Maze

الف) راه حل بازگشتی:
از نقطه (۱،۱) شروع کرده و در هر گام تلاش می‌کنیم که به ترتیب به سمت پایین، چپ، راست و بالا، با استفاده از خود تابع حرکت کنیم(صدا زدن خود تابع با مختصات تغییر یافته) و هر نقطه‌ای را که می‌رویم را علامت زده تا به نقطه تکراری برنخوریم . حالت پایه زمانی است که مختصات نقطه (۱۴،۱۴) باشد که تابع true برمی‌گرداند.(اگر هیچ سمتی قابل رفتن نباشد، تابع بک ترک می‌کند)

ب) راه حل غیر بازگشتی:
اول از همه یک استک برای نقاط که هنوز نیاز به دیده شدن دارند می‌سازیم و نقطه (۱،۱) را به آن اضافه می‌کنیم. همچنین یک ماتریکس 15*15 نیاز داریم که همه به false مقداردهی شده که یعنی این نقاط را هنوز ندیده‌ایم و مانع ها را نیز true در نظر می‌گیریم که آن‌ها جزو مسیر نباشند. سپس حرکت‌ها را به ترتیب داده شده و عملیات مورد نیاز تعریف می‌کنیم. حالا وارد حلقه اصلی می‌شویم که تا زمانی که استک خالی نشده است(هنوز نقطه‌ای برای دیدن وجود دارد) ادامه دارد. در هر گام این حلقه، یک نقطه از استک pop کرده و اول چک می‌کنیم که اگر مقصد بود، true برمی‌گردانیم در غیر این صورت این نقطه را در ماتریکس true کرده و سپس نقاط پایین، چپ، راست و بالا را برای این نقطه محاسبه کرده و چک می‌کنیم که اگر معتبر بود(مانع نباشد، خارج از صفحه نباشد، قبلاً دیده نشده باشد)، آن را به استک نقاط push می‌کنیم. اگر استک خالی شود و به آخر حلقه برسیم یعنی به نقطه (۱۴،۱۴) نرسیدیم و false را برمی‌گردانیم.

```
//maze → matrix representing the maze with zeros and ones.(0→ obstacle)
Function path_in_maze(maze):
   dots_stack = create_empty_stack()
   dots_stack.push((1,1))
   visited = 15*15 matrix of False values
   visited[1][1] = True
   directions = [(1,0), (0,-1), (0,1), (-1,0)]

   while dots_stack is not empty:
      (x,y) = dots_stack.pop()
      if(dot == (14,14)):
         return True

      for each (dx,dy) in directions:
         newx = x + dx
         newy = y + dy

         //is new dot valid?
         If 0 <= newx < 15 and 0 <= newy < 15 and maze[newx][newy] == 1
and visited[newx][newy] == False:
            visited[newx][newy] = True
            dots_stack.push((newx,newy))
            break

    return False
```

یک راه حل بهتر می‌توانست این باشد که استک را برای جهت ها تعریف کنیم و در هر قدم به جهتی که حرکت کردیم تا به نقطه فعلی برسیم را به استک push کنیم و اگر هیچ جهتی قابل رفتن نبود یک عنصر از استک pop کرده و عملیات برعکس جهت pop شده را روی نقطه انجام می‌دهیم و سعی می‌کنیم جهت های دیگر را برای نقطه برویم و اگر همه جهت ها نیز تلاش شده بود، دوباره pop میکنیم. حالت پایه هم مانند راه حل قبلی است.