**Object Oriented Programming**

**Assessment**

Name: SamiUlah Registration No: 2022\_Cs-143

**Guideline:** You are only allowed to make changes in CollisionDetector.cs, form\_load (function), GameGhostHorizontal and GameGhostChaser.cs class.

**Past the code that you inserted in this document under every question**

**Q No 1: Increase score by one point if pacman collide with the pallet.**

Past your code here

**Q No 2: Decrease score by one point if pacman collide with the Ghost.**

Q No 3: add a Horizontal moving Ghost

public class GameGhostHorizontal : GameGhost

{

private GameDirection direction;

public GameGhostHorizontal(Image img, GameCell start) : base(img)

{

this.direction = GameDirection.Right;

}

public override GameCell move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

if (nextCell == currentCell && direction == GameDirection.Right)

{

direction = GameDirection.Left;

}

else if (nextCell == currentCell && direction == GameDirection.Left)

{

direction = GameDirection.Right;

}

if (nextCell != currentCell)

{

nextCell.setGameObject(this);

this.CurrentCell = nextCell;

currentCell.setGameObject(Game.getBlankGameObject());

}

return currentCell.nextCell(direction);

}

}

**Q No 4: add a ChasingGhost**

public class GameGhostChaser : GameGhost

{

GameDirection direction;

public GameDirection Direction { get => direction; set => direction = value; }

public GameGhostChaser(Image ghostImage, GameCell startCell) : base(ghostImage)

{

this.Direction = GameDirection.Left;

}

public GameCell move()

{

double[] distance = new double[4] { 1000000, 1000000, 1000000, 1000000 };

GameCell cell = CurrentCell.nextCell(Direction);

if (cell != null)

{

distance[0] = calculateDistance(cell.X, cell.Y - 1, cell.CurrentGameObject.CurrentCell.X,cell.CurrentGameObject.CurrentCell.Y);

distance[1] = calculateDistance(cell.X, cell.Y + 1, cell.CurrentGameObject.CurrentCell.X, cell.CurrentGameObject.CurrentCell.Y);

distance[2] = calculateDistance(cell.X + 1, cell.Y, cell.CurrentGameObject.CurrentCell.X, cell.CurrentGameObject.CurrentCell.Y);

distance[3] = calculateDistance(cell.X - 1, cell.Y - 1, cell.CurrentGameObject.CurrentCell.X, cell.CurrentGameObject.CurrentCell.Y);

if (distance[0] <= distance[1] && distance[0] <= distance[2] && distance[0] <= distance[3])

{

Direction = GameDirection.Left;

}

else if (distance[1] <= distance[0] && distance[1] <= distance[2] && distance[1] <= distance[3])

{

Direction = GameDirection.Right;

}

else if (distance[2] <= distance[0] && distance[2] <= distance[1] && distance[2] <= distance[3])

{

Direction = GameDirection.Down;

}

else

{

Direction = GameDirection.Up;

}

}

return cell;

}

static double calculateDistance(int x, int Y, int pX, int pY)

{

return Math.Sqrt(Math.Pow((pX - x), 2) + Math.Pow((pY - Y), 2));

}

}