HƯỚNG DẪN TỰ TEST BTL SỐ 2 KTLT

- Xin chào mọi người, do BTL số 2 mình không viết testcase được (độ khó của testcase rất lớn) nên mình chỉ cung cấp file tester cho các bạn tự test, cũng như tự kiểm tra code của mình xem có đúng hay không. Đây là hướng dẫn sử dụng tester, vui lòng đọc kỹ trước khi sử dụng.

- **Về cú pháp lệnh:** sẽ có dạng "key_word: cú pháp lệnh" (chi tiết cú pháp ở dưới)

- **Về tên biến:** các bạn có thấy 4 vector trong hàm main chứ? Đó là 4 nơi lưu trữ các biến mà các bạn sẽ dùng, cú pháp tên biến là:

+ Point: p0, p1, p2, …, p99

+ Node: node0, node1, …, node99

+ Path: path0, path1, …, path99

+ Character: char0, char, …, char99

* Sai cú pháp biến là tester sẽ không chạy, lúc đó đừng hỏi mình vì sao tester không chạy nhé

- Các lệnh:

(Trong các lệnh dưới đây, chuỗi point có thể viết dưới dạng x,y hoặc là 1 point_name)

+ *init_point: point_name/point*: thực hiện khởi tạo biến point_name có giá trị là point

Ví dụ: init_point: p0/1,5; init_point p0/p1 là các lệnh hợp lệ

Kết quả: In ra "Success!" nếu init thành công

+ *print_point: point*: in ra nội dung của point theo đúng format

Ví dụ: print_point: p0; print_point 1,5 là các lệnh hợp lệ

Kết quả: In ra chuỗi point theo đúng format

+ *get_distance: point1/point2* (2 chuỗi này tương tự chuỗi point, có thể ở dạng x,y hoặc là 1 point_name): trả về khoảng cách Euclide giữa 2 điểm point1 và point2

Ví dụ: get_distance: 1,5/2,6; get_distance: p1/1,7 là các lệnh hợp lệ

Kết quả: trả về số nguyên duy nhất là khoảng cách giữa 2 điểm

+ *init_node: node_first/point/node_second* (node_first không được là NULL, node_second có thể là NULL hoặc nullptr): khởi tạo 1 node có tên là node_first gồm point và next = node_second

Ví dụ: init_node: node0/2,5/NULL sẽ khởi tạo node0 có Point = (2,5) và next = NULL;
init_node: node1/p0/node0 sẽ khởi tạo node1 có Point = p0 và next = node0;

Kết quả: In ra "Success!" nếu init thành công, "INVALID POINTER!" nếu node_second trùng với node_first

+ *print_node: node_name*: in ra nội dung của node_name theo đúng format

Ví dụ: print_node: node0 sẽ in nội dung của node0

Kết quả: In ra chuỗi node theo đúng format

+ *init_path: path_name*: khởi tạo 1 path có tên là path_name

Ví dụ: init_path: path0 là 1 lệnh hợp lệ

Kết quả: In ra "Success!" nếu init thành công

+ *add_point: path_name/point*: thêm 1 point vào cuối danh sách path_name

Ví dụ: add_point: path0/1,5 sẽ add Point = (1,5) vào path0, add_point: path1/p0 sẽ add point p0 vào path1

Kết quả: In ra "Success!" nếu thêm thành công

+ *print_path: path_name*: in ra nội dung của path_name theo đúng format

Ví dụ: print_path: path0 là 1 lệnh hợp lệ

Kết quả: In ra chuỗi path theo đúng format

+ *get_last_point: path_name*: trả về point cuối của danh sách tên path_name, và in ra point này

Ví dụ: get_last_point: path0 là 1 lệnh hợp lệ

Kết quả: In ra chuỗi của point cuối cùng trong danh sách path_name theo đúng format

+ *call_path_destructor: path_name*: gọi phương thức destructor của Path, destructor này sẽ thu hồi tất cả vùng nhớ được cấp phát từ class Path, xóa mọi thứ có trong path_name

Ví dụ: call_path_destructor: path0 là 1 lệnh hợp lệ

Kết quả: In ra “Success” nếu phương thức path_name.toString() trả về kết quả là "<Path[count:0,length:-1,[]]>", ngược lại in ra “Failed” và ngừng chương trình

*Đối với các bạn thuộc nhóm Đồ án KTLT, các bạn có thêm 2 lệnh nữa (2 lệnh này được đóng cmt trong hàm main(), có chú thích trong file tester)

+ *remove_last: path_name*: remove node cuối cùng của danh sách path_name

Ví dụ: remove_last: path0 là 1 lệnh hợp lệ

Kết quả: In ra “Success!” nếu remove thành công, “Fail to remove last node” nếu không thể remove được (chương trình vẫn chạy tiếp)

+ *concatenate: save_path/first_path/second_path*: nối second_path vào sau first_path và lưu kết quả vào save_path

Ví dụ: concatenate: path0/path1/path2 là 1 lệnh hợp lệ

Kết quả: in ra nội dung của save_path theo đúng format

+ *init_character: char_var/char_name*: khởi tạo char_var với name = char_name

Ví dụ: init_character: char0/”Watson” là 1 lệnh hợp lệ

Kết quả: In ra “Success” nếu init thành công, “Failed” và ngừng chương trình nếu init thất bại

+ *get_character_name: char_var*: trả về giá trị name của character_var

Ví dụ: get_character_name: char0 là 1 lệnh hợp lệ

Kết quả: In ra chuỗi name của character_var

+ *set_character_name: char_var/char_name*: set name của char_var = char_name

Ví dụ: set_character_name: char0/”Watson” là 1 lệnh hợp lệ

Kết quả: không in gì ra màn hình

+ *move_to_point: char_var/point*: thêm point vào cuối danh sách path của char_var

Ví dụ: move_to_point: char0/1,5 sẽ thêm point = (1,5) vào cuối path của char0, move_to_point: char0/p0 sẽ thêm point p0 vào cuối path của char0

Kết quả: In ra “Success” nếu thêm thành công

+ *print_character: char_var*: in nội chung của char_var theo đúng format

Ví dụ: print_character: char0 là 1 lệnh hợp lệ

Kết quả: In ra chuỗi character theo đúng format

+ *call_character_destructor: char_var*: gọi phương thức destructor của Character, destructor này sẽ thu hồi tất cả vùng nhớ được cấp phát từ class Character, xóa mọi thứ có trong char_var

Ví dụ: call_character_destructor: char0 là 1 lệnh hợp lệ

Kết quả: In ra “Success” nếu phương thức char_var.toString() trả về kết quả là “<Character[name:,path:<Path[count:0,length:-1,[]]>]>”, ngược lại in ra “Failed” và ngừng chương trình

+ *get_current_position: char_var:* trả về điểm cuối cùng của danh sách path

Ví dụ: get_current_position: char0 là 1 lệnh hợp lệ

Kết quả: trả về chuỗi string từ method toString của point biểu diễn điểm cuối cùng trong danh sách path;

+ *init_Murderer: “Murderer”/N*: khởi tạo 1 character tên là “Murderer” (tên cố định) và danh sách path có N điểm (mỗi điểm trên 1 dòng) được viết dưới dạng x,y cần được add vào danh sách path thông qua method moveToPoint()

+ *init_Watson: “Watson”/N*: tương tự với hàm init_Murder, khởi tạo 1 character tên Watson

Kết quả: 2 hàm này đều in ra “Success!” nếu init thành công

+ *checkRescue: max_length/max_distance*: kiểm tra xem Watson có giải cứu được cho Sherlock hay không?

Kết quả: gồm 4 dòng
Dòng 1: Ghi theo dạng “Max length = “;

Dòng 2: Ghi theo dạng “Max distance = “;

Dòng 3: Ghi theo dạng “Out distance = “;

Dòng 4: Ghi theo dạng “Can rescue = “ (1 nếu bắt được, 0 trong trường hợp ngược lại)

- Chương trình sẽ kết thúc khi ấn số 0

Lưu ý: Trong hàm main của tester có sử dụng các method trong class Point: int getX() và int getY(), các bạn **PHẢI** thêm 2 hàm này vào (với quyền truy cập public), cú pháp 2 hàm đơn giản:

```
int getX()
{
	return this→x;
}
int getY()
{
	return this→y;
}
```