AP Computer Science
Mr Hanley

The Hood

Recursion

Ver: 3.1

Last Updated:11/3/2024 9:22 AM

Assignment 15:Recursion

Binary **Ones Comp** **Twos Comp**

1. **Write the following 2 methods recursively**

   a. **Find the greatest common factor**

   A recursive approach could use a "candidate" value.
   This means that each call to the recursive method takes 3 parameters, num1, num2 and a candidate.
   If the candidate divides evenly, return it…otherwise recurse down
   DO NOT use doubles or negatives for this assign.
   DO NOT USE EUCLID'S ALGORITHM,
   DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM, DO NOT USE EUCLID'S ALGORITHM

   b. **Raise an integer to a power**

   Allow the user to type in a base and exponent. Exponents may be positive, negative or zero.
   Exponents MAY NOT have decimal points with numbers after
   Examples.
   Please enter in a base: 3
   Please enter in an exponent: 4
   Your answer is **81**

   Please enter in a base: 5
   Please enter in an exponent: -2
   Your answer is **.04**
   INCLUDE RAISING TO A NEGATIVE POWER, INCLUDE

RAISING TO A NEGATIVE POWER, INCLUDE RAISING TO A NEGATIVE POWER, INCLUDE RAISING TO A NEGATIVE POWER, INCLUDE RAISING TO A NEGATIVE POWER

2. Complete the AreaFill Skeleton
    a. Complete the areaFill ()method and then fill with & so that we can see where the initial mouse click was and the filled area
    b. Finish the GraphMap method isInBounds

    The recursive area fill problem comes from the Litvin's book C++ for You++. Page 341-343 (isbn 0-9654853-9-0) 1998
    Suppose a text file is stored that represents a computer image. We will allow characters to represent the pixels.
    An image may have a number of arbitrarily shaped blobs, contours, isolated pixel, etc. Which each white pixel in an image we can associate a certain white area (space) called the connectivity component of that pixel. This is defined as the set of all pixels that can be connected to the given pixel with a continuous chain of white pixels.

    Metaphorically, we can think of all black pixels and contours as "walls" between white "containers". If we pour in black paint at a given point, then the container filled with black paint is the connectivity component of that point – a concept familiar to all users of paint programs. The areaFill method takes a specified white pixel in an image and fills the connectivity component of that pixel with black.

For example, here is an example of an image

```
...**.....
..*..***..
..*....***
.*........
.*......**
.***...*..
...****...
```

Assuming someone clicks at row 3(0 based, the 4th row) and col 4(0 based the 5th column), we are going to mark this bad boy with a @ to connote a FIRST CLICK

```
...**.....
..*..***..
..*....***
.*..@.....
.*......**
.***...*..
...****...
```

Now, let’s start “blob filling” as the great Phil Sun calls it.

We can go north, south, east and west in any order we choose.

Assuming we go North first, let’s mark that bad boy with a fill character.

```
...**.....
..*..***..
..*.&..***
.*..@.....
.*......**
.***...*..
...****...
```

And again, since we filled a space we can go north again

```
...**.....
..*.&***..
..*.&..***
.*..@.....
.*......**
.***...*..
...****...
```

Let's try to go north again …oh, that is a WALL. We must simply return and not go that direction any more. Let's go WEST next (arbitrary, doesn't matter which compass direction we use…..!!!!)

```
...**.....
..*&&***..
..*.&..***
.*..@.....
.*......**
.***...*..
...****...
```

When its all said and done, it looks like dis

```
...**.....
..*&&***..
..*&&&&***
.*&&@&&&&&
.*&&&&&&**
.***&&&*..
...****...
```

The Litvins had a C++ class that represented the 2d image.
Their class was called IMAGE
We will be using some java classes on this project.

GraphMap
The GraphMap class represents an image. It has a constructor written by Josh Komoroske(2009 shen grad) that reads in the colors from a bitmap and decides whether or not the color connotes a wall.
It is based on a 2d array of characters called map

You must write the constructor to read in a text file and set up the character array to correspond to the walls and spaces. (Remember . is a space while * is a wall)

AreaFillApp
Here is Josh's app, it by default loads up one of the bitmaps and then calls resizeToFit to make sure there are sufficient JButtons. Each JButton represents a pixel. We're gonna color these bad boys different colors so we can see the image fillin' while we are chillin'! ;-)(Uh, oh, I had too much coffee again!!)

```
public class AreaFillApp {

  public static void main(String[] args) {
    AreaFillFrame a = new AreaFillFrame("image2.bmp",12, 12, 0,
0);
    a.setTitle("JDK Area Fill");
    a.resizeToFit( -1, -1, 6, 27);
    a.setVisible(true);
  }

}
```

AreaFillFrame

Method breakdown:

private void copyMapToScreen()

copies all of the symbols from the gm global GraphMap to the JButton array on the frame. This is used to refresh after we click

Play with the colors if you like!!!!!

```
    private void copyMapToScreen() {
        //Colors are in RGB format 0..255 for each
        for (int y = 0; y < gm.getHeight(); y++) {
            for (int x = 0; x < gm.getWidth(); x++) {
                switch (gm.getPixel(x, y)) {

                    case '*':  //Wall or black
                        buttons[x][y].setBackground(Color.BLACK);
                        break;

                    case '@'://First Click of each fill
                        buttons[x][y].setBackground(new
Color(0x99ff00));
                        break;

                    case '&': //Subsequent pixels after first click
                        buttons[x][y].setBackground(new
Color(0xccccc00));
                        break;

                    case '.'://Space or white
                        buttons[x][y].setBackground(Color.RED);
                        break;              } } } }
```

```
private void areaFill(int x, int y)
TO DO: written by you, this will make sure x and y are on the
image and then check to see if the pixel is a space, if so

public AreaFillFrame(String file,int w, int h, int x, int y)
written by Komo, this sucker takes in a filename and does some
mojo to set up the correct width of the window

public void actionPerformed(ActionEvent e) {
    Point p = getButtonPos( (JButton) e.getSource()); //this tells us
       int x = p.x, y = p.y;

       if (gm.getPixel(x, y) == '.') {
         areaFill(x, y);  //starts the areaFill method, you will need to
recurse inside the method itself
         gm.setPixel(x, y, '@');  //when done, over-write the click
spot with a
           //@ symbol so we know where we clicked!!!
       }

        this.copyMapToScreen();  //update the buttons
}
private Point getButtonPos(JButton b)
this dudette looks at the array of figures out the x and y of the
jButton of wherever you clicked

setPixelSize(int w, int h)
cool stuff by Josh!
public void setPixelMargins(int x, int y)
cool stuff by Josh!
public Point getGridSize()
```

cool stuff by Josh!

public void resizeToFit(int x, int y, int bx, int by
cool stuff by Josh!

public void setFullscreen(boolean m)
cool stuff by Josh!

private void jbInit() throws Exception
This is what our old IDE JBuilder called its auto generated window set up logic. Adds buttons to the window and does set up stuff.
Thanks to the creators of JBuilder, a Borland product that we used here for years. It was a darn good product!!!

public void clearFrame(){

//This gets rid of all JButtons on the frame
getContentPane().removeAll();
}
public void keyTyped(KeyEvent ke) {
if (ke.getKeyChar() == KeyEvent.VK_SPACE) {
String filename;
JFileChooser fc = new JFileChooser();
int rc = fc.showDialog(null, "Select Data File");

if (rc == JFileChooser.APPROVE_OPTION) {
//The user chose a file, let's see if we can open it
File file = fc.getSelectedFile();
filename = file.getAbsolutePath();

```
TO DO: Uncomment the next line once you have a constructor
defined
                //gm = new GraphMap(filename, true);
                clearFrame();
                try {
                    //resizeToFit(-1, -1, 6, 27);
                    jbInit();
                } catch (Exception ex) {

Logger.getLogger(AreaFillFrame.class.getName()).log(Level.SEVE
RE, null, ex);
                }

                resizeToFit(-1, -1, 4, 10);

                copyMapToScreen();
            }
        }

    }
```

Checklist for things to do on AreaFill

☐Finish the GraphMap method isinBounds

☐Finish the AreaFillFrame method areaFill

☐Create a two arg constructor for GraphMap

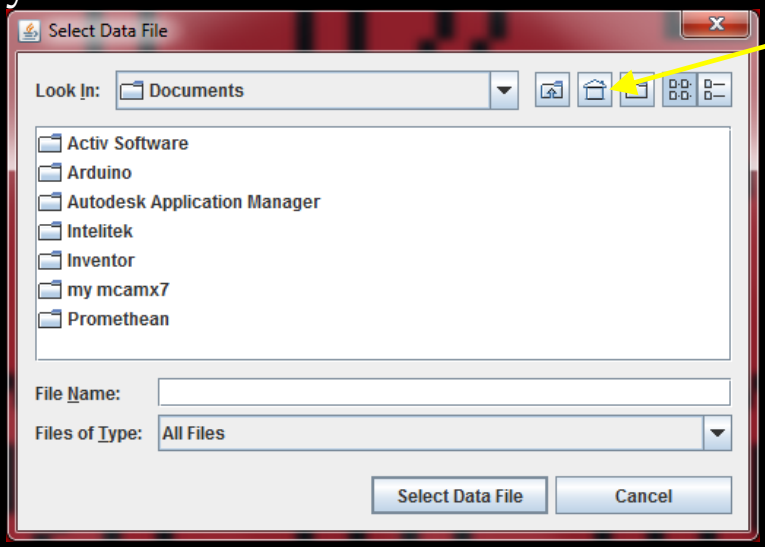
You will be passing in a file name and a boolean

This is done for you in the keyTyped method

```
 //gm = new GraphMap(filename, true);
```

(You may ignore the Boolean as I just put it in so I could have a different constructor that passed in a filename(overloaded))

The file format is
50 (width or columns -> second index into array)
50(height or rows ->first index into array)
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
FILES ARE NOT ALWAYS 50 X 50, COULD BE DIFFERENT SIZES,
.....****...***
...***... etc
If its all good, you should be able to hit the space bar and then load up the pict1.txt file provided
HINT: Just copy the pict1.txt file to your desktop, it is WAY faster than navigating through all the folders to find your file



3. Complete the codingbat recursion problems
   a. Recursion 1: factorial, bunnyEars, bunnyEars2, triangle, count7,count8, powerN, countX, changeXY, noX, array6, countPairs, nestParen

Project Name	Assign 15 – Recursion Practice
Class 1 Name	RecursionClient
Class 2 Name	RecursionMethods

Project Name	AreaFill
Class 1 Name	GraphMap: represents an image
Class 2 Name	AreaFillApp: starts application
Class 3 Name	AreaFillFrame:manages buttons, etc.

RUBRIC	
Greatest Common Factor No EUCLIDS ALGORITHM	20
Raise Integer to a Power	20
GraphMap IsInBounds	20
areaFill()	30
GraphMap Constructor, reads in length and width, populates *'s and .'s	30
Comments	10
Codingbat recursion1	52
TOTAL	152


Artificial_Intelligence
Dynamic_Memory
Waterfall_Model
Nested_Loops
Stacks
Short_Circuit_Evaluation
Insertion_Sort
Binary_Search
Inheritance
Files
Linear_nlogn
Extreme_Programming
Client_Server
Hong_Kong_Fooey
VideoGames
String
Recursion
Search
File_Processing
Arrays
Selection_Sort
Big_O_Notation
2D_Arrays
Hangman
ArrayList