Một robot cần tìm đường đi ra khỏi mê cung với vị trí bắt đầu và kết thúc của mê cung sẽ được cho trước. Mê cung được biểu diễn bằng một ma trận kích thước m x n, trong đó giá trị tại mỗi ô là 0 hoặc 1 với giá trị 0 là tường của mê cung, không thể đi được; vị trí biểu diễn bởi số 1 là vị trí có thể đi được trong mê cung. Tại một vị trí bất kỳ trong mê cung, robot có thể di chuyển lên, xuống, sang trái, sang phải sao cho các vị trí này là hợp lệ (các ô này phải có giá trị là 1). Cần xác định đường đi của robot từ vị trí bắt đầu đến điểm cuối trong mê cung.

Để giải quyết bài toán trên, cần cung cấp dữ liệu đầu vào bao gồm: thông tin mê cung, trạng thái đầu, trạng thái cuối. Dữ liệu này có thể truyền trực tiếp thông qua khai báo như sau (hoặc có thể đọc từ file txt):

```
16  const int COLS = 10;
17  const int ROWS = 9;
18
19
20  // Truong hop me cung co 5 dong va 5 cot
21
22  const int maze[][COLS] =        { {1, 1, 0, 1, 1},
23                                    { 0, 1, 0, 1, 0},
24                                    { 1, 1, 1, 1, 0},
25                                    { 0, 1, 0, 1, 0},
26                                    { 1, 0, 1, 1, 1}};
27
```

Mỗi trạng thái của mê cung sẽ được khai báo theo cấu trúc State:

```
80  struct State {
81      int robot_x;                // Toa do x cua con robot
82      int robot_y;                // Toa do y cua con robot
83  };
```

Robot chỉ có thể di chuyển lên, xuống, trái, phải sang các ô có giá trị là 1, vì vậy cần kiểm tra tính hợp lệ của các ô này thông qua hàm check_position được cho bên dưới:

```
102 // Dinh nghia ham kiem tra toa do x,y co hop le khong
103 bool check_position(int x, int y) {
104     if(x < 0 || y < 0)
105         return false;
106     if(x >= ROWS || y >= COLS)
107         return false;
108     if(maze[x][y] == 0)     // o khong di duoc
109         return false;
110     return true;
111 }
```

Hàm di chuyển lên của robot được cho như sau:

```
115 bool up(State current_state, State *new_state) {
116
117     int x = current_state.robot_x-1;
118     int y = current_state.robot_y ;
119
120     if(check_position(x,y)) {
121         new_state->robot_x = x;
122         new_state->robot_y = y;
123         return true;
124     } else {
125         return false;
126     }
127 }
```

Hàm call_operator sẽ lần lượt gọi các phép toán theo thứ tự di chuyển lên, xuống, trái, phải:

```
bool call_operator(State s, State *out, int option) {
    switch (option) {
        case 1:
            return up(s, out);
        case 2:
            return down(s, out);
        case 3:
            return left(s, out);
        case 4:
            return right(s, out);
        default:
            return false;
    }
}
```

Cấu trúc một node được cho như sau:

```
typedef struct Node{
    State state;
    struct Node* parent;
    int no_function; //hanh dong sinh ra trang thai hien tai
} Node;
```

Chuỗi các hành động lần lượt là:

```
// Hang chuoi de in ra ten cac hanh dong
const char* action[] = {"Trang thai bat dau",
                        "UP",
                        "DOWN",
                        "LEFT",
                        "RIGHT"};
```

s

<u>Câu hỏi:</u> Hãy viết các hàm cần thiết để thực hiện các công việc sau (test case được cho bên dưới):

a. Cài đặt hàm DOWN, LEFT, RIGHT và in các trạng thái con có thể có của một trạng thái hiện tại**.**

b. Sử dụng **giải thuật BFS**, in thứ tự duyệt các trạng thái, tổng số trạng thái đã duyệt và chuỗi các hành động cần thực hiện để tìm ra trạng thái cuối

c. Sử dụng **giải thuật A*** (sinh viên tự đề xuất hàm heuristic phù hợp), in thứ tự duyệt các trạng thái, tổng số trạng thái đã duyệt và chuỗi các hành động cần thực hiện để tìm ra trạng thái cuối

Kiểm tra với các test case sau:

1 1 0 1 1

0 1 0 1 0

1 1 1 1 0

0 1 0 1 0

1 0 1 1 1

Trạng thái đầu: (0, 0)

Trạng thái cuối: (4, 4), (0,4), (1,4)

Lưu ý: các câu a, b và c lưu vào các file riêng.

Quy tắc đặt tên file: MSSV_CT332XY_STT_lab3a.c hoặc MSSV_CT332XY_STT_lab3a.cpp (sinh viên nhớ thay thế XY là tên nhóm và STT là số thứ tự của sinh viên trong danh sách lớp). Tương tự cho câu b và c.