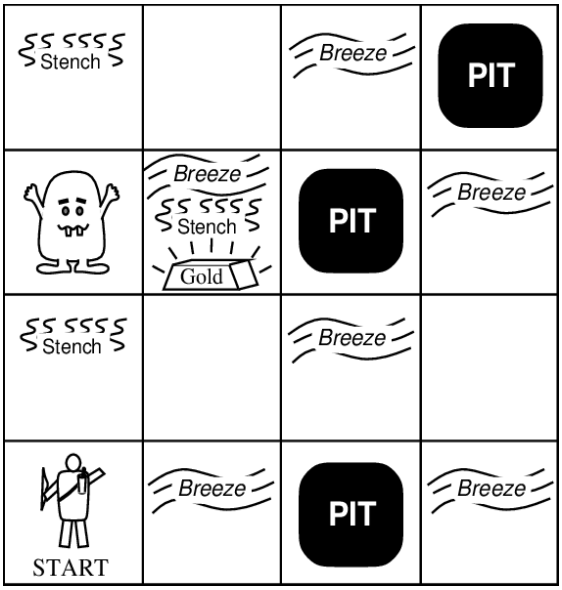
Wumpus World



Wumpus world is a popular grid game. The grid is a 4x4 array of cells. The bottom left cell is [0,0] and the top right cell is [3,3]. In any of these 16 cells, there can be:

1. Wumpus
2. Pit
3. Gold

If you want to survive in this world, you must get to the gold and hold it above your head to announce your victory. You do not know where the gold is. You may think scanning the whole grid would suffice to win, but the Wumpus and the Pit are there to stop you.

There is a Wumpus in the grid. The Wumpus is a smelly blind creature that devours souls when someone is in the same cell with it. It gives away a terrible smell, and if you are right by its cell, you will experience a stench.

There is a Pit in the grid. (The original version of the game can have any number of pits. For simplicity, we assume that there is only one pit). If you step in the Pit, you will have to rot there forever! However, you can always feel a breeze if you are right by the Pit.

As a player, you start from [0,0], moving upwards. You have the following moves:

1. Turn Right: Changes your moving direction
2. Turn Left: Changes your moving direction
3. Move Forward: Move to the next cell along the moving direction

4. Shoot: You have 3 arrows with you. You can shoot the arrow at your moving direction. If the Wumpus is in that direction, it will scream loudly in pain and die. But if it is behind you, you cannot touch it. You must turn around before shooting to kill it.

You are given a C++ skeleton code that (nearly) implements the above scenario. You must complete the code.

Tasks:

Part A:

Study the given skeleton carefully before going into the following tasks.

1. void Position::moveRight(): moves a certain position to the cell immediately to its right. Modify it so that the position cannot go beyond the grid.
2. void Position::moveLeft(): moves a certain position to the cell immediately to its left. Modify it so that the position cannot go beyond the grid.
3. void Position::moveUp(): moves a certain position to the cell immediately above it. Modify it so that the position cannot go beyond the grid.
4. void Position::moveDown(): moves a certain position to the cell immediately below it. Modify it so that the position cannot go beyond the grid.
5. bool Position::isAdjacent(Position p): implement the function. It returns true if two are adjacent. Two cells are adjacent if they are side by side. Note that [0,0] and [1,1] are not adjacent.
6. bool isSamePoint(Position p): implement the function. It returns true if two cells are the same.
7. Wumpus::Wumpus(): implement this default constructor of Wumpus class so that it initialized the position randomly within the grid, and sets it to alive.
8. Player::Player(): implement this default constructor of Player class. All necessary information is already given in the game description.
9. void Player::turnLeft(): implement this function. It changes the direction of the player as follows:

Before	**After**
UP	LEFT
LEFT	DOWN
DOWN	RIGHT
RIGHT	UP

10. void Player::turnRight(): implement this function. Figure out how the direction is affected by yourself.

11. void Player::moveForward(): implement this function. The player moves to the next cell along the direction.

12. WumpusWorld::WumpusWorld(): implement this default constructor. It will initialize the Wumpus, the Pit and the Gold at random positions.

13. WumpusWorld::WumpusWorld(int wumpus_x, int wumpus_y): implement this parameterized constructor. This is like the default constructor, only the location of the Wumpus is provided here.

14. WumpusWorld::WumpusWorld(int wumpus_x, int wumpus_y, int gold_x, int gold_y): implement this constructor. Here, all information except that of the Pit is delivered.

15. void WumpusWorld::shoot(): implement this function. This takes effect when a player shoots. Give proper messages if the player hits or misses.

Part B:

This is more of a planning and implementing task. The code that you now have works just fine (given that you have implemented it properly), except for the part of the Pit. There is no Pit in the code now. Implement the concept of Pit in your code. Identify the functions that need to be modified to do so. Modify those functions. You must think in object-oriented way and implement in object-oriented way.

Restrictions:

You cannot use any 2D array for the assignment. You cannot change the encapsulations that have been enforced in the given skeleton code.

You must not copy anyone's code. If copy is found, it will be made sure that you fail in the course.

Input Specifications:

Take six integers as input, x and y positions of the Wumpus, the Gold and the Pit respectively. All six integers will be within (0..3) inclusive.

Submission Guidelines:

You will submit a single .cpp file. Name it as follows: 2105xyz.cpp. Here, xyz are the last three digits of your student ID.