H -> I -> K -> L -> K
    - Path4: A -> B -> C -> D -> E -> F -> G -> G -> H -> I -> K -> L -> M -> N -> M
    - Path5: A -> B -> C -> D -> E -> F -> G -> G -> H -> I -> K -> L -> M -> N -> M -> K
  + Các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn
    - Path1 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, màu (0,128,128)
      * Kết quả mong đợi: Gán màu đen (0,0,0) cho từng phần tử trong grid
    - Path2 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, màu (0,128,128)
      * Kết quả mong đợi: Vẽ 1 đường ngang và 1 đường dọc hiển thị đúng vị trí, kích cỡ, màu sắc
    - Path3 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, màu (0,128,128)
      * Kết quả mong đợi: Vẽ 20 đường hàng ngang cách đều nhau, hiển thị đúng vị trí, kích cỡ, màu sắc
    - Path4 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, màu (0,128,128)
      * Kết quả mong đợi: Vẽ 1 đường ngang và 10 đường dọc cách đều nhau, hiển thị đúng vị trí, kích cỡ, màu sắc
    - Path5 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, màu (0,128,128)
      * Kết quả mong đợi: Các đường thẳng ngang dọc đan xen tạo thành 1 bảng với nhiều ô vuông với kích thước, vị trí và màu sắc mong muốn
  + Kết quả thực tế của từng trường hợp thử nghiệm **không trùng** với kết quả mong đợi => do các đường không cách nhau nên dẫn đến hiện tượng chồng lên nhau (lỗi ở test case 3 và 4 bạn phải thêm tham số block\_size)
* **Test2:**
  + Biểu đồ luồng (Các số trong hình tròn biểu thị vị trí dòng code, 11 ->16 thể hiện từ dòng code 11 đến dòng 16)



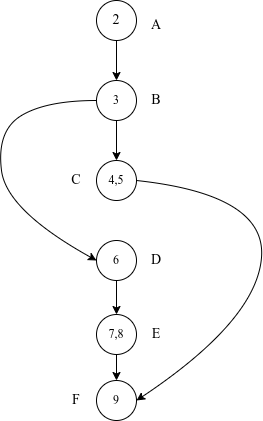
* + Độ phức tạp: V(G) = E - N + 2\*P = 7 – 8 + 2 = 1
  + Xác định tập cơ bản của các đường đi tuyến tính độc lập
    - Path1: A -> B -> C -> D -> E -> F -> G -> H
  + Các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn
    - Path1 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, size chữ = 60, color = (255,255,255) - màu trắng, font chữ = “comicsans” , nội dung hiển thị: **Press Any Key To Play**
      * Kết quả mong đợi: Dòng chữ **Press Any Key To Play** hiển thị trên cửa sổ đúng như các thông số yêu cầu
  + Kết quả thực tế của từng trường hợp thử nghiệm **trùng** với kết quả mong đợi
* **Test3:**
  + Biểu đồ luồng (Các số trong hình tròn biểu thị vị trí dòng code)



* + Độ phức tạp: V(G) = E - N + 2\*P = 9 – 9 + 2 = 2
  + Xác định tập cơ bản của các đường đi tuyến tính độc lập
    - Path1: A -> B -> C -> D -> E -> F -> F
    - Path2: A -> B -> C -> D -> E -> F -> F -> G -> H -> I
  + Các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn
    - Path1 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, size chữ, color, font chữ, nội dung các nhãn muốn hiển thị
      * Kết quả mong đợi: Gán màu đen (0,0,0) cho từng phần tử trong grid
    - Path2 test case:
      * Đầu vào: s\_width = 800, s\_height = 700, play\_width = 300, play\_height = 600, size chữ, color, font chữ, nội dung các nhãn muốn hiển thị
      * Kết quả mong đợi: Các nhãn và hình chữ nhật hiển thị đúng kích cỡ, màu sắc và vị trí mong muốn
  + Kết quả thực tế của từng trường hợp thử nghiệm **trùng** với kết quả mong đợi
* **Test4:**
  + Biểu đồ luồng (Các số trong hình tròn biểu thị vị trí dòng code)



* + Độ phức tạp: V(G) = E - N + 2\*P = 4 – 5 + 2 = 1
  + Xác định tập cơ bản của các đường đi tuyến tính độc lập
    - Path1: A->B->C->D->E
  + Các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn
    - Path1 test case:
      * Đầu vào: file **scores.txt** lưu sẵn giá trị 10 trong đó
      * Kết quả mong đợi: score = 10, với kiểu dữ liệu string
  + Kết quả thực tế của từng trường hợp thử nghiệm **không trùng** với kết quả mong đợi => Kiểu dữ liệu là list không phải string như mong đợi
* **Test5:**
  + Biểu đồ luồng (Các số trong hình tròn biểu thị vị trí dòng code)



* + Độ phức tạp: V(G) = E - N + 2\*P = 6 – 6 + 2 = 2
  + Xác định tập cơ bản của các đường đi tuyến tính độc lập
    - Path1: A -> B -> D -> E -> F
    - Path2: A -> B -> C -> F
  + Các trường hợp kiểm thử bắt đầu thực thi mỗi đường dẫn
    - Path1 test case:
      * Đầu vào: score = 10, nscore = 20, file **scores.txt**
      * Kết quả mong đợi: a = 20
    - Path2 test case:
      * Đầu vào: score = 30, nscore = 10, file **scores.txt**
      * Kết quả mong đợi: a = 30
  + Kết quả thực tế của từng trường hợp thử nghiệm **trùng** với kết quả mong đợi.