Pseudo Code of A Mazing Problem

Note from the instructor:

If you have the textbook, please refer to the **Program 3.15** and **Program 3.16** in page 156 and 158, respectively. If you don't have the textbook, please refer to the pseudo code of a mazing problem for **Assignment 2**.

```
initialize stack to the maze entrance coordinates and direction east;
while (stack is not empty)
{
    (i, j, dir) = coordinates and direction deleted from top of stack;
    delete last element of stack;
    while (there are more moves from (i, j))
    {
        (g, h) = coordinates of next move;
        if ((g = = m) && (h = = p)) success;
        if ((!maze [g][h])  // legal move
            && (!mark [g][h])) // haven't been here before
        {
            mark [g ][h] = 1;
            dir = next direction to try;
            add (i, j, dir) to top of stack;
            (i, j, dir) = (g, h, N) ;
        }
    }
}
cout << "No path in maze." << endl;
```

Program 3.15: First pass at finding a path through a maze

```
void Path(const int m, const int p)
{ // Output a path (if any) in the maze; maze[0][i] = maze[m+1][i] =
 // maze[j][0] = maze[j][p+1] = 1, 0 ≤ i ≤ p+1, 0 ≤ j ≤ m+1。

     // start at (1,1)
     mark[1][1] = 1;
     Stack<Items> stack(m*p);
     Items temp(1, 1, E);
        // set temp.x, temp.y, and temp.dir
     Stack.Push(temp);

     while (!stack.IsEmpty( ))
     { // stack not empty
          temp = stack.Top( );
          stack.Pop( ); // unstack
          int i = temp.x; int j = temp.y; int d = temp.dir;
          while (d < 8) // move forward
          {
               int g = i + move[d].a; int h = j + move[d].b;
               if ((g = = m) && (h = = p)) { // reached exit
                  // output path
                    cout << stack;
                    cout << i << " " << j << endl; // last two squares on the path
                    cout << m << " " << p << endl;
                    return;
               }
               if ((!maze [g ][h]) && (!mark [g ][h])) { // new position
                    mark[g][h] = 1;
                    temp.x = i; temp.y = j; temp.dir = d+1;
                    stack.Push(temp); // stack in
                    i = g; j = h; d = N; // move to (g, h)
               else d++; // try next direction
          }
     }
     cout << "No path in maze." << endl;
}
```

Program 3.16: Finding a path through a maze