```
/***GOOSE ESCAPE GAME PLAY CPP FILE BEGINNING!!***/

#include <iostream>
#include <cmath>
using namespace std;
#include "BearLibTerminal_0.15.7\Include\C\BearLibTerminal.h"
#include "gooseEscapeUtil.hpp"
#include "gooseEscapeActors.hpp"
#include "gooseEscapeConsole.hpp"
#include "gooseEscapeGamePlay.hpp"

//Static variables to store teleporter things and keeps things clean:
static int teleportX1 = 0, teleportX2 = 0, teleportY1 = 0, teleportY2 = 0;
static bool hasJustTeleported = false; //Makes sure you don't teleport twice

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 functions should draw characters to present features of the game
    board, e.g. win location, obstacles, power ups
*/

// print the game board function
void printGameBoard(int gameBoard[NUM_BOARD_Y][NUM_BOARD_X])
{
    for(int row = 0; row <= MAX_BOARD_Y; row++)
    {
        for(int col = 0; col <= MAX_BOARD_X; col++)
        {
            int charToPut = BLANK_CHAR;
            if(gameBoard[row][col] == SHALL_NOT_PASS)
            {
                charToPut = WALL_CHAR;
            }
```

```
            else if(gameBoard[row][col] == WINNER)
            {
                charToPut = WIN_CHAR;
            }
            else if(gameBoard[row][col] == TELEPORT)
            {
                charToPut = TELEPORT_CHAR;
            }
            terminal_put(col, row, charToPut);
        }
    }


}

//Updates the private static variables to store the teleporter locations
void sendGameBoardCoordinates(int teleX1, int teleY1, int teleX2, int teleY2)
{
    teleportX1 = teleX1;
    teleportY1 = teleY1;
    teleportX2 = teleX2;
    teleportY2 = teleY2;
}

/*
    Do something when the goose captures the player

    If you want to attack or something else, this is the function you
    need to change.  For example, maybe the two touch each other and
    then fight.  You could add a health to the Actor class that is
    updated.  Run, use weapons, it's 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.  You may want
    to modify this if there are extra controls you want to add.

    All key presses start with "TK_" then the character.  So "TK_A" is the a
    key being pressed.

    A look-up table might be useful.
    You could decide to learn about switch statements and use them here.
*/
```

```
void movePlayer(int key, Actor & player, int gameBoard[NUM_BOARD_Y][NUM_BOARD_X])
{
    //Inbuilt function
    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;

    int playerX = player.get_x();
    int playerY = player.get_y();

    if (player.can_move(xMove, yMove)
      && gameBoard[playerY+yMove][playerX+xMove] != SHALL_NOT_PASS)
      {
        //Checks if the player can teleport
        if(isTeleportable(gameBoard, player) && !hasJustTeleported)
        {
            teleportActor(player);
            hasJustTeleported = true;
        }
        else
        {
            hasJustTeleported = false;
            updatePlayerLocation(player, xMove, yMove);

        }

      }

      place_teleporter(); //Done to keep the teleporters on screen

}

void updatePlayerLocation(Actor &player, int xMove, int yMove)
{

    int playerX = player.get_x();
    int playerY = player.get_y();
    player.update_location(xMove, yMove);
    terminal_put(playerX,playerY, BLANK_CHAR); //This is done to keep the blank char
      on screen
    //instead of the actor leaving a trail of nothing
    terminal_refresh();
```

```
}

bool isTeleportable(int gameBoard[NUM_BOARD_Y][NUM_BOARD_X], Actor &player)
{
    return gameBoard[player.get_y()][player.get_x()] == TELEPORT; //Checks the
      gameboard for the teleport string
}

void teleportActor(Actor &player)
{
    int playerX = player.get_x();
    int playerY = player.get_y();

    int chosenX = 0, chosenY = 0;
    if(playerX == teleportX1 && playerY == teleportY1)
    {
        chosenX = teleportX2;
        chosenY = teleportY2;
    }
    else
    {
        chosenX = teleportX1;
        chosenY = teleportY1;
    }

    player.setLocation(chosenX, chosenY);

    updatePlayerLocation(player, 0, 0);
    terminal_refresh();
    player.put_actor();
}

void place_teleporter()
{
    terminal_put(teleportX1, teleportY1, TELEPORT_CHAR);
    terminal_put(teleportX2, teleportY2, TELEPORT_CHAR);
    terminal_refresh();
}

bool won(Actor &player, int gameBoard[NUM_SCREEN_Y][NUM_SCREEN_X])
{
    int playerX = player.get_x();
    int playerY = player.get_y();

    return (gameBoard[playerY][playerX] == WINNER);
}

void chasePlayer(Actor & monster, Actor const & player, int gameBoard[NUM_BOARD_Y]
  [NUM_BOARD_X])
```

```
{
    int playerX = player.get_x();
    int playerY = player.get_y();

    int monsterX = monster.get_x();
    int monsterY = monster.get_y();

    // Check move buffer for monster
    if(monster.get_buffer() > 0){
        monster.increment_buffer();
        return;
    }

    int deltaX = 0;
    int deltaY = 0;
    // Moves the monster one unit in both X and Y towards player
    // If there is a wall in the way the monster will jump over unless wall is more
      then 1 unit thick

    //Player is Right of Monster
    if(playerX > monsterX)
    {
        deltaX++;
        if(gameBoard[monsterY][monsterX + 1] == SHALL_NOT_PASS && gameBoard[monsterY]
          [monsterX + 2] != SHALL_NOT_PASS)
        {
            deltaX++;
        }
        else if(gameBoard[monsterY][monsterX + 1] == SHALL_NOT_PASS && gameBoard
          [monsterY][monsterX + 2] == SHALL_NOT_PASS)
        {
            deltaX--;
        }
    }
    //Player is Left of Monster
    else if(playerX < monsterX)
    {
        deltaX--;
        if(gameBoard[monsterY][monsterX - 1] == SHALL_NOT_PASS && gameBoard[monsterY]
          [monsterX - 2] != SHALL_NOT_PASS)
        {
            deltaX--;
        }
        else if(gameBoard[monsterY][monsterX - 1] == SHALL_NOT_PASS && gameBoard
          [monsterY][monsterX - 2] == SHALL_NOT_PASS)
        {
            deltaX++;
        }
    }
```

```
    //Player is Below Monster
    if(playerY > monsterY)
    {
        deltaY++;
        if(gameBoard[monsterY + 1][monsterX] == SHALL_NOT_PASS && gameBoard[monsterY +
           2][monsterX] != SHALL_NOT_PASS)
        {
            deltaY++;
        }
        else if(gameBoard[monsterY + 1][monsterX] == SHALL_NOT_PASS && gameBoard
          [monsterY + 2][monsterX] == SHALL_NOT_PASS)
        {
            deltaY--;
        }

    }
    //Player is Above Monster
    else if(playerY < monsterY)
    {
        deltaY--;
        if(gameBoard[monsterY - 1][monsterX] == SHALL_NOT_PASS && gameBoard[monsterY -
           2][monsterX] != SHALL_NOT_PASS)
        {
            deltaY--;
        }
        else if(gameBoard[monsterY - 1][monsterX] == SHALL_NOT_PASS && gameBoard
          [monsterY - 2][monsterX] == SHALL_NOT_PASS)
        {
            deltaY++;
        }
    }
    monster.update_location(deltaX, deltaY);
    monster.increment_buffer();
    terminal_put(monsterX,monsterY, BLANK_CHAR);
    terminal_refresh();
}


/*
    What other functions do you need to make the game work?  What can you
    add to the basic functionality to make it more fun to play?
*/
```