Selina Wang Genetic Picture Report : Oct. 27Th 2020.

Q .1 Score routine :

I implemented this function by using some of the ideas outlined in your assignment report.
i.e I used the "realisticness" of the outputted bitmap at each iteration as what to score upon.
So when eg. Water – sea is surrounded by land – as this does not occur in reality I scored this down by 1 unit.

When land was surrounded immediately by water – this population's score was decremented .
In all of these sub cases I defined "near" [i.e water should not be surrounded near a land position (x,y)] as the immediate locations above, below , left and right of the location (x,y) in Q. These would thus be denoted as [x - 1, y] ,[x + 1, y], [x, y + 1], [x, y – 1] .

Also I coded the term "mostly surrounded by" as 3/4 s of these locations in a certain state (i.e water = 2 or 0). So if 3 out of 4 of these locations were in the state mentioned a scoring up or down would occur.

- land connected to mostly other land (say at least 75 % of horizontal & vert. cells) - score up by 7.

- Ensure the percentage of fresh h2o in the picture is less than 4 %.Score up or down by 2 accordingly.
- Ensure the percentage of land in the picture is less than 70% -- also try to encourage land to at least be greater than 35% of the pic's area. Score up or down by 7.

- A piece of land mostly (75%) surrounded by sea or fresh water. Using the 'phenoSum' – a sum of > 6 or < 2 would indicted this criterion.
- Water surrounded by land again is unrealistic and so it is scored down accordingly as with other criteria.
- Look at each cell If the cell is freshwater and more or less surrounded by Sea, score it down 2
- A stricter definition of land connected to mostly other land (say at least 75 % of ALL connecting cells) - score up or down accordingly by 9.

- Look at each cell If the cell is Sea and more or less surrounded by freshwater, score it down
- Look at each cell If the cell is water and more or less surrounded by water, score it up 6

Code list:

I built and ran the code in 'Visual Studio 2017”. This project was created/opened as “generateLandnSea.csproj”. On a Windows 10 machine.

A snippet of the code I wrote for the setScore routine shown here :

I also added a helper function within the phenotype class to process that boundaries of the image as this code became quite clunky to be all in one function.

```
 public int setScore()
        {
            int local_score = 0;

            int seaCount = 0;
            int landCount = 0;
            int freshCount = 0;

            //store %s of the pic - land, sea and frechH20 - use these against the Global % contraints on the
terrain for scoring also below
            double landPercent = 0.0;
            double seaPercent = 0.0;
            double freshPercent = 0.0;

            int every = 1;

            int smallDelta = 1;
            int intermedMediumDelta = 2;
            int mediumDelta = 3;
            int largeDelta = 5;
            int intermediatelargeDeta = 6;
            int vLargeDelta = 7;


            for (int x = 0; x < Params.dimX; x = x + every)
            {
                for (int y = 0; y < Params.dimY; y++){  // { pheno[x,y] = 0; } // initialise to 0

                    // TODO :
                    if (getTerrainSafe(x, y) != 0) {  // && getTerrainSafe(x, y) && getTerrainSafe(x, y) &&
getTerrainSafe(x, y) && getTerrainSafe(x, y) && getTerrainSafe(x, y) && getTerrainSafe(x, y)

                        if ( (x == Params.dimX - 1 )|| (y == Params.dimY - 1) || (x == 0) || (y == 0)) {

                                scoringBordersHelperFunc(x, y);
                        }

                        // TODO : put in equivalents to below's logic
here !               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                        else
                        {

                            //Console.WriteLine(z);
                            //create a summation for these'conecting' pixels which can be used in the checks
ahead...
                            int phenoTerrainSum = pheno[x - 1, y] + pheno[x + 1, y] + pheno[x, y + 1] + pheno[x,
y - 1];


                            //ensure the percentage of land in the picture is less than 70% -- also try to
```

```
encourage land to at least be greater than 35% of th epic's area.
                            if(landPercent > Params.percentLand || landPercent < 0.35) {

                                score -= largeDelta;
                            }

                            if (landPercent < Params.percentLand && landPercent > 0.35)
                            {

                                score += vLargeDelta;
                            }

                            //ensure the percentage of land in the picture is less than 70%
                            if (freshPercent > Params.percentFresh) {

                                score -= intermediatelargeDeta;
                            }

                            //ensure the percentage of land in the picture is less than 70%
                            if (freshPercent < Params.percentFresh)
                            {

                                score += intermediatelargeDeta;
                            }

                            // Just focus on the simpler  horizontal and vertical connecting
                            //current is on land and one neighbour is water - score up
                            if (pheno[x, y] == 1)
                            {
                                landCount++;
                                if ((landCount + seaCount + freshCount) != 0)
                                {
                                    landPercent = (landCount) / (landCount + seaCount + freshCount);
                                }
                                else
                                {
                                    landPercent = 0.0;
                                }

                                if (pheno[x - 1, y] != 1 | pheno[x + 1, y] != 1 || pheno[x, y + 1] != 1 ||
pheno[x, y - 1] != 1)
                                {
                                    //Just print for debug puposes
                                    //Console.WriteLine("1st if block :  in land and one neighbour is water -
score up part of setScore func. ");
                                    score += smallDelta;
                                }


                            // land surrounded by water - score down
                            // these sums indicted > 6 or < 2 would correspond to a piece of land mostly (75%)
surrounded by sea or fresh water.
                            if (pheno[x, y] == 1 && (phenoTerrainSum > 6 | phenoTerrainSum < 2))
                            {
                                //Console.WriteLine("2nd if block :  in land and one neighbour is water - score
up part of setScore func. ");
                                score -= intermedMediumDelta;
                                local_score -= intermedMediumDelta;
                            }

                            //land connected to mostly other land (say at least 75 % of horizonal & vert. cells)
- score up
                            if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y
+ 1] == 1 && pheno[x, y - 1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1]
== 1 && pheno[x, y + 1] != 1)
                                            || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1]
== 1 && pheno[x - 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] == 1)))
```

```
                            {
                                //Console.WriteLine(" 3rd if block - land connected to mostly other land - score
up part of setScore func. ");
                                score += largeDelta;

                            }


                            //land connected to mostly other land- this time all 8 locations 'touching it' (say
at least 75 % of horizonal & vert. cells) - score up
                            if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y
+ 1] == 1 && pheno[x, y - 1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1]
== 1 && pheno[x - 1, y] == 1 &&
                                            pheno[x + 1, y+1] == 1 && pheno[x+1, y - 1] == 1 && pheno[x - 1, y -
1] == 1 && pheno[x-1, y + 1] != 1)
                                            || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y + 1] == 1 &&
                                            pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x - 1,
y-1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 &&  pheno[x + 1, y + 1] == 1 &&
                                            pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x + 1,
y-1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] == 1  &&
                                            pheno[x + 1, y + 1] == 1 && pheno[x + 1, y - 1] == 1 && pheno[x - 1,
y - 1] == 1 && pheno[x - 1, y + 1] == 1)))
                            {
                                //Console.WriteLine(" 3rd if block - land connected to mostly other land - score
up part of setScore func. ");
                                score += 9;
                                local_score += 9;
                            }

                            //  If the cell is water  and more or less surrounded by land,  score it down
                            //Splitting this into two separate loops so that i can increment the sea and
freshwater counts properly.
                            if (pheno[x, y] == 0)
                            {
                                seaCount++;

                                if ((landCount + seaCount + freshCount) != 0) {
                                    seaPercent = (seaCount) / (landCount + seaCount + freshCount);
                                }
                                else
                                {
                                    seaPercent = 0.0;
                                }



                                if ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 &&
pheno[x, y - 1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1]
== 1 && pheno[x, y + 1] != 1)
                                            || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1]
== 1 && pheno[x - 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] == 1))
                                {
                                    //Console.WriteLine(" 4th if block - SetScore func....");
                                    score -= mediumDelta;
                                }
                            }
                            //same for fresh H2O
                            if (pheno[x, y] == 2)
                            {
                                freshCount++;
                                freshPercent = 1.0 - (seaPercent + landPercent);

                                if ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 &&
pheno[x, y - 1] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1]
== 1 && pheno[x, y + 1] != 1)
                                            || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1]
== 1 && pheno[x - 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] != 1)
                                            || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1]
== 1 && pheno[x + 1, y] == 1))
                                {
                                    //Just for debug puposes
                                    //Console.WriteLine(" 4th if block - SetScore func....");
                                    score -= mediumDelta;

                                }
                            }

                            // !!  CHANGE THIS INTO 2 SPECIFIC LOOPS FOR SEA AND FRESH - TO BE MORE SENSIBLE -
DONT MIX THEM !
                            //Look at each cell  If the cell is water and more or less surrounded by water,
score it up
                            //if (pheno[x, y] != 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x,
y + 1] != 1 && pheno[x, y - 1] == 1)
                            //              || (pheno[x - 1, y] != 1 && pheno[x + 1, y] != 1 && pheno[x, y - 1]
!= 1 && pheno[x, y + 1] == 1)
                            //              || (pheno[x, y - 1] != 1 && pheno[x + 1, y] != 1 && pheno[x, y + 1]
!= 1 && pheno[x - 1, y] == 1)
                            //              || (pheno[x - 1, y] != 1 && pheno[x, y - 1] != 1 && pheno[x, y + 1]
!= 1 && pheno[x + 1, y] == 1)))
                            //{
                            //Console.WriteLine(" 5th if block - SetScore func....");

                            //    score += 1;
                            //    local_score += 1;
                            //}

                            //Look at each cell If the cell is Sea and more or less surrounded by freshwater,
score it down
                            if (pheno[x, y] == 2 && ((pheno[x - 1, y] == 0 && pheno[x + 1, y] == 0 && pheno[x, y
+ 1] == 0 && pheno[x, y - 1] != 0)
                                            || (pheno[x - 1, y] == 0 && pheno[x + 1, y] == 0 && pheno[x, y - 1]
== 0 && pheno[x, y + 1] != 0)
                                            || (pheno[x, y - 1] == 0 && pheno[x + 1, y] == 0 && pheno[x, y + 1]
== 0 && pheno[x - 1, y] != 0)
                                            || (pheno[x - 1, y] == 0 && pheno[x, y - 1] == 0 && pheno[x, y + 1]
== 0 && pheno[x + 1, y] != 0)))
                            {
                                //Console.WriteLine(" 6th if block - SetScore func....");

                                score -= smallDelta;

                            }

                            //Look at each cell If the cell is freshwater and more or less surrounded by Sea,
score it down
                            if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 2 && pheno[x + 1, y] == 2 && pheno[x, y
+ 1] == 2 && pheno[x, y - 1] != 2)
                                            || (pheno[x - 1, y] == 2 && pheno[x + 1, y] == 2 && pheno[x, y - 1]
== 2 && pheno[x, y + 1] != 2)
                                            || (pheno[x, y - 1] == 2 && pheno[x + 1, y] == 2 && pheno[x, y + 1]
== 2 && pheno[x - 1, y] != 2)
                                            || (pheno[x - 1, y] == 2 && pheno[x, y - 1] == 2 && pheno[x, y + 1]
== 2 && pheno[x + 1, y] != 2)))
                            {
                                //Console.WriteLine(" 7th if block - SetScore func....");

                                score -= smallDelta;

                            }
```

```
                      }// end else loop - check for border 'pixels'

                    }// end getTerrain() check on (x,y) co-ords

                  }//end inner - y for loop

                }//end inner - x for loop

               //method 1  - cell is on land & more or less surrounded by H2O:

               return score;
            }
```

TEST format :

```
 public void scoringBordersHelperFunc(int i,int j)
        {
            //create a summation for these'conecting' pixels which can be used in the 
checks ahead...
            int phenoTerrainSum = 0; // pheno[x - 1, y] + pheno[x + 1, y] + pheno[x, y + 1] 
+ pheno[x, y - 1];
            if (i == Params.dimX - 1)
            {
                //cater for corner cases first - (i,j) = (Params.dimX - 1,Params.dimX - 1)
                if (j == Params.dimX - 1)
                {
                    if (pheno[i, j] == 1 && ((pheno[i - 1, j] == 1 && pheno[i, j - 1] != 1)
                         || (pheno[i, j - 1] == 1 && pheno[i - 1, j] != 1)
                         || (pheno[i - 1, j] == 1 && pheno[i, j - 1] == 1)))
                    {
                        //Console.WriteLine(" 3rd if block - land connected to mostlj other 
land - score up part of setScore func. ");
                        score += 5;
                    }

                    if (pheno[i, j] == 0 && (pheno[i - 1, j] == 1 && pheno[i, j - 1] != 1)
                                || (pheno[i, j - 1] == 1 && pheno[i - 1, j] != 1)
                                || (pheno[i - 1, j] == 1 && pheno[i, j - 1] == 1))
                    {
                        //Console.WriteLine(" 4th if block - SetScore func....");
                        score -= 3;
                    }
                }

                else if (j == 0)
                {
                    if (pheno[i, j] == 1 && ((pheno[i - 1, j] == 1 && pheno[i, j + 1] == 
1 )
                                || (pheno[i, j + 1] == 1 && pheno[i - 1, j] != 1)
                                || (pheno[i - 1, j] == 1 && pheno[i, j + 1] == 1)))
                                               {
                        //Console.WriteLine(" 3rd if block - land connected to mostlj other 
land - score up part of setScore func. ");
                        score += 5;
                    }
```

end TEST

```
               for (int x = 0; x < Params.dimX; x = x + every)

                  for (int y = 0; y < Params.dimY; y++){  // { pheno[x,y] = 0; } // initialise to
```

```
                    int z = 0;
                    if (x == Params.dimX- 1 || y == Params.dimY -1 || x == 0 || y == 0)
                        z = x;
                    // TODO : put in equivalents to below's logic
here !               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!

else{
             //create a summation for these'conecting' pixels which can be used in  the checks ahead...
                int phenoTerrainSum = pheno[x - 1, y] + pheno[x + 1, y] + pheno[x, y + 1] + pheno[x, y - 1];
                // Just focus on the simpler  horizontal and vertical connecting
 /              /current is on land and one neighbour is water - score up
        if (pheno[x, y] == 1 && (pheno[x - 1, y] != 1 | pheno[x + 1, y] != 1 | pheno[x, y + 1] != 1 | pheno[x, y
- 1] != 1)){
                            score += 1;
                            score += 1;}

                        // land surrounded by water - score down
                        // these sums indicted > 6 or < 2 would correspond to a piece of land mostly (75%)
surrounded by sea or fresh water.
                  if (pheno[x, y] == 1 && (phenoTerrainSum > 6 | phenoTerrainSum < 2))
                        {
                            score -= 1;
                            score -= 1;
                        }

                        //land connected to mostly other land (ay at least 75 % of horizonal & vert. cells) -
score up
           if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 &&
pheno[x, y - 1] != 1)
                                        | (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1] == 1
&& pheno[x, y + 1] != 1)
                                        | (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1
&& pheno[x - 1, y] != 1)
                                        | (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1
&& pheno[x + 1, y] != 1)))
                        {
                            score += 1;
                            score += 1;
                        }


                        // Look at each cell    If the cell is water  and more or less surrounded by land,
score it down
                        if (pheno[x, y] != 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1]
== 1 && pheno[x, y - 1] != 1)
                                        | (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1] == 1
&& pheno[x, y + 1] != 1)
                                        | (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1
&& pheno[x - 1, y] != 1)
                                        | (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1
&& pheno[x + 1, y] != 1)))
                        {
                            score -= 1;
                            score -= 1;
                        }
```

As you can see I used the this pointer to reference the phenotype in Question being scored – to make sure i am not writing to just a copy of said variable.
In Addition I took care of the boundaries of the bit map as they are a special cases.

Q .2 Mutation routine :

For the mutation routine I used 5 different mutation types. For each (most) of these i madde a small change to some of a Gene's data members – x,y,repeatX/Y etc. I did this for 3 random indices (Genes) along a genotype, and the delta – which I either added or subrtracted (randomly each one 1/2 of the time) by a value of 5.

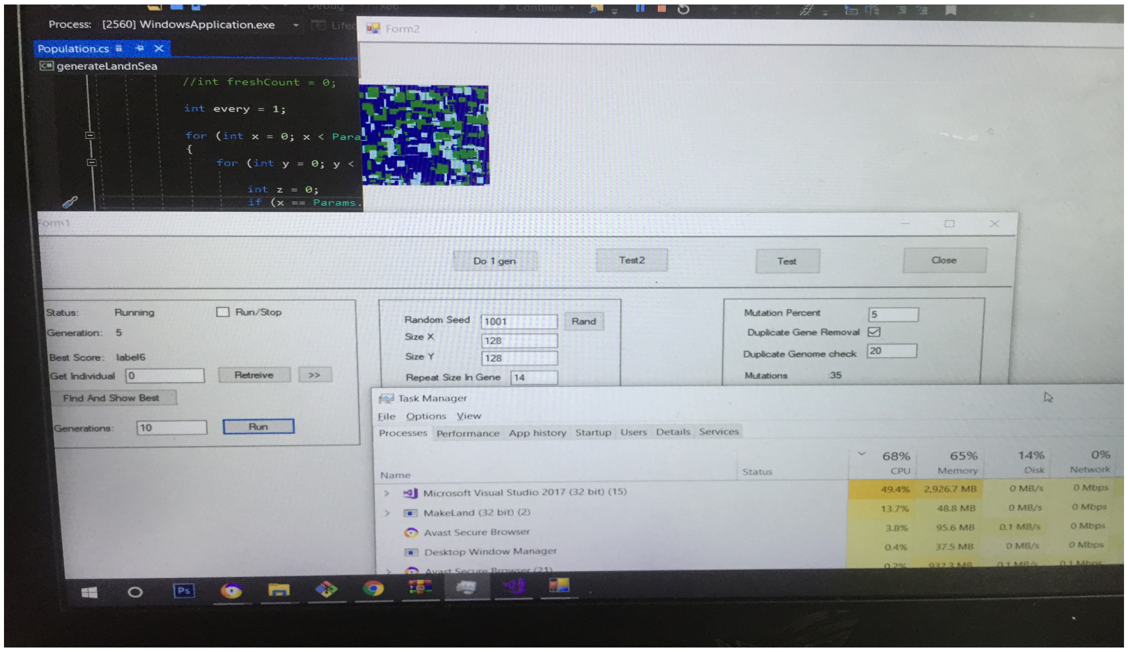
- add or subtract a small delta from 3 random indices' x value along the genotype
- add or subtract a small delta from 3 random indices' y value along the genotype
- add or subtract a small delta from 3 random indices' repeatX value along the genotype
- add or subtract a small delta from 3 random indices' repeatX value along the genotype
- Also swapped the gene with another existing one in the population.
- Finally, I replaced the current Genotype with a randomly generated one. IN case 5.

I created a random number "mutateToggle" to evenly and randomly choose between these 5 strategies each time. Then I used a simple Switch flow control to do this – which suited it quite well.

Code

```
public void mutate(Genotype g, Random r)
        {

            // generate a random no. in [1,6] to choose which mutation strategy to employ each time:
            int toggleMutate = (int) (r.NextDouble() * 6);

        //debugging print ...
            //String togStr = toggleMutate.ToString();
            //Console.WriteLine("toggleMutate = " + togStr);
            switch (toggleMutate) {

                case 1:

                    //mutate a couple (3 - fixed for now) random indices's repeatX  along the genotype
                    for (int k=0; k < 3; k++) {

                    int mutate_index = (int)r.NextDouble() * Params.genotypeSize;

                    if (G.rnd.NextDouble() < 0.5)
                        g.genes[mutate_index].repeatX += (int)r.NextDouble() * 5; //int(1 / r);
                    else
                        g.genes[mutate_index].repeatX -= (int)r.NextDouble() * 5;

                    }
                    break;

                case 2:

                    //mutate a couple (3 - fixed for now) random indices' x value along the genotype
                    for (int k = 0; k < 3; k++)
                    {

                        int mutate_index = (int)r.NextDouble() * Params.genotypeSize;

                        if (G.rnd.NextDouble() < 0.5)
                            g.genes[mutate_index].x -= (int)r.NextDouble() * 5; //int(1 / r);
                        else
                            g.genes[mutate_index].x += (int)r.NextDouble() * 5;


                    }
                    break;

                case 3:

                    //mutate a couple (3 - fixed for now) random indices's repeatY value along the genotype
                    for (int k = 0; k < 3; k++)
                    {
                        int mutate_index = (int)r.NextDouble() * Params.genotypeSize;
                        if (G.rnd.NextDouble() < 0.5)
                            g.genes[mutate_index].repeatY += (int)r.NextDouble() * 5; //int(1 / r);
                        else
                            g.genes[mutate_index].repeatY -= (int)r.NextDouble() * 5;

                    }
                    break;

                case 4:

                    //mutate a couple (3 - fixed for now) random indices' y value along the genotype
                    for (int k = 0; k < 3; k++)
                    {

                        int mutate_index = (int)r.NextDouble() * Params.genotypeSize;

                        if (G.rnd.NextDouble() < 0.5)
                            g.genes[mutate_index].y += (int)r.NextDouble() * 5; //int(1 / r);
                        else
                            g.genes[mutate_index].y -= (int)r.NextDouble() * 5;
                    }
                    break;

                case 5:

                    //replace gene with a new randomly generated one:
                    g = new Genotype(r);
                    break;

                case 6:

                    //grab another genotype and swap with g :

                    Genotype swap_genotype = getPhenotype((int)r.NextDouble() * numInPop).genotype;
                    Genotype temp = new Genotype(r);

                    //swap them using the temp as intermediary:
                    temp = g;
                    g = swap_genotype;
                    swap_genotype = temp;
                    break;

                default:
                    break;
            }
            G.mutationCount++;
        }
```

Experiments using these strategies :



Process: [2560] WindowsApplication.exe

Population.cs — generateLandnSea

```
//int freshCount = 0;

int every = 1;

for (int x = 0; x < Para
{
    for (int y = 0; y <

        int z = 0;
        if (x == Params.
```

Form2

Form1

Status:	Running		☐ Run/Stop
Generation:	5		
Best Score:	label6		
Get Individual	0	Retreive	>>
Generations:	10	Run	

Random Seed	1001	Rand
Size X	128	
Size Y	128	
Repeat Size In Gene	14	

Mutation Percent	5
Duplicate Gene Removal	☑
Duplicate Genome check	20
Mutations	:35

Task Manager

Name	Status	68% CPU	65% Memory	14% Disk	0% Network
Microsoft Visual Studio 2017 (32 bit) (15)		49.4%	2,926.7 MB	0 MB/s	0 Mbps
MakeLand (32 bit) (2)		13.7%	48.8 MB	0 MB/s	0 Mbps
Avast Secure Browser		3.8%	95.6 MB	0.1 MB/s	0 Mbps
Desktop Window Manager		0.4%	37.5 MB	0 MB/s	0 Mbps

Q 3 . I did not extend or change the Genotype structure as provided in the initial code.

Q4. i did not add any new data structures per se, but I did add several variables which made some of the code more succintly and easier to read.

***************************** Possibly remove ??! ************************************

I just ran 10 iterations and reduced down the population size to 200 & the no. Of genes/genotype to 300 as my system's memory seemed to be struggling. Anyway I overcame this memory issue and tried again but each time it was the case that the bitmap did not evolve from a very early iteration if at all beyond just one. You can see this in in the screenshot above.

Looking into the code for the reason for this I noticed (!!) that the actually optimization function itself – 'findBest()' was not even being called anywhere !! Normally in a GA this is called in every or some iterations or there will be no evolution. Parents are usually selected explicitly based on the findBest() optimization and then crossed -over to create good offspring. But in this code the parents are both randomly. Clearly though this still works as with the scoring alg. The findBest() ensures that this 'best' genotype improves over the course of generations of populations.

********************************** Possibly remove ??! **********************************

Reflection :

What I found Interesting or difficult / Challenging ?

It was challenging to figure out why the bitmap was not evolving over many generation initially. I had to ensure i had several different mutation and Scoring strategies so that it was powerful enough to capture the complexity required. I also had to make sure that they were well tested /debugged before I was confident this was the cause of the static image. It turned out that some debugging and inclusion of several strategies in both routines then allowed the image to evolve to an acceptably realistic final state. This was quite interesting to see how much complexity you need to capture in code (esp. the scoring function) to really leverage the power of a GA !

Initially when running this my laptop caused the running program to crash repeatedly, it seemed it could not handle the memory usage. So I had to ensure all other applications were not running and even some unneeded background ones via task-manager.

Q .What effect did changes in setScore() and mutate() strategies have on the algorithm's output ?

For the scoring function, I noticed that the addition of the % land and % (sea,fresh) water relative to the entire map really improved how sensible the bitmap became after 100 or so iterations.
These were constraints set in Global.cs that I applied. I did this by re-calculating these %s as running values within each generation.

On top of this I saw the same effect when I was more strict on how I defined land for being connected to more land. This of course incremented the score value by a lot (7 units as opposed to other checks that only had a delta change of +/- 2 or so.) In fact the combination of these two constraints in the scoring really boosted how the bit image evolved.
Before this I was pretty much hitting an 'optimal' poor quality map that stayed static after only a small number n.o generations (< 10).

The improvement really showed with the more strict scoring system after 250 or so iterations and began to look like the best one shown in the assignment document.

Also I noticed, as mentioned in the assignment doc. That when different scoring strategies scored varying and intuitive relative values, better results followed. By this I mean I assigned the larger deltas to add or subtract in the strategies which I felt were more 'important'

I say strict when referring to the land surrounded by land check, rather than just comparing (x,y) with neighbouring location along the horizonal and vertical directions, i.e (x-1,y),(x+1,y),(x,y-1),(x,y+1). I also added in the neighbouring pixels which connected to x,y at the diagonals , i.e (x-1,y+1),(x-1,y-1) ,(x+1,y-1) ,(x+1,y+1).

Attributes specific to the 256x256 resolution ...

When I ran this with the same set up as above and as many generations it did not converge or end up looking anywhere as good as the 128x128 version. Land did not take over the screen so much and did not 'clump' together in any noticeably obvious amount. The problem at this resolution was much more difficult I found. It was also more intensive on the PC's CPUs and memory usage.

*********** remove ? ***************

The attempts I made using the same scoring and mutation routines as with the 128x128 version were not as fruitful . They did not evolve into anything resembling the final state of the latter.

*********** remove ? ***************

I played with generation number, population and Gene no.'s too . I thought increasing the no.of pixels to check in the 'land- near land' scoring strategy coupled with raising the population , gene no. In genotype and even generation no.s would work. But this did not seem to be working for a while , until I found a 'bug' when I was scoring the land and water %' s (`percentLand in Global.cs`). I had foolishly increased the score for when these %s agreed with the constraints and negated the inverse case – which would lead to having a zero net effect. So I chose to just negate the score accordingly. Additionally I added a lower bound on the land % as well , as I felt a realistic state should have ~ 40 % or so land in it.

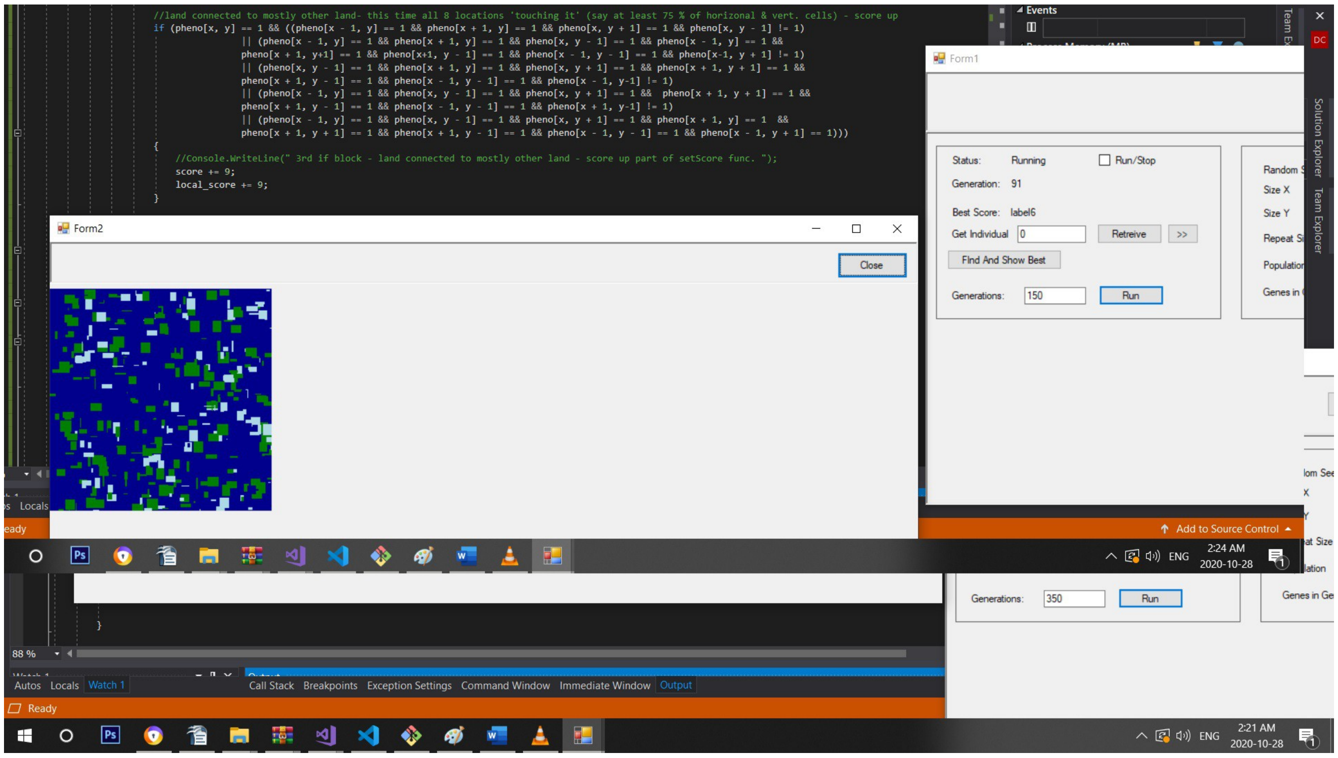
This can be seen in the screenshot with the cumbersome name – 'AddPixelsToLandClumpingScore_onlyHaveScoredownFor%s_256Res_mutate %_5__genNum_400_pop500_Genes75.jpg'

Q 5, 6 , 7 ,

<u>Results of trials on both resolution schemes and variations in other paramters :</u>

The following are some results ,screenshot of the results I mention above :

256x256 res. Experiment screenshots @ 150 generations, before the improvements :

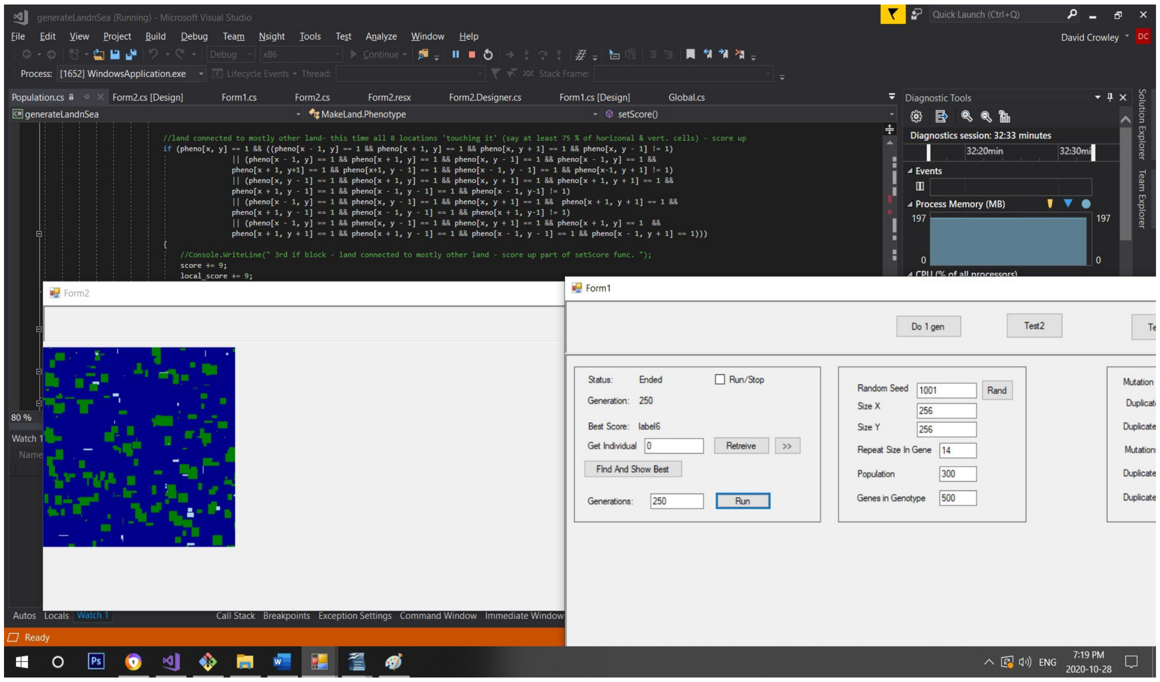


```
//land connected to mostly other land- this time all 8 locations 'touching it' (say at least 75 % of horizonal & vert. cells) - score up
if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 && pheno[x, y - 1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x - 1, y] == 1 &&
                pheno[x + 1, y+1] == 1 && pheno[x+1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x-1, y + 1] != 1)
                || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 && pheno[x + 1, y + 1] == 1 &&
                pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x - 1, y-1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1 &&  pheno[x + 1, y + 1] == 1 &&
                pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x + 1, y-1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1 && pheno[x + 1, y] == 1  &&
                pheno[x + 1, y + 1] == 1 && pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x - 1, y + 1] == 1)))
{
    //Console.WriteLine(" 3rd if block - land connected to mostly other land - score up part of setScore func. ");
    score += 9;
    local_score += 9;
}
```

Status: Running
Generation: 91
Best Score: label6
Get Individual: 0
Generations: 150

Generations: 350

for 350 generations :



```
//land connected to mostly other land- this time all 8 locations 'touching it' (say at least 75 % of horizonal & vert. cells) - score up
if (pheno[x, y] == 1 && ((pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 && pheno[x, y - 1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x + 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x - 1, y] == 1 &&
                pheno[x + 1, y+1] == 1 && pheno[x+1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x-1, y + 1] != 1)
                || (pheno[x, y - 1] == 1 && pheno[x + 1, y] == 1 && pheno[x, y + 1] == 1 && pheno[x + 1, y + 1] == 1 &&
                pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x - 1, y-1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1 &&  pheno[x + 1, y + 1] == 1 &&
                pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x + 1, y-1] != 1)
                || (pheno[x - 1, y] == 1 && pheno[x, y - 1] == 1 && pheno[x, y + 1] == 1 && pheno[x + 1, y] == 1  &&
                pheno[x + 1, y + 1] == 1 && pheno[x + 1, y - 1] == 1 && pheno[x - 1, y - 1] == 1 && pheno[x - 1, y + 1] == 1)))
{
    //Console.WriteLine(" 3rd if block - land connected to mostly other land - score up part of setScore func. ");
    score += 9;
    local_score += 9;
```

Process: [1652] WindowsApplication.exe
Diagnostics session: 32:33 minutes

Status: Ended
Generation: 250
Best Score: label6
Get Individual: 0
Generations: 250

Random Seed: 1001
Size X: 256
Size Y: 256
Repeat Size In Gene: 14
Population: 300
Genes in Genotype: 500