1. a. I. To control a non-player character within the game

Ii. Sensing stage is when an AI is roaming, patroling or inactiving while waiting for a player character to come near the AI

Monster\_Transitions sense\_surroundings()

{

if (theManagerAsksToRoam()) return Monster\_Transitions.Go\_Roaming;

if (IcanSeePlayer()) return Monster\_Transitions.See\_Player;

if (IamnextToPlayer()) return Monster\_Transitions.Catch\_player;

return Monster\_Transitions.None;

}

private bool IamnextToPlayer()

{

throw new NotImplementedException();

}

private bool IcanSeePlayer()

{

throw new NotImplementedException();

}

private bool theManagerAsksToRoam()

{

throw new NotImplementedException();

}

Thinking Stage is when an AI is following the player

Monster\_State next\_State(Monster\_State current\_State, Monster\_Transitions transition )

{

switch(current\_State)

{

case Monster\_State.Inactive:

switch (transition)

{

case Monster\_Transitions.Catch\_player:

return Monster\_State.Attacking;

case Monster\_Transitions.See\_Player:

return Monster\_State.Chasing;

case Monster\_Transitions.Go\_Roaming:

return Monster\_State.Roaming;

}

return Monster\_State.Roaming;

break;

case Monster\_State.Roaming:

switch(transition)

{

case Monster\_Transitions.Catch\_player:

return Monster\_State.Attacking;

case Monster\_Transitions.See\_Player:

return Monster\_State.Chasing;

}

return Monster\_State.Roaming;

case Monster\_State.Chasing:

break;

case Monster\_State.Attacking

break;

}

}

Action Stage is when an AI is in combat with the player character or another AI

void implement(Monster\_State current\_State)

{

switch (current\_State)

{

case Monster\_State.Inactive:

// Do nothing

break;

case Monster\_State.Roaming:

//Choose random direction

// Move character in that direction

break;

case Monster\_State.Chasing:

// Move monster towards Character

break;

case Monster\_State.Attacking:

// Hit Character

break;

}

}

}

Iii FSM is simple to use and implement however the AI is stupid

Iv.

B I. In a Search Problem the is unpredictable

Ii A utility function assigns values to certain actions that the AI can do to give it a good or bad outcome

Iii An evaluation function allows the AI to decide whether a move is good or bad

Iv Evaluation can be used in a search tree problem by evaluating each branch to decide the the best route

V There are far too many possibilities in a game like chess or draughts. Performance can be improved by removing branches

Vi 3,5 8,8,-2 6,9

3 -2 6

6