```
/***MAIN.CPP FILE BEGINNING!!***/

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

//set up the console.   Don't modify this line!
Console out;


void create_walls(int gameBoard[NUM_BOARD_Y][NUM_BOARD_X])
{
    // Creates walls at predetermined locations
    const int MID_BOARD_Y = MAX_BOARD_Y/2, MID_BOARD_X = MAX_BOARD_X/2;
    for(int width = 21; width < NUM_BOARD_X - 20; width++)
    {
        gameBoard[MID_BOARD_Y + 2][width] = SHALL_NOT_PASS;
    }

    for(int height = 0; height < 4; height++)
    {
        gameBoard[height + 3][MID_BOARD_X - 10] = SHALL_NOT_PASS;
        gameBoard[height + 3][MID_BOARD_X + 10] = SHALL_NOT_PASS;

        gameBoard[height + 14][MID_BOARD_X - 25] = SHALL_NOT_PASS;
        gameBoard[height + 14][MID_BOARD_X + 27] = SHALL_NOT_PASS;
    }
}

int main()
{
    //Set up the window.  Don't edit these two lines
    terminal_open();
    terminal_set(SETUP_MESSAGE);

/*
    The code below provides a skeleton of the game play.  You will need to
    write code for setting up the game board, and playing the game itself.
    You can modify the code given as needed.

    Call the functions that you have written in the game play file, and that
    you have added to the Actor class.
*/
```

```
    //make the player
    Actor player(PLAYER_CHAR, 39,5, 0, 0);  // you probably don't want to start in the
       same place each time

    //make the monster
    Actor monster(MONSTER_CHAR, 39,19, 0, 1);
    terminal_refresh();

    // Declare the array that will hold the game board "map"
    int gameBoard[NUM_BOARD_Y][NUM_BOARD_X] = {0};


/*
    Initiallize locations in the game board to have game features.  What if you
    have man things to add to the game board?  Should you use a loop?  Does it
    make sense to store this information in a file?  Should this code be a
    function as well?
*/
    create_walls(gameBoard);

    gameBoard[MAX_BOARD_Y][MAX_BOARD_X/2] = WINNER;

    //Teleporter Locations:
    int teleX1 = 5, teleY1 = 2, teleX2 = 67, teleY2 = 3;

    gameBoard[teleY1][teleX1] = TELEPORT;
    gameBoard[teleY2][teleX2] = TELEPORT;

    sendGameBoardCoordinates(teleX1, teleY1, teleX2, teleY2);

    // Call the function to print the game board
    printGameBoard(gameBoard);

    // Printing the instructions
    out.writeLine("Escape the Goose! " + monster.get_location_string());
    out.writeLine("Use the arrow keys to move");
    out.writeLine("If the goose catches you, you lose!");
    out.writeLine("Be careful! Sometimes the goose can jump through walls!");
    player.put_actor();

/*
    This is the main game loop.  It continues to let the player give input
    as long as they do not press escape or close, they are not captured by
    the goose, and they didn't reach the win tile
*/
/*
    All key presses start with "TK_" then the character.  So "TK_A" is the "a"
    key being pressed.
*/
```

```
    int keyEntered = TK_A;  // can be any valid value that is not ESCAPE or CLOSE

    while(keyEntered != TK_ESCAPE && keyEntered != TK_CLOSE
                    && !captured(player,monster) && !won(player,gameBoard))
    {
        // get player key press
        keyEntered = terminal_read();

        if (keyEntered != TK_ESCAPE && keyEntered != TK_CLOSE)
        {
            // move the player, you can modify this function
            movePlayer(keyEntered,player,gameBoard);

            // call the goose's chase function
            chasePlayer(monster, player, gameBoard);

            // call other functions to do stuff?
        }
    }

    if (keyEntered != TK_CLOSE)
    {
        //once we're out of the loop, the game is over
        out.writeLine("Game has ended");

        if(won(player, gameBoard))
            {
            out.writeLine("Player has escaped!!!");
            }
        else
            {
            out.writeLine("Player has been captured!");
            }

        // Wait until user closes the window
        while (terminal_read() != TK_CLOSE);
    }

    //game is done, close it
    terminal_close();
}
```