# 2010 AP Computer Science A Problem #4

## Part (a):

public Actor actorWithMostNeighbors() {

if (0 == this.gr.getOccupiedLocations().size()) {

return null;

}

Location where = null;

int most = -1;

for (Location loc : this.gr.getOccupiedLocations()) {

if (most < this.gr.getOccupiedAdjacentLocations(loc).size()) {

most = this.gr.getOccupiedAdjacentLocations(loc).size();

where = loc;

}

}

return this.gr.get(where);

}

## // Alternative solution (uses getNeighbors):

public Actor actorWithMostNeighbors() {

if (0 == this.gr.getOccupiedLocations().size()) {

return null;

}

Location where = this.gr.getOccupiedLocations().get(0);

for (Location loc : this.gr.getOccupiedLocations()) {

if (this.gr.getNeighbors(where).size() < this.gr.getNeighbors(loc).size()) {

where = loc;

}

}

return this.gr.get(where);

}

## Part (b):

public List<Location> getOccupiedWithinTwo(Location loc) {

List<Location> occupied = new ArrayList<Location>();

for (int row = loc.getRow() - 2; row <= loc.getRow() + 2; row++) {

for (int col = loc.getCol() - 2; col <= loc.getCol() + 2; col++) {

Location loc1 = new Location(row, col);

if (gr.isValid(loc1) && this.gr.get(loc1) != null && !loc1.equals(loc)) {

occupied.add(loc1);

}

}

}

return occupied;

}

## // Alternative solution (uses getOccupiedLocations):

public List<Location> getOccupiedWithinTwo(Location loc) {

List<Location> occupied = new ArrayList<Location>();

for (Location loc1 : this.gr.getOccupiedLocations()) {

if ((Math.abs(loc.getRow() - loc1.getRow()) <= 2) &&

(Math.abs(loc.getCol() - loc1.getCol()) <= 2) &&

!loc1.equals(loc)) {

occupied.add(loc1);

}

}

return occupied;

}

# 2011 AP Computer Science A Problem #3

## Part (a):

public int nextTankToFill(int threshold) {

int minLevel = this.tanks.get(0).getFuelLevel();

int minTankIndex = 0;

for (int i = 1; i < this.tanks.size(); i++) {

if (this.tanks.get(i).getFuelLevel() < minLevel) {

minLevel = this.tanks.get(i).getFuelLevel();

minTankIndex = i;

}

}

if (minLevel <= threshold) {

return minTankIndex;

}

else {

return this.filler.getCurrentIndex();

}

}

// Alternative solution

public int nextTankToFillA(int threshold) {

int minTankIndex = this.filler.getCurrentIndex();

for (int i = 0; i < this.tanks.size(); i++) {

if (this.tanks.get(i).getFuelLevel() <= threshold &&

this.tanks.get(i).getFuelLevel() <

this.tanks.get(minTankIndex).getFuelLevel()) {

minTankIndex = i;

}

}

return minTankIndex;

}

## Part (b):

public void moveToLocation(int locIndex) {

if (this.filler.getCurrentIndex() > locIndex) {

if (this.filler.isFacingRight()) {

this.filler.changeDirection();

}

this.filler.moveForward(this.filler.getCurrentIndex() - locIndex);

}

if (this.filler.getCurrentIndex() < locIndex) {

if (!this.filler.isFacingRight()) {

this.filler.changeDirection();

}

this.filler.moveForward(locIndex - this.filler.getCurrentIndex());

}

}