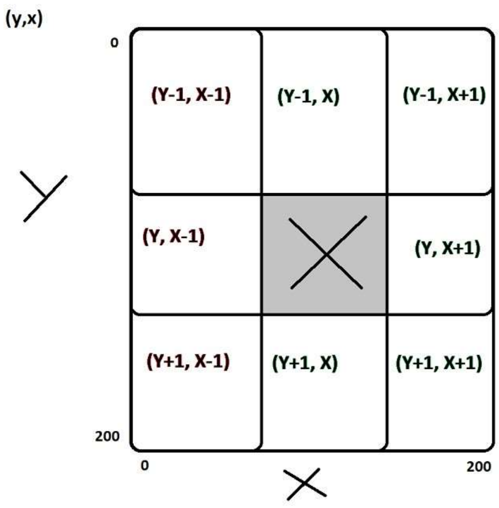
C Programming

Peak of Mountain

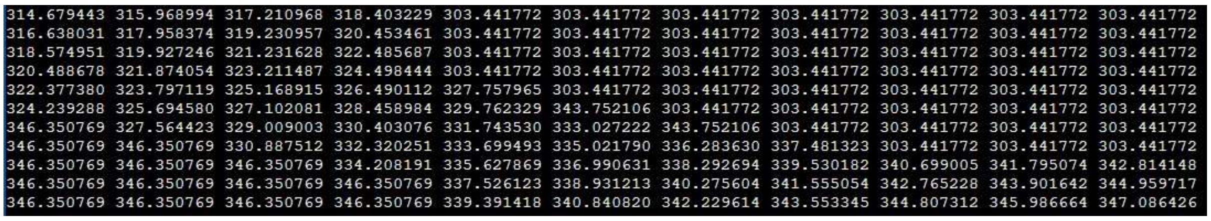
Cian Herlihy – R00205604

X,Y Diagram



I used this as a reference of what the x and y values should do when travelling in a certain direction.

False Peak



```
314.679443 315.968994 317.210968 318.403229 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772
316.638031 317.958374 319.230957 320.453461 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772
318.574951 319.927246 321.231628 322.485687 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772
320.488678 321.874054 323.211487 324.498444 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772
322.377380 323.797119 325.168915 326.490112 327.757965 303.441772 303.441772 303.441772 303.441772 303.441772 303.441772
324.239288 325.694580 327.102081 328.458984 329.762329 343.752106 303.441772 303.441772 303.441772 303.441772 303.441772
346.350769 327.564423 329.009003 330.403076 331.743530 333.027222 343.752106 303.441772 303.441772 303.441772 303.441772
346.350769 346.350769 330.887512 332.320251 333.699493 335.021790 336.283630 337.481323 303.441772 303.441772 303.441772
346.350769 346.350769 346.350769 334.208191 335.627869 336.990631 338.292694 339.530182 340.699005 341.795074 342.814148
346.350769 346.350769 346.350769 346.350769 337.526123 338.931213 340.275604 341.555054 342.765228 343.901642 344.959717
346.350769 346.350769 346.350769 346.350769 339.391418 340.840820 342.229614 343.553345 344.807312 345.986664 347.086426
```

I found a false peak with 343.752106 with no higher values adjacent to it. This should not have been possible and forced me to change my program completely to check areas nearby for higher values rather than touching units. My first iteration of the program I created did not need a check surrounding areas until I met this false peak.

gradient_sol.c

#include "gradient.h"

path_point find_highest_point(){

```
path_point my_point;
float my_view[VIEW_SIZE][VIEW_SIZE];
int y, x, xValue, yValue, xCounter, yCounter;
int option = 0;
int lastOption = 0;

//Start at (70,70) in hopes of finding mountain quikcer
my_point.x = 70;
my_point.y = 70;
generate_view(my_view, my_point.y, my_point.x);

// Reset to 0 before loop starts
xValue = my_point.x = 5;
yValue = my_point.y = 5;
xCounter = 5;
yCounter = 5;
while(1)
{
float value = my_view[yValue][xValue]; // Current Value
float t_val = my_view[yValue-1][xValue]; // Top Value
float tr_val = my_view[yValue-1][xValue+1]; // Top Right Value
float r_val = my_view[yValue][xValue+1]; // Right Value
float br_val = my_view[yValue+1][xValue+1]; // Bottom Right Value
float b_val = my_view[yValue+1][xValue]; // Bottom Value
float bl_val = my_view[yValue+1][xValue-1]; // Bottom Left Value
float l_val = my_view[yValue][xValue-1]; // Left Value
float tl_val = my_view[yValue-1][xValue-1]; // Top Left Value

//Array full of values
float arrayVals[] = {t_val, tr_val, r_val, br_val, b_val, bl_val, l_val, tl_val};
```

```
int highestVal = 0;

for (int i=0; i<5; i++)

{

    if (highestVal < arrayVals[i])

    {

        highestVal = arrayVals[i];

    }

}




/*

Checks if it is at the edge of the view and then proceeds to check certain directions.

Sends option number to switch case to iterate the x or y value depending on what direction it went.

*/

if (xValue == 0 || xValue == 10 || yValue == 0 || yValue == 10)

{

    generate_view(my_view, yCounter, xCounter);

    xValue = my_point.x = 5;

    yValue = my_point.y = 5;

}

else if (value < br_val && br_val < 999) // Bottom Right Value

{

    option = 0;

    lastOption = 0;

}

else if (value < b_val && b_val < 999) // Bottom Value

{

    option = 1;

    lastOption = 1;
}
else if (value < r_val && r_val < 999) // Right Value
{
   option = 2;
   lastOption = 2;
}
else if (value < tl_val && tl_val < 999) // Top Left Value
{
   option = 3;
   lastOption = 3;
}
else if (value < l_val && l_val < 999) // Left Value
{
   option = 4;
   lastOption = 4;
}
else if (value < t_val && t_val < 999) // Top Value
{
   option = 5;
   lastOption = 5;
}
else if (value < bl_val && bl_val < 999) // Left Value
{
   option = 6;
   lastOption = 6;
}
else if (value < tr_val && tr_val < 999) // Top Value
{
   option = 7;
   lastOption = 7;
}
```

```
  else if (value == t_val && value == r_val && value == b_val && value == l_val)
  {
    option = 9;
  }
  else if (value == b_val && lastOption != 5)
  {
    option = 1;
    lastOption = 1;
  }
  else if (value == l_val && lastOption != 2)
  {
    option = 4;
    lastOption = 4;
  }
  else if (value == t_val && lastOption != 1)
  {
    option = 5;
    lastOption = 5;
  }
  else if (value == r_val && lastOption != 4)
  {
    option = 2;
    lastOption = 2;
  }
  else if (value >= highestVal) // Has found false peaks in the past so I have it do option 9 if it is not
the true peak. Example shown in document
  {
    if(declare_peak(xCounter, yCounter) == 1)
    {
      my_point.x = xCounter;
      my_point.y = yCounter;
```

```
            return my_point;
        }
        else
        {
            option = 9;
        }
    }
    else
    {
        option = 9;
    }

    switch (option)
        {
            case 0: // Bottom Right Value
                xValue++;
                yValue++;
                xCounter++;
                yCounter++;
                break;
            case 1: // Bottom Value
                yValue++;
                yCounter++;
                break;
            case 2: // Right Value
                xValue++;
                xCounter++;
                break;
            case 3: // Top Left Value
                xValue--;
                yValue--;
                xCounter--;
                yCounter--;
                break;
            case 4: // Left Value
                xValue--;
                xCounter--;
                break;
            case 5: // Top Value
                yValue--;
                yCounter--;
                break;
            case 6: // Bottom Left Value
                yValue++;
                yCounter++;
                xValue--;
                xCounter--;
                break;
            case 7: // Top Right Value
                yValue--;
                yCounter--;
                xValue++;
                xCounter++;
                break;
            case 9: // Checks nearby areas when in the middle of plateau


                /*
                    generate view to the Top Right
                */
                generate_view(my_view, yCounter-10, xCounter+10);
                int highTRx=0, highRx=0, highBRx=0, highBx=0, highBLx=0, highLx=0, highTLx=0, highTx=0,
    highCx=0;
```

```
        int highTRy=0, highRy=0, highBRy=0, highBy=0, highBLy=0, highLy=0, highTLy=0, highTy=0,
highCy=0;
        float highTRval=0, highRval=0, highBRval=0, highBval=0, highBLval=0, highLval=0,
highTLval=0, highTval=0, highCval=0;



        for (y=0; y<VIEW_SIZE; y++)
        {
          for (x=0; x<VIEW_SIZE; x++)
          {
            if (highTRval < my_view[y][x] && my_view[y][x] < 999)
            {
              highTRx = x;
              highTRy = y;
              highTRval = my_view[y][x];
            }
          }
        }

        /*
          generate view to the Right
        */
        generate_view(my_view, yCounter, xCounter+10);

        for (y=0; y<VIEW_SIZE; y++)
        {
          for (x=0; x<VIEW_SIZE; x++)
          {
            if (highRval < my_view[y][x] && my_view[y][x] < 999)
            {
              highRx = x;
              highRy = y;
            highRval = my_view[y][x];
        }
    }
}


/*
    generate view to the Bottom Right
*/
generate_view(my_view, yCounter+10, xCounter+10);

for (y=0; y<VIEW_SIZE; y++)
{
    for (x=0; x<VIEW_SIZE; x++)
    {
        if (highBRval < my_view[y][x] && my_view[y][x] < 999)
        {
            highBRx = x;
            highBRy = y;
            highBRval = my_view[y][x];
        }
    }
}



/*
    generate view to the Bottom
*/
generate_view(my_view, yCounter+10, xCounter);

for (y=0; y<VIEW_SIZE; y++)
{
    for (x=0; x<VIEW_SIZE; x++)
    {
        if (highBval < my_view[y][x] && my_view[y][x] < 999)
        {
            highBx = x;
            highBy = y;
            highBval = my_view[y][x];
        }
    }
}



/*
    generate view to the Bottom Left
*/
generate_view(my_view, yCounter+10, xCounter-10);



for (y=0; y<VIEW_SIZE; y++)
{
    for (x=0; x<VIEW_SIZE; x++)
    {
        if (highBLval < my_view[y][x] && my_view[y][x] < 999)
        {
            highBLx = x;
            highBLy = y;
            highBLval = my_view[y][x];
        }
    }
}
```

```
/*
   generate view to the Left
*/
generate_view(my_view, yCounter, xCounter-10);


for (y=0; y<VIEW_SIZE; y++)
{
   for (x=0; x<VIEW_SIZE; x++)
   {
      if (highLval < my_view[y][x] && my_view[y][x] < 999)
      {
         highLx = x;
         highLy = y;
         highLval = my_view[y][x];
      }
   }
}


/*
   generate view to the Top Left
*/
generate_view(my_view, yCounter-10, xCounter-10);


for (y=0; y<VIEW_SIZE; y++)
{
   for (x=0; x<VIEW_SIZE; x++)
   {
        if (highTLval < my_view[y][x] && my_view[y][x] < 999)
        {
            highTLx = x;
            highTLy = y;
            highTLval = my_view[y][x];
        }
    }
}

/*
    generate view to the Top
*/
generate_view(my_view, yCounter-10, xCounter-10);


for (y=0; y<VIEW_SIZE; y++)
{
    for (x=0; x<VIEW_SIZE; x++)
    {
        if (highTval < my_view[y][x] && my_view[y][x] < 999)
        {
            highTx = x;
            highTy = y;
            highTval = my_view[y][x];
        }
    }
}


/*
    generate view to the Centre
```

```
*/
generate_view(my_view, yCounter, xCounter);


for (y=0; y<VIEW_SIZE; y++)
{
   for (x=0; x<VIEW_SIZE; x++)
   {
      if (highCval < my_view[y][x] && my_view[y][x] < 999)
      {
         highCx = x;
         highCy = y;
         highCval = my_view[y][x];
      }
   }
}


// Gathers all highest values found and puts into an array
float platCheckVals[] = {highTRval, highRval, highBRval, highBval, highBLval, highLval,
highTLval, highTval, highCval};
float max = 0;
int arrayPos = 0;
for (int i=0; i<9; i++) // Iterates through array and picks highest option and travels to that
altitude to carry on search
{
   if (max < platCheckVals[i] && platCheckVals[i] > 1)
   {
      max = platCheckVals[i];
      arrayPos = i;
   }
}


if (max < 1)
```

```
{
   arrayPos = 9;
}



// Switch case to handle x and y values being plus/minus respectively
switch (arrayPos)
{
   case 0: //  Top Right
      yCounter = yCounter - (15 - highTRy);
      xCounter = xCounter + 5 + highTRx;
      break;
   case 1: // Right
      yCounter = yCounter - 5 + highRy;
      xCounter = xCounter + 5 + highRx;
      break;
   case 2: // Bottom Right
      yCounter = yCounter + 5 + highBRy;
      xCounter = xCounter + 5 + highBRx;
      break;
   case 3: // Bottom
      yCounter = yCounter + 5 + highBy;
      xCounter = xCounter - 5 + highBx;
      break;
   case 4: // Bottom Left
      yCounter = yCounter + 5 + highBLy;
      xCounter = xCounter - (15 - highBLx);
      break;
   case 5: //  Left
      yCounter = yCounter - 5 + highLy;
      xCounter = xCounter - (15 - highLx);
```

```
                break;
            case 6: // Top Left
                yCounter = yCounter - (15 - highTLy);
                xCounter = xCounter - (15 - highTLx);
                break;
            case 7: // Top
                yCounter = yCounter - (15 - highTy);
                xCounter = xCounter - 5 + highTx;
                break;
            case 8: // Centre
                yCounter = yCounter - 5 + highTy;
                xCounter = xCounter - 5 + highTx;
                break;
            case 9: // Randomise if no higher value found
                xValue = my_point.x = random()%100;
                yValue = my_point.y = random()%100;
                xCounter = xValue;
                yCounter = yValue;
                generate_view(my_view, my_point.y, my_point.x);
                break;
        }

        generate_view(my_view, yCounter, xCounter);
        xValue = my_point.x = 5;
        yValue = my_point.y = 5;
        break;
    }
  }

  return my_point;
```

```
}
```