**Car Hurdle 2d Game**

****

Session: 2022 – 2026

**Submitted by:**

Muhammad Taha Saleem 2022-CS-139

**Supervised by:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology, Lahore, Pakistan**

**Table of Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Description and Story………………………………………………………………………….**

**Game Characters.................…………………………………………………………………….**

**Players…………………………………………………………………………………………..**

**Enemies…………………………………………………………………………………………**

**Objects Description.....…………………………………………………………………………**

**Rules and Interactions……........……………………………………………………………..**

**Goal of Game……........……………………………………………………………..……........….**

**Wireframes……........……………………………………………………………..……........…......**

**Data Structures……............………………………………………………..……........…...........**

**Function Prototypes………………………………………………………..……........……….**

**Complete Code....……………………………………………………………..……........……….**

* **Description and Story of Game**

Basically it is a Car Hurdle 2d Game in which there are different Difficulty Levels. There are different no of Cars(Player+Enemy) according to the Difficulty Level and also Game becomes harder with difficulty level and Instructions are written in the Game.

* **Game Characters Description** 
  + **Player Car**

There is one human Player Car in the Game in 4 levels. But in Basic level there is a Tank which plays role as a Player.

* + **Tank**

But in Basic level there is a Tank which plays role as a Player.

* + **Opponent Car**

There is room for an Opponent Car in Single Player Level.

* + **Enemies**

There are Total 4 enemies in the game.

**Car1:**

It is a Car with less ability to Save the Bala. It can move To and Fro for some distance. It defends itself and Bala by firing bullets when player car comes close to it.

**Car2:**

It playes an important role by chasing the enemy car.

**Car3:**

Its Functionality is to Defend itself and Monster Bala and to defend the lower part of the road.

**Monster Bala:**

Bala is the leader of the opponent cars. It can move Vertically and it also has the ability to move Vertically and Fire bullets from a long distance.

* Game Objects Description

Following are the Objects in the Game

* + **Monster Bala:** Monster Bala should be protected by Enemy cars from the Player Car.
* **Rules & Interactions**

Player Car is moved and Protected from enemy cars .It has many difficulty Levels. Follow the following instructions and Enjoy Game.

1. Player Car Have 100 Bullets only.
2. Player Car will have to Kill Monster Bala before losing its Life.
3. Collision with any enemy would result in Lost the Game.
4. Health of Player is 15.
5. Health of Bala is 20.

* **Goal of the Game**

The goal of the game is to Destroy Monster Bala and Reach score of 20 before losing its Life.

* **Wireframes**



**Figure 1: Level and Welcome Menu**

****

**Figure 2: Basic Level**

****

**Figure 3: Easy Mode**

****

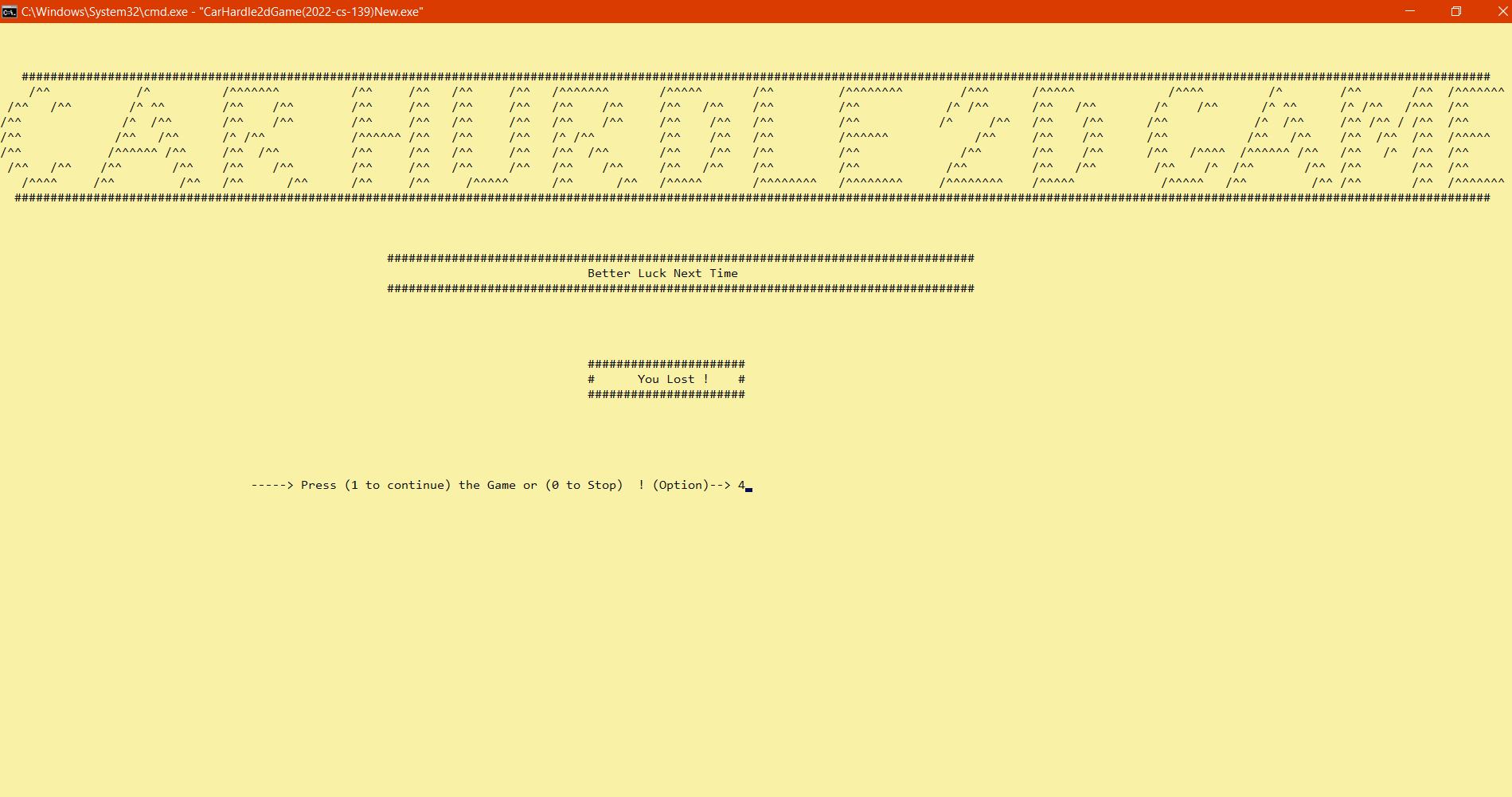
**Figure 4: Medium Mode**

****

**Figure 5: Single Player**

****

**Figure 6: Double Player**

****

**Figure 7: Lost screen**

* **Data Structures**
* char enemy[4][6] = {
* {' ', ' ', ' ', '-', '-', '-'},
* {'<', '=', '=', '(', '-', ')'},
* {' ', ' ', ' ', '\\', '@', '/'},
* {' ', ' ', ' ', '\*', '\*', '\*'}};
* char erase\_Enemy[4][6] = {{' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '}};
* char erase\_TankEnemy[4][6] = {{' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '}};
* char erase\_Car[3][7] = {{' ', ' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' ', ' '}};
* char car[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},
* {' ', box, box, box, box, box, ' '},
* {' ', wheel, ' ', ' ', ' ', wheel, ' '}};
* char car1[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},
* {' ', box, box, box, box, box, ' '},
* {' ', wheel, ' ', ' ', ' ', wheel, ' '}};
* char car2[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},
* {' ', box, box, box, box, box, ' '},
* {' ', wheel, ' ', ' ', ' ', wheel, ' '}};
* char car3[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},
* {' ', box, box, box, box, box, ' '},
* {' ', wheel, ' ', ' ', ' ', wheel, ' '}};
* char tank[2][6] = {{tankBox, tankBox, tankBox, '-', '-', '>'},
* {'0', ' ', '0', ' ', ' ', ' '}};
* char erase\_tank[2][6] = {{' ', ' ', ' ', ' ', ' ', ' '},
* {' ', ' ', ' ', ' ', ' ', ' '}};
* char TankEnemy[4][6] = {
* {' ', ' ', ' ', '-', '-', '-'},
* {'<', '=', '=', '(', '-', ')'},
* {' ', ' ', ' ', '\\', '@', '/'},
* {' ', ' ', ' ', '\*', '\*', '\*'}};
* int store\_score;
* int store\_life;
* char randomPrevious = ' ';
* char smartPrevious = ' ';
* char horizontalPrevious = ' ';
* // int enemyCar3X = 130;
* // int enemyCar3Y = 19;
* int randomCarx = 2;
* int randomCary = 2;
* int tankBulletX[100];
* int tankBulletY[100];
* int bulletX[150];
* int bulletY[150];
* int EbulletX[100];
* int EbulletY[100];
* int EC1bulletX[100];
* int EC1bulletY[100];
* int EC2bulletX[100];
* int EC2bulletY[100];
* bool isBulletActive[150];
* bool isEBulletActive[100];
* bool isEC1BulletActive[100];
* bool isEC2BulletActive[100];
* bool state1;
* int enemyTankBulletX[100];
* int enemyTankBulletY[100];
* bool isEnemyTankBulletActive[100];
* int enemyTankBulletCount = 0;
* int carX = 6;
* int carY = 6;
* int enemyX = 141;
* int enemyY = 10;
* int enemyCar1X = 130;
* int enemyCar1Y = 6;
* int enemyCar2X = 130;
* int enemyCar2Y = 32;
* int enemyCar3X = 130;
* int enemyCar3Y = 19;
* string tankEnemyDirection = "Up";
* string enemyCar1Direction = "Left";
* string enemyDirection = "Up";
* int tankX = 5;
* int tankY = 5;
* int TankEnemyX = 30;
* int TankEnemyY = 10;
* int timer = 0;
* int score = 0;
* int health = 20;
* int life = 15;
* **Function Prototypes**
* void printTank();
* void eraseTank();
* void printTankRoad();
* void moveTankBullet();
* void generateTankBullet();
* void removeTankBulletFromArray(int index);
* void moveTankUp();
* void moveTankDown();
* void moveTankLeft();
* void moveTankRight();
* void moveEnemyTankBullet();
* void moveTankEnemy();
* void generateTankEnemyBullet();
* void printTankEnemy();
* void eraseTankEnemy();
* void printTankScore();
* void TankbulletCollisionWithEnemy();
* void gotoxy(int x, int y);
* char getCharAtxy(short int x, short int y);
* void printLostMenu();
* void printLostMenu2();
* void printWonMenu();
* void printWelcomeMenu();
* void moveEnemy();
* void moveEnemyCar1();
* void moveEnemyCar2();
* // void moveEnemyCar3();
* void bulletCollisionWithEnemy();
* void bulletCollisionWithPlayer();
* void enemyCar1CollisionWithPlayer();
* void enemyCar2CollisionWithPlayer();
* void enemyCar3CollisionWithPlayer();
* void enemyCarCollisionWithPlayer();
* void C1bulletCollisionWithPlayer();
* void C2bulletCollisionWithPlayer();
* void makeBulletInactive(int index);
* void makeEBulletInactive(int index);
* void makeEC1BulletInactive(int index);
* void makeEC2BulletInactive(int index);
* void removeBulletFromArray(int index);
* void removeEBulletFromArray(int index);
* void removeEC1BulletFromArray(int index);
* void removeEC2BulletFromArray(int index);
* void printCar();
* void eraseCar();
* void printEnemy();
* void eraseEnemy();
* void printECar();
* void eraseECar();
* void printE1Car();
* void eraseE1Car();
* void printE2Car();
* void eraseE2Car();
* void printHeader();
* void printRoad();
* void moveCarUp();
* void moveCarDown();
* void moveCarLeft();
* void moveCarRight();
* void moveEnemy();
* void moveECarUp();
* void moveECarDown();
* void moveECarLeft();
* void moveECarRight();
* void moveBullet();
* void generateBullet();
* void generateEC1Bullet();
* void moveEC1Bullet();
* void generateEC2Bullet();
* void moveEC2Bullet();
* void moveEBullet();
* void generateEBullet();
* void enemyFiring();
* void enemyCar1Firing();
* void enemyCar2Firing();
* void printMenuAndDifficultyLevelScreen();
* void easyMode(void);
* void mediumMode(void);
* void hardMode(void);
* void singlePlayerHardMode();
* void basicMode(void);
* void addScore();
* void printScore();
* void print\_enemyHealth();
* void printLife();
* void decreaseLife();
* void printBullets();
* void bulletsCapacity();
* void printBullet(int x, int y);
* void eraseBullet(int x, int y);
* void printInGameFooter();
* void printInstructions();
* **Complete Code**

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <cmath>

#include <fstream>

// #include<random>

using namespace std;

char box = 223;

char wheel = 233;

char w1 = 201;

char w2 = 203;

char w3 = 187;

char tankBox = 219;

int count = 0;

int bulletCount = 0;

int bulletCounter = 100;

int EbulletCount = 0;

int EC1bulletCount = 0;

int EC2bulletCount = 0;

char TankRoad[18][42] = {{'#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'},

{'#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'}};

char Game\_Road[41][164] = {{'\*', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '\*'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '|', '#'},

{'#', '|', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '|', '#'},

{'#', '|', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', '#', '#', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'#', '|', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', '|', '#'},

{'\*', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '\*'}};

char enemy[4][6] = {

{' ', ' ', ' ', '-', '-', '-'},

{'<', '=', '=', '(', '-', ')'},

{' ', ' ', ' ', '\\', '@', '/'},

{' ', ' ', ' ', '\*', '\*', '\*'}};

char erase\_Enemy[4][6] = {{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '}};

char erase\_TankEnemy[4][6] = {{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '}};

char erase\_Car[3][7] = {{' ', ' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', ' '}};

char car[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},

{' ', box, box, box, box, box, ' '},

{' ', wheel, ' ', ' ', ' ', wheel, ' '}};

char car1[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},

{' ', box, box, box, box, box, ' '},

{' ', wheel, ' ', ' ', ' ', wheel, ' '}};

char car2[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},

{' ', box, box, box, box, box, ' '},

{' ', wheel, ' ', ' ', ' ', wheel, ' '}};

char car3[3][7] = {{' ', ' ', w1, w2, w3, ' ', ' '},

{' ', box, box, box, box, box, ' '},

{' ', wheel, ' ', ' ', ' ', wheel, ' '}};

char tank[2][6] = {{tankBox, tankBox, tankBox, '-', '-', '>'},

{'0', ' ', '0', ' ', ' ', ' '}};

char erase\_tank[2][6] = {{' ', ' ', ' ', ' ', ' ', ' '},

{' ', ' ', ' ', ' ', ' ', ' '}};

char TankEnemy[4][6] = {

{' ', ' ', ' ', '-', '-', '-'},

{'<', '=', '=', '(', '-', ')'},

{' ', ' ', ' ', '\\', '@', '/'},

{' ', ' ', ' ', '\*', '\*', '\*'}};

int store\_score;

int store\_life;

char randomPrevious = ' ';

char smartPrevious = ' ';

char horizontalPrevious = ' ';

// int enemyCar3X = 130;

// int enemyCar3Y = 19;

int randomCarx = 2;

int randomCary = 2;

int tankBulletX[100];

int tankBulletY[100];

int bulletX[150];

int bulletY[150];

int EbulletX[100];

int EbulletY[100];

int EC1bulletX[100];

int EC1bulletY[100];

int EC2bulletX[100];

int EC2bulletY[100];

bool isBulletActive[150];

bool isEBulletActive[100];

bool isEC1BulletActive[100];

bool isEC2BulletActive[100];

bool state1;

int enemyTankBulletX[100];

int enemyTankBulletY[100];

bool isEnemyTankBulletActive[100];

int enemyTankBulletCount = 0;

int carX = 6;

int carY = 6;

int enemyX = 141;

int enemyY = 10;

int enemyCar1X = 130;

int enemyCar1Y = 6;

int enemyCar2X = 130;

int enemyCar2Y = 32;

int enemyCar3X = 130;

int enemyCar3Y = 19;

string tankEnemyDirection = "Up";

string enemyCar1Direction = "Left";

string enemyDirection = "Up";

int tankX = 5;

int tankY = 5;

int TankEnemyX = 30;

int TankEnemyY = 10;

int timer = 0;

int score = 0;

int health = 20;

int life = 15;

void printTank();

void eraseTank();

void printTankRoad();

void moveTankBullet();

void generateTankBullet();

void removeTankBulletFromArray(int index);

void moveTankUp();

void moveTankDown();

void moveTankLeft();

void moveTankRight();

void moveEnemyTankBullet();

void moveTankEnemy();

void generateTankEnemyBullet();

void printTankEnemy();

void eraseTankEnemy();

void printTankScore();

void TankbulletCollisionWithEnemy();

void gotoxy(int x, int y);

char getCharAtxy(short int x, short int y);

void printLostMenu();

void printLostMenu2();

void printWonMenu();

void printWelcomeMenu();

void moveEnemy();

void moveEnemyCar1();

void moveEnemyCar2();

// void moveEnemyCar3();

void bulletCollisionWithEnemy();

void bulletCollisionWithPlayer();

void enemyCar1CollisionWithPlayer();

void enemyCar2CollisionWithPlayer();

void enemyCar3CollisionWithPlayer();

void enemyCarCollisionWithPlayer();

void C1bulletCollisionWithPlayer();

void C2bulletCollisionWithPlayer();

void makeBulletInactive(int index);

void makeEBulletInactive(int index);

void makeEC1BulletInactive(int index);

void makeEC2BulletInactive(int index);

void removeBulletFromArray(int index);

void removeEBulletFromArray(int index);

void removeEC1BulletFromArray(int index);

void removeEC2BulletFromArray(int index);

void printCar();

void eraseCar();

void printEnemy();

void eraseEnemy();

void printECar();

void eraseECar();

void printE1Car();

void eraseE1Car();

void printE2Car();

void eraseE2Car();

void printHeader();

void printRoad();

void moveCarUp();

void moveCarDown();

void moveCarLeft();

void moveCarRight();

void moveEnemy();

void moveECarUp();

void moveECarDown();

void moveECarLeft();

void moveECarRight();

void moveBullet();

void generateBullet();

void generateEC1Bullet();

void moveEC1Bullet();

void generateEC2Bullet();

void moveEC2Bullet();

void moveEBullet();

void generateEBullet();

void enemyFiring();

void enemyCar1Firing();

void enemyCar2Firing();

void printMenuAndDifficultyLevelScreen();

void easyMode(void);

void mediumMode(void);

void hardMode(void);

void singlePlayerHardMode();

void basicMode(void);

void addScore();

void printScore();

void print\_enemyHealth();

void printLife();

void decreaseLife();

void printBullets();

void bulletsCapacity();

void printBullet(int x, int y);

void eraseBullet(int x, int y);

void printInGameFooter();

void printInstructions();

//---------------------------------------------------------------

string getField(string record, int field)

{

int commaCount = 1;

string item;

for (int idx = 0; idx < record.length(); idx++)

{

if (record[idx] == ',')

{

commaCount++;

}

else if (commaCount == field)

{

item = item + record[idx];

}

}

return item;

}

void store\_data\_in\_file(int score, int life, int bullets, int health)

{

fstream file;

file.open("Game\_Data.txt", ios::out);

file << score << "," << life << "," << bullets << "," << health << "\n";

file.close();

}

void readData()

{

string record;

fstream file;

file.open("Game\_Data.txt", ios::in);

while (getline(file, record))

{

score = stoi(getField(record, 1));

life = stoi(getField(record, 2));

bulletCounter = stoi(getField(record, 3));

health = stoi(getField(record, 4));

}

file.close();

}

void initialValues()

{

carX = 6;

carY = 6;

score = 0;

health = 20;

life = 15;

bulletCounter = 100;

tankX = 5;

tankY = 5;

timer = 0;

count = 0;

bulletCount = 0;

EbulletCount = 0;

EC1bulletCount = 0;

EC2bulletCount = 0;

enemyTankBulletCount = 0;

enemyX = 141;

enemyY = 10;

enemyCar1X = 130;

enemyCar1Y = 6;

enemyCar2X = 130;

enemyCar2Y = 32;

enemyCar3X = 130;

enemyCar3Y = 19;

}

void ending()

{

initialValues();

printMenuAndDifficultyLevelScreen();

}

void returnToEnding()

{

system("cls");

ending();

}

void resume()

{

readData();

printMenuAndDifficultyLevelScreen();

}

void invalidOption()

{

cout << " Invalid Option MASTI NA KR BAAZ A JA ! [Quiting Game...] ";

Sleep(2000);

state1 = false;

}

void choice()

{

string choice;

cout << " -----> Press (1 to continue) the Game or (0 to Stop) ! (Option)--> ";

cin >> choice;

if (choice == "1")

{

returnToEnding();

}

else if (choice == "0")

{

state1 = false;

}

else

{

invalidOption();

}

}

void back\_option()

{

if (GetAsyncKeyState(VK\_ESCAPE))

{

store\_data\_in\_file(score, life, bulletCounter, health);

resume();

}

}

float distance(int gx, int gy)

{

return sqrt(pow(carX - gx, 2) + pow(carY - gy, 2));

}

void moveCarSmart()

{

float left = distance(enemyCar3X - 1, enemyCar3Y);

float right = distance(enemyCar3X + 1, enemyCar3Y);

float up = distance(enemyCar3X, enemyCar3Y - 1);

float down = distance(enemyCar3X, enemyCar3Y + 1);

if (left <= down && left <= right && left <= up)

{

char next1 = getCharAtxy(enemyCar3X - 1, enemyCar3Y + 4);

char next2 = getCharAtxy(enemyCar3X - 1, enemyCar3Y - 1);

char next = getCharAtxy(enemyCar3X - 1, enemyCar3Y);

if ((next == ' ' && next != '#') || (next1 != '#' && next2 != '#'))

{

eraseE2Car();

gotoxy(enemyCar3X, enemyCar3Y);

cout << smartPrevious;

smartPrevious = next;

enemyCar3X = enemyCar3X - 1;

// gotoxy(enemyCar3X, enemyCar3Y);

printE2Car();

}

}

else if (right <= down && right <= left && right <= up)

{

char next1 = getCharAtxy(enemyCar3X + 1, enemyCar3Y + 4);

char next2 = getCharAtxy(enemyCar3X + 1, enemyCar3Y - 1);

char next = getCharAtxy(enemyCar3X + 1, enemyCar3Y);

if ((next == ' ' && next != '#') || (next1 != '#' && next2 != '#'))

{

eraseE2Car();

gotoxy(enemyCar3X, enemyCar3Y);

cout << smartPrevious;

smartPrevious = next;

enemyCar3X = enemyCar3X + 1;

// gotoxy(enemyCar3X, enemyCar3Y);

printE2Car();

}

}

else if (down <= left && down <= right && down <= up)

{

char next1 = getCharAtxy(enemyCar3X, enemyCar3Y + 4);

char next2 = getCharAtxy(enemyCar3X, enemyCar3Y - 1);

char next = getCharAtxy(enemyCar3X, enemyCar3Y + 3);

if ((next == ' ' && next != '#') || (next1 != '#' && next2 != '#'))

{

eraseE2Car();

gotoxy(enemyCar3X, enemyCar3Y);

cout << smartPrevious;

smartPrevious = next;

enemyCar3Y = enemyCar3Y + 1;

// gotoxy(enemyCar3X, enemyCar3Y);

printE2Car();

}

}

else

{

char next1 = getCharAtxy(enemyCar3X, enemyCar3Y + 4);

char next2 = getCharAtxy(enemyCar3X, enemyCar3Y - 1);

char next = getCharAtxy(enemyCar3X, enemyCar3Y - 1);

if ((next == ' ' && next != '#') || (next1 != '#' && next2 != '#'))

{

eraseE2Car();

gotoxy(enemyCar3X, enemyCar3Y);

cout << smartPrevious;

smartPrevious = next;

enemyCar3Y = enemyCar3Y - 1;

// gotoxy(enemyCar3X, enemyCar3Y);

printE2Car();

}

}

}

main()

{

system("cls");

printMenuAndDifficultyLevelScreen();

}

void singlePlayerHardMode()

{

system("cls");

printRoad();

printInGameFooter();

printInstructions();

printCar();

printECar();

printE1Car();

printE2Car();

printEnemy();

state1 = true;

while (state1 != false)

{

back\_option();

bulletsCapacity();

printScore();

print\_enemyHealth();

printLife();

printBullets();

if (GetAsyncKeyState(VK\_LEFT))

{

moveCarLeft();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveCarRight();

}

if (GetAsyncKeyState(VK\_UP))

{

moveCarUp();

}

if (GetAsyncKeyState(VK\_DOWN))

{

moveCarDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

generateBullet();

}

// if (GetAsyncKeyState(VK\_NUMPAD0))

// {

// generateEBullet();

// }

if (timer == 3)

{

moveEnemy();

moveEnemyCar1();

moveEnemyCar2();

// moveEnemyCar3();

moveCarSmart();

timer = 0;

}

moveBullet();

moveEBullet();

moveEC1Bullet();

moveEC2Bullet();

bulletCollisionWithEnemy();

bulletCollisionWithPlayer();

enemyCar1CollisionWithPlayer();

enemyCar2CollisionWithPlayer();

enemyCar3CollisionWithPlayer();

enemyCarCollisionWithPlayer();

enemyFiring();

enemyCar1Firing();

enemyCar2Firing();

C1bulletCollisionWithPlayer();

C2bulletCollisionWithPlayer();

timer++;

Sleep(10);

}

}

void hardMode()

{

system("cls");

printRoad();

printInGameFooter();

printInstructions();

printCar();

printECar();

printE1Car();

printE2Car();

printEnemy();

// generateEBullet();

// moveEBullet();

state1 = true;

while (state1 != false)

{

back\_option();

bulletsCapacity();

printScore();

print\_enemyHealth();

printLife();

printBullets();

if (GetAsyncKeyState(VK\_LEFT))

{

moveCarLeft();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveCarRight();

}

if (GetAsyncKeyState(VK\_UP))

{

moveCarUp();

}

if (GetAsyncKeyState(VK\_DOWN))

{

moveCarDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

generateBullet();

}

if (GetAsyncKeyState(VK\_NUMPAD0))

{

generateEBullet();

}

if (timer == 3)

{

moveEnemy();

moveEnemyCar1();

moveEnemyCar2();

// moveEnemyCar3();

moveCarSmart();

timer = 0;

}

moveBullet();

moveEBullet();

moveEC1Bullet();

moveEC2Bullet();

bulletCollisionWithEnemy();

bulletCollisionWithPlayer();

enemyCar1CollisionWithPlayer();

enemyCar2CollisionWithPlayer();

enemyCar3CollisionWithPlayer();

enemyCarCollisionWithPlayer();

enemyCar1Firing();

enemyCar2Firing();

C1bulletCollisionWithPlayer();

C2bulletCollisionWithPlayer();

timer++;

Sleep(10);

}

}

void basicMode()

{

system("cls");

state1 = true;

printTankRoad();

printTank();

printTankEnemy();

while (state1 != false)

{

back\_option();

printTankScore();

if (GetAsyncKeyState(VK\_LEFT))

{

moveTankLeft();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveTankRight();

}

if (GetAsyncKeyState(VK\_UP))

{

moveTankUp();

}

if (GetAsyncKeyState(VK\_DOWN))

{

moveTankDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

generateTankBullet();

}

if (timer == 3)

{

moveTankEnemy();

timer = 0;

}

moveTankBullet();

moveEnemyTankBullet();

TankbulletCollisionWithEnemy();

timer++;

Sleep(10);

}

}

void easyMode(void)

{

system("cls");

printRoad();

printInGameFooter();

printInstructions();

printCar();

printEnemy();

state1 = true;

while (state1 != false)

{

back\_option();

bulletsCapacity();

printScore();

print\_enemyHealth();

printLife();

printBullets();

if (GetAsyncKeyState(VK\_LEFT))

{

moveCarLeft();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveCarRight();

}

if (GetAsyncKeyState(VK\_UP))

{

moveCarUp();

}

if (GetAsyncKeyState(VK\_DOWN))

{

moveCarDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

generateBullet();

}

if (GetAsyncKeyState(VK\_NUMPAD0))

{

generateEBullet();

}

if (timer == 3)

{

moveEnemy();

timer = 0;

}

moveBullet();

moveEBullet();

bulletCollisionWithEnemy();

bulletCollisionWithPlayer();

enemyCarCollisionWithPlayer();

timer++;

Sleep(10);

}

}

void mediumMode(void)

{

system("cls");

printRoad();

printInGameFooter();

printInstructions();

printCar();

printECar();

printE1Car();

printEnemy();

state1 = true;

while (state1 != false)

{

back\_option();

bulletsCapacity();

printScore();

print\_enemyHealth();

printLife();

printBullets();

if (GetAsyncKeyState(VK\_LEFT))

{

moveCarLeft();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveCarRight();

}

if (GetAsyncKeyState(VK\_UP))

{

moveCarUp();

}

if (GetAsyncKeyState(VK\_DOWN))

{

moveCarDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

generateBullet();

}

if (GetAsyncKeyState(VK\_NUMPAD0))

{

generateEBullet();

}

if (timer == 3)

{

moveEnemy();

moveEnemyCar1();

moveEnemyCar2();

// moveEnemyCar3();

timer = 0;

}

moveBullet();

moveEBullet();

moveEC1Bullet();

moveEC2Bullet();

bulletCollisionWithEnemy();

bulletCollisionWithPlayer();

enemyCar1CollisionWithPlayer();

enemyCar2CollisionWithPlayer();

// enemyCar3CollisionWithPlayer();

enemyCarCollisionWithPlayer();

enemyCar1Firing();

enemyCar2Firing();

C1bulletCollisionWithPlayer();

C2bulletCollisionWithPlayer();

timer++;

Sleep(10);

}

}

void printScore()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 11);

gotoxy(165, 7);

cout << " Score: " << score;

}

void print\_enemyHealth()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 7);

gotoxy(179, 5);

cout << " Enemy Health: " << health << " ";

}

void printLife()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 10);

gotoxy(179, 7);

cout << " Life: " << life << " ";

}

void printBullets()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 6);

gotoxy(192, 7);

// store\_bullets=bulletCounter;

cout << " Bullets: " << bulletCounter << " ";

}

void bulletsCapacity()

{

if (bulletCounter <= 0)

{

printLostMenu2();

}

}

void moveEnemy()

{

if (enemyDirection == "Up")

{

char next = getCharAtxy(enemyX, enemyY - 1);

if (next == ' ')

{

eraseEnemy();

enemyY--;

printEnemy();

}

if (next == '#')

{

enemyDirection = "Down";

}

}

if (enemyDirection == "Down")

{

char next = getCharAtxy(enemyX, enemyY + 5);

if (next == ' ')

{

eraseEnemy();

enemyY++;

printEnemy();

}

if (next == '#')

{

enemyDirection = "Up";

}

}

}

void printEnemy()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 4);

gotoxy(enemyX, enemyY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 6; j++)

{

cout << enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 3);

for (int i = 3; i < 4; i++)

{

for (int j = 0; j < 6; j++)

{

cout << enemy[i][j];

}

cout << "\n";

}

}

void eraseEnemy()

{

gotoxy(enemyX, enemyY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_Enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_Enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_Enemy[i][j];

}

cout << "\n";

}

gotoxy(enemyX, enemyY + 3);

for (int i = 3; i < 4; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_Enemy[i][j];

}

cout << "\n";

}

}

void moveCarLeft()

{

char next = getCharAtxy(carX - 1, carY);

if (next == ' ')

{

eraseCar();

carX = carX - 1;

printCar();

}

}

void moveCarRight()

{

char next = getCharAtxy(carX + 7, carY);

if (next == ' ')

{

eraseCar();

carX = carX + 1;

printCar();

}

}

void moveCarUp()

{

char next = getCharAtxy(carX, carY - 1);

if (next == ' ')

{

eraseCar();

carY = carY - 1;

printCar();

}

}

void moveCarDown()

{

char next = getCharAtxy(carX, carY + 3);

if (next == ' ')

{

eraseCar();

carY = carY + 1;

printCar();

}

}

void generateBullet()

{

bulletX[bulletCount] = carX + 7;

bulletY[bulletCount] = carY;

isBulletActive[bulletCount] = true;

gotoxy(carX + 7, carY);

cout << "o";

bulletCount++;

bulletCounter--;

}

void generateEBullet()

{

EbulletX[EbulletCount] = enemyX - 7;

EbulletY[EbulletCount] = enemyY;

isEBulletActive[EbulletCount] = true;

gotoxy(enemyX - 7, enemyY);

cout << "o";

EbulletCount++;

}

void generateEC1Bullet()

{

EC1bulletX[EC1bulletCount] = enemyCar1X - 7;

EC1bulletY[EC1bulletCount] = enemyCar1Y;

isEC1BulletActive[EC1bulletCount] = true;

gotoxy(enemyCar1X - 7, enemyCar1Y);

cout << "o";

EC1bulletCount++;

}

void generateEC2Bullet()

{

EC2bulletX[EC2bulletCount] = enemyCar2X - 7;

EC2bulletY[EC2bulletCount] = enemyCar2Y;

isEC2BulletActive[EC2bulletCount] = true;

gotoxy(enemyCar2X - 7, enemyCar2Y);

cout << "o";

EC2bulletCount++;

}

void removeBulletFromArray(int index)

{

for (int x = index; x < bulletCount - 1; x++)

{

bulletX[x] = bulletX[x + 1];

bulletY[x] = bulletY[x + 1];

}

bulletCount--;

}

void removeEBulletFromArray(int index)

{

for (int x = index; x < EbulletCount - 1; x++)

{

EbulletX[x] = EbulletX[x + 1];

EbulletY[x] = EbulletY[x + 1];

}

EbulletCount--;

}

void removeEC1BulletFromArray(int index)

{

for (int x = index; x < EC1bulletCount - 1; x++)

{

EC1bulletX[x] = EC1bulletX[x + 1];

EC1bulletY[x] = EC1bulletY[x + 1];

}

EC1bulletCount--;

}

void removeEC2BulletFromArray(int index)

{

for (int x = index; x < EC2bulletCount - 1; x++)

{

EC2bulletX[x] = EC2bulletX[x + 1];

EC2bulletY[x] = EC2bulletY[x + 1];

}

EC2bulletCount--;

}

void eraseCar()

{

gotoxy(carX, carY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(carX, carY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(carX, carY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

}

void gotoxy(int x, int y)

{

COORD coordinates;

coordinates.X = x;

coordinates.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);

}

void printRoad()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 14);

for (int i = 0; i < 41; i++)

{

for (int j = 0; j < 164; j++)

{

cout << Game\_Road[i][j];

}

cout << "\n";

}

}

void moveBullet()

{

for (int x = 0; x < bulletCount; x++)

{

if (isBulletActive[x] == true)

{

char next = getCharAtxy(bulletX[x] + 1, bulletY[x]);

if (next != ' ')

{

eraseBullet(bulletX[x], bulletY[x]);

removeBulletFromArray(x);

}

else

{

eraseBullet(bulletX[x], bulletY[x]);

bulletX[x] = bulletX[x] + 1;

printBullet(bulletX[x], bulletY[x]);

}

}

}

}

void moveEBullet()

{

for (int x = 0; x < EbulletCount; x++)

{

if (isEBulletActive[x] == true)

{

char next = getCharAtxy(EbulletX[x] - 1, EbulletY[x]);

if (next != ' ')

{

eraseBullet(EbulletX[x], EbulletY[x]);

removeEBulletFromArray(x);

}

else

{

eraseBullet(EbulletX[x], EbulletY[x]);

EbulletX[x] = EbulletX[x] - 1;

printBullet(EbulletX[x], EbulletY[x]);

}

}

}

}

void moveEC1Bullet()

{

for (int x = 0; x < EC1bulletCount; x++)

{

if (isEC1BulletActive[x] == true)

{

char next = getCharAtxy(EC1bulletX[x] - 1, EC1bulletY[x]);

if (next != ' ')

{

eraseBullet(EC1bulletX[x], EC1bulletY[x]);

removeEC1BulletFromArray(x);

// makeEC1BulletInactive(x);

}

else

{

eraseBullet(EC1bulletX[x], EC1bulletY[x]);

EC1bulletX[x] = EC1bulletX[x] - 1;

printBullet(EC1bulletX[x], EC1bulletY[x]);

// Sleep(5);

}

}

}

}

void moveEC2Bullet()

{

for (int x = 0; x < EC2bulletCount; x++)

{

if (isEC2BulletActive[x] == true)

{

char next = getCharAtxy(EC2bulletX[x] - 1, EC2bulletY[x]);

if (next != ' ')

{

eraseBullet(EC2bulletX[x], EC2bulletY[x]);

removeEC2BulletFromArray(x);

}

else

{

eraseBullet(EC2bulletX[x], EC2bulletY[x]);

EC2bulletX[x] = EC2bulletX[x] - 1;

printBullet(EC2bulletX[x], EC2bulletY[x]);

// Sleep(5);

}

}

}

}

void printBullet(int x, int y)

{

gotoxy(x, y);

cout << "o";

}

void eraseBullet(int x, int y)

{

gotoxy(x, y);

cout << " ";

}

void makeBulletInactive(int index)

{

isBulletActive[index] = false;

}

void makeEBulletInactive(int index)

{

isEBulletActive[index] = false;

}

void makeEC1BulletInactive(int index)

{

isEC1BulletActive[index] = false;

}

void makeEC2BulletInactive(int index)

{

isEC2BulletActive[index] = false;

}

void bulletCollisionWithEnemy()

{

for (int x = 0; x < bulletCount; x++)

{

if (isBulletActive[x] == true)

{

if (bulletX[x] + 1 == enemyX && (bulletY[x] == enemyY || bulletY[x] == enemyY + 1 || bulletY[x] == enemyY + 2 || bulletY[x] == enemyY + 3))

{

addScore();

}

if (enemyX - 1 == bulletX[x] && enemyY + 1 == bulletY[x])

{

eraseBullet(bulletX[x], bulletY[x]);

removeBulletFromArray(x);

addScore();

}

}

}

}

void bulletCollisionWithPlayer()

{

for (int x = 0; x < EbulletCount; x++)

{

if (isEBulletActive[x] == true)

{

if ((EbulletX[x] - 7 == carX && EbulletY[x] == carY) || (EbulletX[x] - 7 == carX && EbulletY[x] == carY + 1) || (EbulletX[x] - 7 == carX && EbulletY[x] == carY + 2))

{

decreaseLife();

}

// if (carX + 1 == EbulletX[x] && carY + 1 == EbulletY[x])

// {

// eraseBullet(EbulletX[x], EbulletY[x]);

// removeEBulletFromArray(x);

// decreaseLife();

// }

}

}

}

void C1bulletCollisionWithPlayer()

{

for (int x = 0; x < EC1bulletCount; x++)

{

if (isEC1BulletActive[x] == true)

{

if ((EC1bulletX[x] + 6 == carX || EC1bulletX[x] == carX || EC1bulletX[x] - 6 == carX) && (EC1bulletY[x] == carY))

{

decreaseLife();

}

else if (carX + 1 == EC1bulletX[x] && carY + 1 == EC1bulletY[x])

{

eraseBullet(EC1bulletX[x], EC1bulletY[x]);

removeEC1BulletFromArray(x);

decreaseLife();

}

}

}

}

void C2bulletCollisionWithPlayer()

{

for (int x = 0; x < EC2bulletCount; x++)

{

if (isEC2BulletActive[x] == true)

{

if ((EC2bulletX[x] + 1 == carX || EC2bulletX[x] == carX) && (EC2bulletY[x] == carY || EC2bulletY[x] == carY + 1 || EC2bulletY[x] == carY + 2 || EC2bulletY[x] == carY + 3 || EC2bulletY[x] == carY - 1 || EC2bulletY[x] - 1 == carY))

{

decreaseLife();

}

if (carX + 1 == EC2bulletX[x] && carY + 1 == EC2bulletY[x])

{

eraseBullet(EC2bulletX[x], EC2bulletY[x]);

removeEC2BulletFromArray(x);

decreaseLife();

}

}

}

}

void addScore()

{

score++;

health--;

store\_score = score;

if (score >= 20)

{

printWonMenu();

}

}

void decreaseLife()

{

life--;

store\_life = life;

if (life <= 0)

{

printLostMenu();

}

}

char getCharAtxy(short int x, short int y)

{

CHAR\_INFO ci;

COORD xy = {0, 0};

SMALL\_RECT rect = {x, y, x, y};

COORD coordBufSize;

coordBufSize.X = 1;

coordBufSize.Y = 1;

return ReadConsoleOutput(GetStdHandle(STD\_OUTPUT\_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar : ' ';

}

void printCar()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 3);

gotoxy(carX, carY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car[i][j];

}

cout << "\n";

}

gotoxy(carX, carY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car[i][j];

}

cout << "\n";

}

gotoxy(carX, carY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car[i][j];

}

cout << "\n";

}

}

void printECar()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 12);

gotoxy(enemyCar1X, enemyCar1Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car1[i][j];

}

cout << "\n";

}

gotoxy(enemyCar1X, enemyCar1Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car1[i][j];

}

cout << "\n";

}

gotoxy(enemyCar1X, enemyCar1Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car1[i][j];

}

cout << "\n";

}

}

void eraseECar()

{

gotoxy(enemyCar1X, enemyCar1Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar1X, enemyCar1Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar1X, enemyCar1Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

}

void printE1Car()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 12);

gotoxy(enemyCar2X, enemyCar2Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car2[i][j];

}

cout << "\n";

}

gotoxy(enemyCar2X, enemyCar2Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car2[i][j];

}

cout << "\n";

}

gotoxy(enemyCar2X, enemyCar2Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car2[i][j];

}

cout << "\n";

}

}

void printE2Car()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 12);

gotoxy(enemyCar3X, enemyCar3Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car3[i][j];

}

cout << "\n";

}

gotoxy(enemyCar3X, enemyCar3Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car3[i][j];

}

cout << "\n";

}

gotoxy(enemyCar3X, enemyCar3Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << car3[i][j];

}

cout << "\n";

}

}

void eraseE1Car()

{

gotoxy(enemyCar2X, enemyCar2Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar2X, enemyCar2Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar2X, enemyCar2Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

}

void eraseE2Car()

{

gotoxy(enemyCar3X, enemyCar3Y);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar3X, enemyCar3Y + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

gotoxy(enemyCar3X, enemyCar3Y + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 7; j++)

{

cout << erase\_Car[i][j];

}

cout << "\n";

}

}

void printHeader()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 224);

system("cls");

cout << endl

<< endl

<< endl;

cout << " #############################################################################################################################################################################################################" << endl;

cout << " /^^ /^ /^^^^^^^ /^^ /^^ /^^ /^^ /^^^^^^^ /^^^^^ /^^ /^^^^^^^^ /^^^ /^^^^^ /^^^^ /^ /^^ /^^ /^^^^^^^" << endl;

cout << " /^^ /^^ /^ ^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^ /^^ /^^ /^^ /^ /^^ /^ ^^ /^ /^^ /^^^ /^^" << endl;

cout << "/^^ /^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^ /^^ /^^ /^^ /^^ /^ /^^ /^^ /^^ / /^^ /^^" << endl;

cout << "/^^ /^^ /^^ /^ /^^ /^^^^^^ /^^ /^^ /^^ /^ /^^ /^^ /^^ /^^ /^^^^^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^^^^" << endl;

cout << "/^^ /^^^^^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^^^ /^^^^^^ /^^ /^^ /^ /^^ /^^" << endl;

cout << " /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^ /^ /^^ /^^ /^^ /^^ /^^" << endl;

cout << " /^^^^ /^^ /^^ /^^ /^^ /^^ /^^ /^^^^^ /^^ /^^ /^^^^^ /^^^^^^^^ /^^^^^^^^ /^^^^^^^^ /^^^^^ /^^^^^ /^^ /^^ /^^ /^^ /^^^^^^^" << endl;

cout << " ##############################################################################################################################################################################################################" << endl

<< endl

<< endl

<< endl;

}

void printMenuAndDifficultyLevelScreen()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 160);

string difficultyLevel = "\n";

system("cls");

printHeader();

printWelcomeMenu();

cout << " Enter the Difficulty Level " << endl

<< endl;

cout << " " << endl;

cout << " 1) Basic Level " << endl;

cout << " 2) Easy " << endl;

cout << " 3) Medium " << endl;

cout << " 4) Hard (Single Player) " << endl;

cout << " 5) Hard (Double Player) " << endl;

cout << " 6) Exit " << endl

<< endl

<< endl;

cout << " ==> Your choice : ";

cin >> difficultyLevel;

if (difficultyLevel == "1")

{

basicMode();

}

else if (difficultyLevel == "2")

{

easyMode();

}

else if (difficultyLevel == "3")

{

mediumMode();

}

else if (difficultyLevel == "4")

{

singlePlayerHardMode();

}

else if (difficultyLevel == "5")

{

hardMode();

}

else if (difficultyLevel == "6")

{

cout << " Enter any key to Exit ";

getch();

state1 = false;

}

else

{

invalidOption();

}

}

void printLostMenu()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 160);

system("cls");

printHeader();

cout << " ##################################################################################" << endl;

cout << " Better Luck Next Time" << endl;

cout << " ##################################################################################" << endl

<< endl

<< endl

<< endl

<< endl;

cout << " ######################" << endl;

cout << " # You Lost ! #" << endl;

cout << " ######################" << endl

<< endl

<< endl

<< endl

<< endl

<< endl;

choice();

}

void printLostMenu2()

{

// HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

// SetConsoleTextAttribute(hConsole, 78);

system("cls");

system("cls");

printHeader();

cout << " ##################################################################################" << endl;

cout << " Better Luck Next Time" << endl;

cout << " ##################################################################################" << endl

<< endl

<< endl

<< endl

<< endl;

cout << " #######################################" << endl;

cout << " # You are out of Bullets ! #" << endl;

cout << " #######################################" << endl

<< endl

<< endl

<< endl;

cout << " ######################" << endl;

cout << " # You Lost ! #" << endl;

cout << " ######################" << endl

<< endl

<< endl

<< endl

<< endl

<< endl;

// state1 = false;

choice();

}

void printWelcomeMenu()

{

system("cls");

printHeader();

cout << " ##################################################################################" << endl;

cout << " WELCOME TO CAR HURDLE 2D GAME" << endl;

cout << " ##################################################################################" << endl

<< endl

<< endl

<< endl;

// state1 = false;

}

void printWonMenu()

{

system("cls");

printHeader();

cout << " ##################################################################################" << endl;

cout << " You Played Well" << endl;

cout << " ##################################################################################" << endl

<< endl

<< endl

<< endl

<< endl;

cout << " #######################################" << endl;

cout << " # Hurray ! You Won the Game ! #" << endl;

cout << " #######################################" << endl

<< endl

<< endl

<< endl

<< endl

<< endl;

// state1 = false;

choice();

}

void enemyCar1CollisionWithPlayer()

{

if ((carX + 6 == enemyCar1X || carX + 1 == enemyCar1X + 1 || carX + 1 == enemyCar1X) && (carY + 1 == enemyCar1Y || carY == enemyCar1Y + 1 || carY - 1 == enemyCar1Y - 1))

{

printLostMenu();

}

}

void enemyCar1Firing()

{

if ((carX == enemyCar1X - 19 || carX + 19 == enemyCar1X - 1 || carX + 10 == enemyCar1X + 1) && (carY + 1 == enemyCar1Y || carY == enemyCar1Y + 1 || carY - 1 == enemyCar1Y - 1))

{

generateEC1Bullet();

}

}

void enemyCar2CollisionWithPlayer()

{

if ((carX + 6 == enemyCar2X || carX + 1 == enemyCar2X + 1 || carX + 1 == enemyCar2X) && (carY + 1 == enemyCar2Y || carY == enemyCar2Y + 1))

{

printLostMenu();

}

}

void enemyCar2Firing()

{

if ((carX == enemyCar2X - 19 || carX + 19 == enemyCar2X - 1 || carX + 10 == enemyCar2X + 1) && (carY + 1 == enemyCar2Y || carY == enemyCar2Y + 1 || carY - 1 == enemyCar2Y - 1))

{

generateEC2Bullet();

}

}

void enemyCar3CollisionWithPlayer()

{

if ((carX + 6 == enemyCar3X || carX + 1 == enemyCar3X + 1 || carX + 1 == enemyCar3X) && (carY + 1 == enemyCar3Y || carY == enemyCar3Y + 1))

{

printLostMenu();

}

}

void enemyFiring()

{

if ((carX == enemyX && (carX == enemyX || carX - 1 == enemyX || carX - 1 == enemyX - 1 || carX + 1 == enemyX + 1)) || (carY == enemyY && (carY == enemyY + 3 || carY + 3 == enemyY || carY - 3 == enemyY - 3)))

{

generateEBullet();

}

}

void enemyCarCollisionWithPlayer()

{

if ((carX + 6 == enemyX || carX == enemyX + 1) && (carY + 5 == enemyY || carY + 1 == enemyY + 1))

{

printLostMenu();

}

}

void moveEnemyCar1()

{

if (enemyCar1Direction == "Left")

{

char next = getCharAtxy(enemyCar1X - 82, enemyCar1Y);

if (next == ' ')

{

eraseECar();

enemyCar1X--;

printECar();

}

else

{

enemyCar1Direction = "Right";

}

}

if (enemyCar1Direction == "Right")

{

char next = getCharAtxy(enemyCar1X + 28, enemyCar1Y);

if (next == ' ')

{

eraseECar();

enemyCar1X++;

printECar();

}

else if (next == '#')

{

enemyCar1Direction = "Left";

}

}

}

void moveEnemyCar2()

{

if (enemyCar1Direction == "Left")

{

char next = getCharAtxy(enemyCar2X - 82, enemyCar2Y);

if (next == ' ')

{

eraseE1Car();

enemyCar2X--;

printE1Car();

}

else

{

enemyCar1Direction = "Right";

}

}

if (enemyCar1Direction == "Right")

{

char next = getCharAtxy(enemyCar2X + 28, enemyCar2Y);

if (next == ' ')

{

eraseE1Car();

enemyCar2X++;

printE1Car();

}

else if (next == '#')

{

enemyCar1Direction = "Left";

}

}

}

void printTankRoad()

{

for (int i = 0; i < 18; i++)

{

for (int j = 0; j < 42; j++)

{

cout << TankRoad[i][j];

}

cout << "\n";

}

}

void printTank()

{

gotoxy(tankX, tankY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << tank[i][j];

}

cout << "\n";

}

gotoxy(tankX, tankY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << tank[i][j];

}

cout << "\n";

}

}

void eraseTank()

{

gotoxy(tankX, tankY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_tank[i][j];

}

cout << "\n";

}

gotoxy(tankX, tankY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_tank[i][j];

}

cout << "\n";

}

}

void moveTankLeft()

{

char next = getCharAtxy(tankX - 1, tankY);

if (next == ' ')

{

eraseTank();

tankX = tankX - 1;

printTank();

}

}

void moveTankRight()

{

char next = getCharAtxy(tankX + 6, tankY);

if (next == ' ')

{

eraseTank();

tankX = tankX + 1;

printTank();

}

}

void moveTankUp()

{

char next = getCharAtxy(tankX, tankY - 1);

if (next == ' ')

{

eraseTank();

tankY = tankY - 1;

printTank();

}

}

void moveTankDown()

{

char next = getCharAtxy(tankX, tankY + 2);

if (next == ' ')

{

eraseTank();

tankY = tankY + 1;

printTank();

}

}

void TankbulletCollisionWithEnemy()

{

for (int x = 0; x < bulletCount; x++)

{

if (isBulletActive[x] == true)

{

if (tankBulletX[x] + 1 == TankEnemyX && (tankBulletY[x] == TankEnemyY || tankBulletY[x] == TankEnemyY + 1 || tankBulletY[x] == TankEnemyY + 2 || tankBulletY[x] == TankEnemyY + 3))

{

addScore();

}

if (TankEnemyX - 1 == tankBulletX[x] && TankEnemyY + 1 == tankBulletY[x])

{

addScore();

}

}

}

}

void moveEnemyTankBullet()

{

for (int x = 0; x < enemyTankBulletCount; x++)

{

if (isEnemyTankBulletActive[x] == true)

{

char next = getCharAtxy(enemyTankBulletX[x] - 1, enemyTankBulletY[x]);

if (next != ' ')

{

eraseBullet(enemyTankBulletX[x], enemyTankBulletY[x]);

removeTankBulletFromArray(x);

}

else

{

eraseBullet(enemyTankBulletX[x], enemyTankBulletY[x]);

enemyTankBulletX[x] = enemyTankBulletX[x] - 1;

printBullet(enemyTankBulletX[x], enemyTankBulletY[x]);

}

}

}

}

void moveTankBullet()

{

for (int x = 0; x < bulletCount; x++)

{

if (isBulletActive[x] == true)

{

char next = getCharAtxy(tankBulletX[x] + 1, tankBulletY[x]);

if (next != ' ')

{

eraseBullet(tankBulletX[x], tankBulletY[x]);

removeTankBulletFromArray(x);

}

else

{

eraseBullet(tankBulletX[x], tankBulletY[x]);

tankBulletX[x] = tankBulletX[x] + 1;

printBullet(tankBulletX[x], tankBulletY[x]);

}

}

}

}

void removeTankBulletFromArray(int index)

{

for (int x = index; x < bulletCount - 1; x++)

{

tankBulletX[x] = tankBulletX[x + 1];

tankBulletY[x] = tankBulletY[x + 1];

}

bulletCount--;

}

void generateTankEnemyBullet()

{

enemyTankBulletX[enemyTankBulletCount] = TankEnemyX - 7;

enemyTankBulletY[enemyTankBulletCount] = TankEnemyY;

isEnemyTankBulletActive[enemyTankBulletCount] = true;

gotoxy(TankEnemyX - 7, TankEnemyY);

cout << "o";

enemyTankBulletCount++;

}

void generateTankBullet()

{

tankBulletX[bulletCount] = tankX + 7;

tankBulletY[bulletCount] = tankY;

isBulletActive[bulletCount] = true;

gotoxy(tankX + 7, tankY);

cout << "o";

bulletCount++;

bulletCounter--;

}

void eraseTankEnemy()

{

gotoxy(TankEnemyX, TankEnemyY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 3);

for (int i = 3; i < 4; i++)

{

for (int j = 0; j < 6; j++)

{

cout << erase\_TankEnemy[i][j];

}

cout << "\n";

}

}

void printTankEnemy()

{

// HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

// SetConsoleTextAttribute(hConsole, 4);

gotoxy(TankEnemyX, TankEnemyY);

for (int i = 0; i < 1; i++)

{

for (int j = 0; j < 6; j++)

{

cout << TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 1);

for (int i = 1; i < 2; i++)

{

for (int j = 0; j < 6; j++)

{

cout << TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 2);

for (int i = 2; i < 3; i++)

{

for (int j = 0; j < 6; j++)

{

cout << TankEnemy[i][j];

}

cout << "\n";

}

gotoxy(TankEnemyX, TankEnemyY + 3);

for (int i = 3; i < 4; i++)

{

for (int j = 0; j < 6; j++)

{

cout << TankEnemy[i][j];

}

cout << "\n";

}

}

void moveTankEnemy()

{

if (tankEnemyDirection == "Up")

{

char next = getCharAtxy(TankEnemyX, TankEnemyY - 1);

if (next == ' ')

{

eraseTankEnemy();

TankEnemyY--;

printTankEnemy();

}

if (next == '#')

{

tankEnemyDirection = "Down";

}

}

if (tankEnemyDirection == "Down")

{

char next = getCharAtxy(TankEnemyX, TankEnemyY + 4);

if (next == ' ')

{

eraseTankEnemy();

TankEnemyY++;

printTankEnemy();

}

if (next == '#')

{

tankEnemyDirection = "Up";

}

}

}

void printTankScore()

{

gotoxy(48, 8);

cout << "Score: " << score;

}

void printInGameFooter()

{

gotoxy(6, 40);

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 14);

cout << endl

<< endl;

cout << " \_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_\_ \_\_ \_\_\_\_\_\_ " << endl;

cout << " / \_\_\_\_/ \_\_\_\_ \_ \_\_\_\_\_ / / / / \_\_ \_\_ \_\_\_\_\_ \_\_\_\_/ / / / \_\_\_ |\_\_ \\ \_\_\_\_/ / / \_\_\_\_/ \_\_\_\_ \_ \_\_\_\_ \_\_\_ \_\_\_ " << endl;

cout << " / / / \_\_ `/ / \_\_\_/ / /\_/ / / / / / / \_\_\_/ / \_\_ / / / / \_ \\ \_\_/ / / \_\_ / / / \_\_ / \_\_ `/ / \_\_ `\_\_ \\ / \_ \\ " << endl;

cout << " / /\_\_\_ / /\_/ / / / / \_\_ / / /\_/ / / / / /\_/ / / / / \_\_/ / \_\_/ / /\_/ / / /\_/ / / /\_/ / / / / / / // \_\_/ " << endl;

cout << " \\\_\_\_\_/ \\\_\_,\_/ /\_/ /\_/ /\_/ \\\_\_,\_/ /\_/ \\\_\_,\_/ /\_/ \\\_\_\_/ /\_\_\_\_/ \\\_\_,\_/ \\\_\_\_\_/ \\\_\_,\_/ /\_/ /\_/ /\_/ \\\_\_\_/ " << endl;

cout << " " << endl

<< endl;

}

void printInstructions()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 14);

gotoxy(165, 9);

cout << "Instructions : " << endl;

gotoxy(165, 10);

cout << "1)Use Arrow Keys to move Player." << endl;

gotoxy(165, 11);

cout << "2)Press SpaceBar to fire Player bullets." << endl;

gotoxy(165, 12);

cout << "3)Press 0 Key to fire moving enemy bullets." << endl;

gotoxy(165, 13);

cout << "4)Reach 20 Score by using 100 bullets." << endl;

gotoxy(165, 14);

cout << "5)Press Esc to STOP game." << endl;

}