```
/*
gooseEscapeGamePlay.cpp
Jaden Durnford and Dennis Li
*/
#include <iostream>
#include <cmath>
using namespace std;
#include <BearLibTerminal.h>
#include "gooseEscapeUtil.hpp"
#include "gooseEscapeActor.hpp"
#include "gooseEscapeConsole.hpp"
#include "gooseEscapeGamePlay.hpp"

/*
    This file is all about the game world.  You will modify this to add
    functionality to your game, first to get it working, and later to make
    it fun.

    If you add or modify functions, be sure to update the prototypes in the
    gooseEscapeGamePlay.hpp file.
*/

extern Console out;

/*
With graphics, screens are given an x,y coordinate system with the origin
in the upper left corner.  So it means the coordinate axes are:
---------------->   x direction
|
|
|
|
|
V

y direction
*/

/*
  Print the game world

    The purpose of this function is to update the console to reflect the
    current state of the game board. It works by calling the terminal_put(...)
    function once for each of the game board elements, and then calling
    terminal_refresh() once after all elements have been put. Note that empty
    spaces do not need to be drawn.
*/
void printBoard(int gameBoard[20][70])
{
  for (int y_location_on_board = 0;y_location_on_board<20;y_location_on_board++)
  {
    for (int x_location_on_board = 0;x_location_on_board<70;x_location_on_board++)
    {
      if (gameBoard[y_location_on_board][x_location_on_board]!= 0)
      {
        if (gameBoard[y_location_on_board][x_location_on_board] == SHALL_NOT_PASS)
            terminal_put(x_location_on_board,y_location_on_board,'o');
          else if (gameBoard[y_location_on_board][x_location_on_board] == WINNER)
            terminal_put(x_location_on_board,y_location_on_board,'%');

        // after putting items on the game board, refresh the terminal to see the
items
        terminal_refresh();
      }
    }
  }
}

/*
    Do something when the goose captures the player

    At the moment the function just checks to see if the player and the goose
    are in the same location.  If you want to attack or do something else, this
    function would need to change.  For example, maybe the two touch each other
    and then fight.  You could use the health that is given in the Actor class,
    and update it.  Fight, run, use weapons, it is up to you!
*/

bool captured(Actor const & player, Actor const & monster)
{
    return (player.get_x() == monster.get_x()
         && player.get_y() == monster.get_y());
}

/*
    Move the player to a new location based on the user input

    All key presses start with "TK_" then the character.  So "TK_A" is the A
    key that was pressed.  At the moment, only the arrow keys are used,
    but feel free to add more keys that allow the player to do something else
    like pick up a power up.

    A look-up table might be useful.

    Going further: You could decide to learn about switch statements
*/
void movePlayer(int key, Actor & player, int gameBoard[20][70])
{
    int yMove = 0, xMove = 0;
    if (key == TK_UP)
```

```
        yMove = -1;
    else if (key == TK_DOWN)
        yMove = 1;
    else if (key == TK_LEFT)
        xMove = -1;
    else if (key == TK_RIGHT)
        xMove = 1;

    if (player.can_move(xMove, yMove)
      && gameBoard[player.get_y() + yMove][player.get_x()+ xMove] !=
SHALL_NOT_PASS)
    {
      player.update_location(xMove, yMove);
    }
}

/*
    Move the goose to a new location based on its position relative to the player

    the function checks whether the goose is farther from the player in the x
    direction or y direction to determine which way the goose should be moved.
    If the goose was previously covering a wall or win character, it gets
    replaced on the following move it makes.
*/
void moveGoose(Actor & player, Actor & monster, int gameBoard[20][70])
{
  int playerX = 0, playerY = 0, gooseX = 0, gooseY = 0, diffX = 0, diffY = 0,
      gooseMoveX = 0, gooseMoveY = 0;
  playerX = player.get_x();
  playerY = player.get_y();
  gooseX = monster.get_x();
  gooseY = monster.get_y();

  diffX = gooseX - playerX;
  diffY = gooseY - playerY;
  if (abs(diffX) > abs(diffY))
  {
    if (diffX > 0)
    {
      gooseMoveX = -1;
    }
    else if (diffX < 0)
    {
      gooseMoveX = 1;
    }
  }
  else
  {
    if (diffY > 0)
    {
      gooseMoveY = -1;
    }
    else if (diffY < 0)
    {
      gooseMoveY = 1;
    }
  }

  monster.update_location(gooseMoveX,gooseMoveY);
  if (gameBoard[monster.get_y() - gooseMoveY][monster.get_x() - gooseMoveX] ==
SHALL_NOT_PASS)
  {
    terminal_put(monster.get_x() - gooseMoveX,monster.get_y() - gooseMoveY,'o');
  }
  else if (gameBoard[monster.get_y() - gooseMoveY][monster.get_x() - gooseMoveX] ==
WINNER)
  {
    terminal_put(monster.get_x() - gooseMoveX,monster.get_y() - gooseMoveY,'%');
  }
}
```