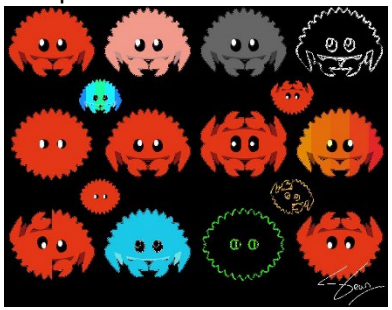
Sean Reyboz

Completed:



Original:

```
# CS 120 - Project 2
# Title: "The Rustacean's Evolution"
# Name: Sean Reyboz
def collage():
  setMediaPath()
  ferris = makePicture(getMediaPath("ferris.png"))
  #ferris = makePicture(pickAFile())
  WIDTH = getWidth(ferris)
  HEIGHT = getHeight(ferris)
  # Perform image manipulations
  blendFerris = blendWhite(ferris)
  blendFerrisBlackBg = changeColor(blendFerris, gray, black, 25)
  grayscaleFerris = grayscale(ferris)
  mirroredTopFerris = mirrorTop(ferris)
  mirroredRightFerris = mirrorRight(ferris)
  mirroredBottomFerris = mirrorBottom(ferris)
  mirroredLeftFerris = mirrorLeft(ferris)
  edgeFerris = edge(ferris, 10)
  edgeColorFerris = edgeColor(ferris)
  edgeColorMirrorTopFerris = mirrorTop(edgeColorFerris)
  negativeFerris = negative(ferris)
  negativeFerrisBlackBg = changeColor(negativeFerris, white, black, 10)
  reversedHalfFerris = reverseHalf(ferris)
  reversedFerris = reverse(ferris)
  rainbowFerris = horizontalRainbow(ferris)
  negativeRainbowFerris = horizontalRainbow(negativeFerrisBlackBg)
  mirroredLeftRainbowFerris = horizontalRainbow(mirroredLeftFerris)
  rainbowNegativeMirroredRightFerris = mirrorRight(negativeRainbowFerris)
  # Create canvas
  canvas = makeEmptyPicture(1000, 736, black)
```

```
  # Copy images into canvas  ## -- 1st row
  copyInto(ferris, canvas, 0, 0)
  copyInto(blendFerrisBlackBg, canvas, WIDTH, 0)
  copyInto(grayscaleFerris, canvas, 2 * WIDTH, 0)
  copyInto(edgeFerris, canvas, 3 * WIDTH, 0)
  ## -- 2nd row
  copyInto(mirroredTopFerris, canvas, 0, HEIGHT + 58)
  copyInto(mirroredRightFerris, canvas, WIDTH, HEIGHT + 58)
  copyInto(mirroredBottomFerris, canvas, 2 * WIDTH, HEIGHT + 58)
  copyInto(mirroredLeftRainbowFerris, canvas, 3 * WIDTH, HEIGHT + 58)
  ## -- 3rd row
  copyInto(reversedHalfFerris, canvas, 0, 2 * HEIGHT + 116)
  copyInto(negativeFerrisBlackBg, canvas, WIDTH, 2 * HEIGHT + 116)
  copyInto(edgeColorMirrorTopFerris, canvas, 2 * WIDTH, 2 * HEIGHT + 116)
  copyInto(reversedFerris, canvas, 3 * WIDTH, 2 * HEIGHT + 116)
  ## -- Small pictures between rows
  startX = WIDTH - (WIDTH / 4)
  startY = HEIGHT - (HEIGHT / 18)
  scaleDown(rainbowNegativeMirroredRightFerris, canvas, startX, startY)
  startX = (3 * WIDTH) - (WIDTH / 4)
  scaleDown(reversedFerris, canvas, startX, startY)
  startX = WIDTH - (WIDTH / 4)
  startY = (2 * HEIGHT) + (HEIGHT / 4)
  scaleDown(mirroredTopFerris, canvas, startX, startY)
  startX = (3 * WIDTH) - (WIDTH / 4)
  scaleDown(changeColor(edge(reversedHalfFerris, 10), white, orange, 10), canvas, startX, startY)
  # Sign the picture
  sig = makePicture(getMediaPath("signature.png"))
  sigX = getWidth(canvas) - getWidth(sig)
  sigY = getHeight(canvas) - getHeight(sig)
  chromakeySig(sig, canvas, sigX, sigY)
  # Show final canvas and export it to a jpeg image
  show(canvas)
  writePictureTo(canvas, getMediaPath() + "Rustacean's evolution.jpg")
# ----- FUNCTIONS -----
# horizontalRainbow()
# Modifies the green and blue colors of a given image by 50px horizontal chunks
def horizontalRainbow(pic):
  newPic = duplicatePicture(pic)
  width = getWidth(newPic)
  height = getHeight(newPic)
  for x in range(0, width):
    for y in range(0, height):
      px = getPixel(newPic, x, y)
      if x < 50:
```

```
        setGreen(px, getGreen(px) * 2.5)
        setBlue(px, getBlue(px) * 0.2)
      elif x < 100:
        setGreen(px, getGreen(px) * 2)
        setBlue(px, getBlue(px) * 0.5)
      elif x < 150:
        setGreen(px, getGreen(px) * 1.8)
        setBlue(px, getBlue(px) * 0.7)
      elif x < 200:
        setGreen(px, getGreen(px) * 1.2)
        setBlue(px, getBlue(px) * 1.2)
      else:
        setGreen(px, getGreen(px) * 0.7)
        setBlue(px, getBlue(px) * 1.8)
  return newPic
# reverse()
# Reverse the given picture upside down
def reverse(pic):
  width = getWidth(pic)
  height = getHeight(pic)
  targetPic = makeEmptyPicture(width, height, black)
  targetX = 0
  for x in range(0, width):
    targetY = height - 1
    for y in range(0, height):
      currPx = getPixelAt(pic, x, y)
      targPx = getPixel(targetPic, targetX, targetY)
      setColor(targPx, getColor(currPx))
      targetY -= 1
    targetX += 1
  return targetPic
# reverseHalf()
# Flip the right side of a given image upside down

def reverseHalf(pic):
  newPic = duplicatePicture(pic)
  pixels = getAllPixels(newPic)
  width = getWidth(newPic)
  height = getHeight(newPic)
  for x in range(0, width):
    for y in range(0, height):
      pxStart = getPixelAt(newPic, x, y)
      pxEnd = getPixelAt(newPic, width - x - 1, height - y - 1)
      endColor = getColor(pxEnd)
      setColor(pxStart, endColor)
```

```
  return newPic

# grayscale()
# Create a grayscaled version of a given picture
def grayscale(pic):
  newPic = duplicatePicture(pic)
  for px in getPixels(newPic):
    redCol = getRed(px)
    greenCol = getGreen(px)
    blueCol = getBlue(px)
    average = (redCol + greenCol + blueCol) / 3
    newColor = makeColor(average, average, average)
    setColor(px, newColor)
  return newPic
# negative()
# Create a negative version of a given picture
def negative(pic):
  newPic = duplicatePicture(pic)
  for px in getPixels(newPic):
    redCol = getRed(px)
    greenCol = getGreen(px)
    blueCol = getBlue(px)
    negColor = makeColor(255 - redCol, 255 - greenCol, 255 - blueCol)
    setColor(px, negColor)
  return newPic

# blendWhite()
# Increases the "lightness" of a given picture
def blendWhite(picture):
  newPic = duplicatePicture(picture)
  amount = 0.5
  for pixel in getPixels(newPic):
    newRed = 255 * amount + getRed(pixel) * (1 - amount)
    newGreen = 255 * amount + getGreen(pixel) * (1 - amount)
    newBlue = 255 * amount + getBlue(pixel) * (1 - amount)
    newColor = makeColor(newRed, newGreen, newBlue)
    setColor(pixel, newColor)
  return newPic
# mirrorTop()
# Mirror the top half of a given picture
def mirrorTop(source):
  newPic = duplicatePicture(source)
  width = getWidth(newPic)
  height = getHeight(newPic)
  mirrorPoint = height / 2
```

```
  for x in range(0, width):
    for y in range(0, mirrorPoint):
      topPixel = getPixelAt(newPic, x, y)
      bottomPixel = getPixelAt(newPic, x, height - y - 1)
      topColor = getColor(topPixel)
      setColor(bottomPixel, topColor)
  return newPic
# mirrorBottom()
# Mirror the bottom half of a given picture
def mirrorBottom(source):
  newPic = duplicatePicture(source)
  width = getWidth(newPic)
  height = getHeight(newPic)
  mirrorPoint = height / 2
  for x in range(0, width):
    for y in range(0, mirrorPoint):
      topPixel = getPixelAt(newPic, x, y)
      bottomPixel = getPixelAt(newPic, x, height - y - 1)
      bottomColor = getColor(bottomPixel)
      setColor(topPixel, bottomColor)
  return newPic
# mirrorRight()
# Mirror the right half of a given picture
def mirrorRight(source):
  newPic = duplicatePicture(source)
  width = getWidth(newPic)
  height = getHeight(newPic)
  mirrorPoint = width / 2
  for x in range(0, mirrorPoint):
    for y in range(0, height):
      leftPixel = getPixelAt(newPic, x, y)
      rightPixel = getPixelAt(newPic, width - x - 1, y)
      rightColor = getColor(rightPixel)
      setColor(leftPixel, rightColor)
  return newPic

# mirrorLeft()
# Mirror the left half of a given picture
def mirrorLeft(source):
  newPic = duplicatePicture(source)
  width = getWidth(newPic)
  height = getHeight(newPic)
  mirrorPoint = width / 2
  for x in range(0, mirrorPoint):
    for y in range(0, height):
```

```
      leftPixel = getPixelAt(newPic, x, y)
      rightPixel = getPixelAt(newPic, width - x - 1, y)
      leftColor = getColor(leftPixel)
      setColor(rightPixel, leftColor)
  return newPic

# edgeColor()
# Draw a colorful outline of a given picture
def edgeColor (pic):
  newPic = duplicatePicture(pic)
  for px in getPixels(newPic):
    x = getX(px)
    y = getY(px)
    if y < getHeight(newPic) - 1 and x < getWidth(newPic) - 1:
      colorSum = getRed(px) + getGreen(px) + getBlue(px)
      pixelOverOne = getPixel(newPic, x + 1, y + 1)
      colorSumOverOne= getRed(pixelOverOne) + getGreen(pixelOverOne) + getBlue(pixelOverOne)
      colorDifference = abs(colorSum - colorSumOverOne)
      if colorDifference < 25:
        setColor(px, black)
      elif colorDifference > 25 and colorDifference < 60:
        setColor(px, yellow)
      else:
        setColor(px, green)
  return newPic
# changeColor()
# Replace the given color with another one
def changeColor(pic, targetColor, newColor, tolerance):
  newPic = duplicatePicture(pic)
  for px in getPixels(newPic):
    pxColor = getColor(px)
    if distance(pxColor, targetColor) <= tolerance:
      setColor(px, newColor)
  return newPic
# scaleDown()
# Scale a given picture down by 50%
def scaleDown(pic, canvas, startX, startY):
  width = getWidth(pic)
  height = getHeight(pic)
  sourceX = 0
  for targetX in range(startX, startX + width / 2):
    sourceY = 0
    for targetY in range(startY, startY + height / 2):
      sourcePx = getPixelAt(pic, int(sourceX), int(sourceY))
      sourceColor = getColor(sourcePx)
```

```
      targetPx = getPixelAt(canvas, targetX, targetY)
      setColor(targetPx, sourceColor)
      sourceY += 1.0 / 0.5
    sourceX += 1.0 / 0.5

# edge()
# Draw the edge of a given picture
def edge(pic, tolerance = 25):
  newPic = duplicatePicture(pic)
  for px in getAllPixels(newPic):
    x = getX(px)
    y = getY(px)
    if y < getHeight(newPic) - 1 and x < getWidth(newPic) - 1:
      colorSum = getRed(px) + getGreen(px) + getBlue(px)
      pixelOverOne = getPixelAt(newPic, x + 1, y + 1)
      colorSumOverOne= getRed(pixelOverOne) + getGreen(pixelOverOne) + getBlue(pixelOverOne)
      colorDifference = abs(colorSum - colorSumOverOne)
      if colorDifference > tolerance:
        setColor(px, white)
      if colorDifference <= tolerance:
        setColor(px, black)
  return newPic
# chromakeySig()
# Add a signature to the canvas at the given coordinates
def chromakeySig(pic, canvas, targetX, targetY):
  width = getWidth(pic)
  height = getHeight(pic)
    for sX in range(0, width):
    for sY in range(0, height):
      sPx = getPixelAt(pic, sX, sY)
      sColor = getColor(sPx)
      targetPx = getPixelAt(canvas, sX + targetX, sY + targetY)
      if distance(gray, sColor) < 180:
        setColor(targetPx, white)
```