NAME:
EMAIL:
SIGNATURE:
CIRCLE COURSE SECTION: TTH 11-1 MW 1-3 TTH 4-6 MW 6-8
MW 4-6 MW 11-1 MW 9-11

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CIS 166 & CMP 230 Final Exam, Version 1, Fall 2012

1. What will the following code print:

```
places = "Connecticut*New York*New Jersey*Pennsylvania"
num = places.count("*")
states = places.split("*")
print("There are", num, "states")
print(states[0], states[-1])
mess = "cahbecedseef gchaikje"
eat = ""
for i in range(len(mess)):
    if i % 2 == 0:
        print(mess[i])
        eat = eat + mess[i]
print("I love", states[1], eat)
```

Output:

2. Write a **program** that asks the user to enter the number of grams, and prints out the equivalent number of ounces.
Helpful Fact: 1 gram = 28.3495 ounces

3. Fill in the missing function definitions for this program:

```
def main():
    w = setUp()                 #Creates and returns a graphics window
    x1,y1,x2,y2 = userInput()   #Asks user for 4 inputs and returns numbers entered
    displayLine(w,x1,y1,x2,y2) #Draws a line from (x1,y1) to (x2,y2) on window w
    conclusion(w)               #Gets a mouse click and closes window w
main()
```

(That is, write the functions `setUp()`, `userInput()`, `displayLine()` and `conclusion()`.)

4. Write a **function** that takes as two parameters: the zone and the ticket type, and returns the Copenhagen Transit fare.

   If the zone is 2 or smaller and the ticket type is "adult," the fare is 23. If the zone is 2 or smaller and the ticket type is "child," the fare is 11.5. If the zone is 3 and the ticket type is "adult," the fare is 34.5. If the zone is 3 or 4 and the ticket type is "child," the fare is 23. If the zone is 4 and the ticket type is "adult," the fare is 46. If the zone is greater than 4, return a negative number (since your calculator does not handle inputs that high).

5. What is returned when the function is invoked on the inputs below:

```
def enigma1(x,y,z):
    if x == len(y):
        return(z)
    elif x < len(y):
        return(y[0:x])
    else:
        s = cont1(z)
        return(s+y)
```

```
def cont1(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(a) `enigma1(7,"caramel","dulce de leche")`

Return: _____

(b) `enigma1(3,"cupcake","vanilla")`

Return: _____

(c) `enigma1(10,"pie","nomel")`

Return: _____

6. Given the following program and input file, what is printed:

```
def sixV1():
    infile = open("in1.txt", "r")
    for line in infile.readlines():
        if line.find("St") == 0:
            print(line)
    infile.close()
sixV1()
```

in1.txt

```
S'more
Sopaipilla
Stack cake
Strawberry Delight
Strawberry rhubarb pie
```

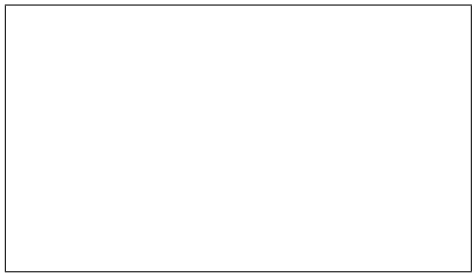
Output:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the number of times the letter `K` occurs in the file.

8. What is the graphical output:

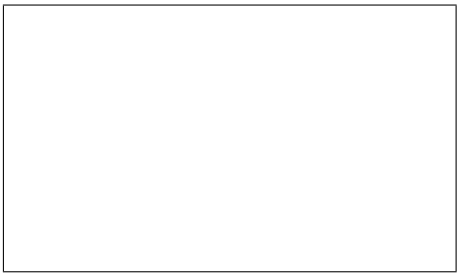
(a)
```
from turtle import *
def once1(t,x):
    for i in range(4):
        forward(x)
        right(90)
t = Turtle()
once1(t,50)
```

Output:



(b)
```
from turtle import *
def mystery1(t,x):
    for i in range(4):
        forward(x)
        right(90)
    if x > 0:
        mystery1(t,x-10)
t = Turtle()
mystery1(t,50)
```

Output:



9. Write the python code for the algorithm below:
(from `http://rosettacode.org/wiki/Sorting_algorithms/Stooge_sort`)

```
algorithm stoogesort(L, i, j)
     if L[j] < L[i] then
         swap L[i] and L[j]
     if j - i > 1 then
         t := (j - i + 1)/3
         stoogesort(L, i, j-t)
         stoogesort(L, i+t, j)
         stoogesort(L, i, j-t)
     return L
```

10. Write a simulation for a *Hangman* word guessing game, where players take turns guessing letters of the secret word. A player loses if they make 6 bad guesses. You are to simulate a version of the game with two players. Your code should explain to the user what is going on; ask for the probability of the player correctly guessing a letter and the number of letters in the initial word; simulate the game; and report the winner. *You are not to write the entire program, instead, do the following*:

    (a) Design the top level (main). At this first level of abstraction there should be no details pertaining to how the work is done, just function calls and assignments that tell us what will be done in terms of functions that read input, functions that do calculations, and functions that write output. Think of the racquetball program as a guide.

    (b) Refine the top level by writing code with two more levels of abstraction. More specifically, write out one of the functions that main calls on, and write out one of the functions that is called by that function.

Useful String Methods: (from p 140 of textbook)

Function	Meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` is centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of `sub` in `s`.
`s.find(sub)`	Find the first position where `sub` occurs in `s`.
`s.join(list)`	Concatenate `list` into a string using `s` as a separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` with all characters converted to lowercase.
`s.lstrip()`	Copy of `s` with leading whitespace removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns rightmost position.
`s.rjust(sub)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing whitespace removed.
`s.split()`	Split `s` into a list of substrings.
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

Useful Turtle Methods: (from `http://docs.python.org/3.0/library/turtle.html`)

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

NAME:
EMAIL:
SIGNATURE:
CIRCLE COURSE SECTION: TTH 11-1 MW 1-3 TTH 4-6 MW 6-8
MW 4-6 MW 11-1 MW 9-11

Lehman College, CUNY
CIS 166 & CMP 230 Final Exam, Version 2, Fall 2012

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
colorStr = "purple^red^blue^green^yellow"
num = colorStr.count("^")
colors = colorStr.split("^")
print("There are", num, "colors")
print(colors[0], colors[-1])
mess = "vaeblcvdeetf gchaikje"
eat = ""
for i in range(len(mess)):
    if i % 2 == 0:
        print(mess[i])
        eat = eat + mess[i]
print("I love", colors[1], eat, "!")
```

Output:

2. Write a **program** that asks the user to enter the number of ounces, and prints out the equivalent number of grams.
Helpful Fact: 1 ounce = 0.03527 grams.

3. Fill in the missing function definitions for this program:

```
def main():
    w = setUp()                 #Creates and returns a graphics window
    x1,y1,r = userInput()       #Asks user for 3 inputs and returns numbers entered
    displayCircle(w,x1,y1,r)    #Draws a circle at (x1,y1) with radius r on window w
    conclusion(w)               #Gets a mouse click and closes window w
main()
```

(That is, write the functions `setUp()`, `userInput()`, `displayCircle()` and `conclusion()`.)

4. Write a **function** that takes as two parameters: the zone and the duration, and returns the Barcelona metro and bus (TMB) fare.

   If the zone is 1 and duration is "oneDay," the fare is 7. If the zone is 1 and the duration is "tenRide," the fare is 9.45. If the zone is 1 and the duration is "oneMonth," the fare is 50.95. If the zone is 2 and duration is "oneDay," the fare is 11.15. If the zone is 2 and the duration is "tenRide," the fare is 18.75. If the zone is 2 and the duration is "oneMonth," the fare is 74.85. If the zone is greater than or equal to 3, return a negative number (since your calculator does not handle inputs that high).

5. What is returned when the function is invoked on the inputs below:

```
def enigma2(x,y,z):
    if x < y:
        return(z)
    elif x == y:
        return(z+z)
    else:
        s = cont2(z)
        return(s)
```

```
def cont2(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(a) `enigma2(1,2,"cake")`

Return: _____

(b) `enigma2(100,100,"Yum")`

Return: _____

(c) `enigma2(10,-1,"eipdum")`

Return: _____

6. Given the following program and input file, what is printed:

```
def sixV2():
    infile = open("in2.txt", "r")
    for line in infile.readlines():
        if line.count(" ") >= 2:
            print(line)
    infile.close()
sixV2()
```

in2.txt

```
Caramel
Carrot cake
Chocolate brownie
Chocolate chip cookie
Chocolate-covered potato chips
```

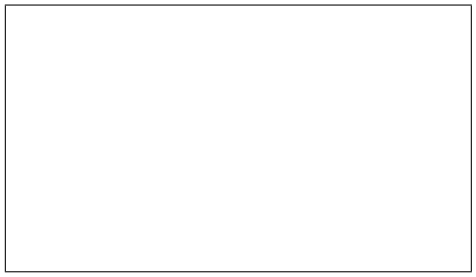
Output:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the number of times the letter `A` occurs in the file.

8. What is the graphical output:

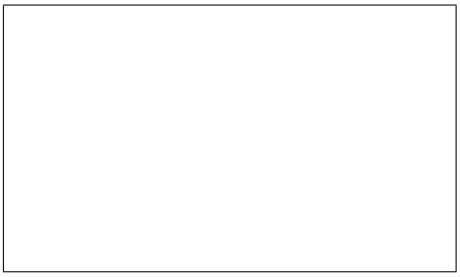
(a)
```
from turtle import *
def once2(t,x):
    for i in range(6):
        forward(x)
        right(60)
t = Turtle()
once2(t,30)
```

Output:



(b)
```
from turtle import *
def mystery2(t,x):
    for i in range(6):
        forward(x)
        right(60)
    if x > 0:
        mystery2(t,x-10)
t = Turtle()
mystery2(t,30)
```

Output:



9. Write the python code for the algorithm below: (from `http://en.wikipedia.org/wiki/Binary_search_algorithm`)

```
binary_search(A, key, imin, imax)
  if (imax < imin)
      return -1;
  else
      imid = imin + (imax-imin)/2
      if (A[imid] > key)
          return binary_search(A, key, imin, imid-1)
      else if (A[imid] < key)
          return binary_search(A, key, imid+1, imax)
      else
        return imid;
    }
}
```

10. Write a simulation for a *Pin-the-Tail-on-the-Donkey* game, where players take turns trying to pin the tail on the target. A player wins if they come within 5 centimeters of the target. You are to simulate a version of the game with multiple players. Your code should explain to the user what is going on; ask for the probability of the player hitting the target on any given turn and the number of players; simulate the game; and report the winner. *You are not to write the entire program, instead, do the following*:

    (a) Design the top level (main). At this first level of abstraction there should be no details pertaining to how the work is done, just function calls and assignments that tell us what will be done in terms of functions that read input, functions that do calculations, and functions that write output. Think of the racquetball program as a guide.

    (b) Refine the top level by writing code with two more levels of abstraction. More specifically, write out one of the functions that main calls on, and write out one of the functions that is called by that function.

Useful String Methods: (from p 140 of textbook)

Function	Meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` is centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of `sub` in `s`.
`s.find(sub)`	Find the first position where `sub` occurs in `s`.
`s.join(list)`	Concatenate `list` into a string using `s` as a separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` with all characters converted to lowercase.
`s.lstrip()`	Copy of `s` with leading whitespace removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns rightmost position.
`s.rjust(sub)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing whitespace removed.
`s.split()`	Split `s` into a list of substrings.
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

Useful Turtle Methods: (from `http://docs.python.org/3.0/library/turtle.html`)

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

NAME:
EMAIL:
SIGNATURE:
CIRCLE COURSE SECTION: TTH 11-1 MW 1-3 TTH 4-6 MW 6-8
MW 4-6 MW 11-1 MW 9-11

Lehman College, CUNY
CIS 166 & CMP 230 Final Exam, Version 3, Fall 2012

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
geology = "dirt-soil-mud-rocks-stones"
num = geology.count("-")
stuff = geology.split("-")
print("There are", num, "kinds")
print(stuff[0], stuff[-1])
mess = "ate*iss*ipp*M*ake*C*hocol"
pieces = mess.split("*")
print("I love", pieces[3],end="")
for i in range(2):
    print(pieces[1],end="")
print(pieces[2]+"i", stuff[2])
print(stuff[2], pieces[5]+pieces[4], "!")
```

Output:

2. Write a **program** that asks the user to enter the number of pounds, and prints out the equivalent number of kilograms.
Helpful Fact: 1 pound = 0.4536 kilograms

3. Fill in the missing function definitions for this program:

```
def main():
    w = setUp()                 #Creates and returns a graphics window
    x1,y1,x2,y2 = userInput()   #Asks user for 4 inputs and returns numbers entered
    displayLine(w,x1,y1,x2,y2) #Draws a line from (x1,y1) to (x2,y2) on window w
    conclusion(w)               #Gets a mouse click and closes window w
main()
```

(That is, write the functions `setUp()`, `userInput()`, `displayLine()` and `conclusion()`.)

4. Write a **function** that takes as two parameters the zone and the direction, and returns the San Fransicso Bay Area Rapid Transit (BART) fare.

   If the zone is 2 or smaller, the fare is 1.75. If the zone is 3 and the direction is outbound, the fare is 2.15. If the zone is 3 and the direction is inbound, the fare is 2.45. If the zone is 4 and the direction is outbound, the fare is 2.65. If the zone is 4 and the direction is inbound, the fare is 2.95. If the zone is greater than 4, return a negative number (since your calculator does not handle inputs that high).

5. What is returned when the function is invoked on the inputs below:

```
def enigma3(x,y,z):
    if x < y:
        return(z)
    elif x == y:
        return(z+z)
    else:
        s = cont3(z)
        return(s)
```

```
def cont3(st):
    r = ""
    for i in range(len(st)):
        if i % 2 == 0:
            r = r + st[i]
    return(r)
```

(a) `enigma3(1,2,"gelato")`

Return: _____

(b) `enigma3(100,100,"dum")`

Return: _____

(c) `enigma3(10,-1,"ibcced ecfrgehaim")`

Return: _____

6. Given the following program and input file, what is printed:

```
def sixV3():
    infile = open("in3.txt", "r")
    for line in infile.readlines():
        if len(line) < 14:
            print(line)
    infile.close()
sixV3()
```

in3.txt

```
Bacon sundae
Banana pudding
Banana split
Bananas Foster
Blondie (confection)
```

Output:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the number of times the letter `T` occurs in the file.

8. What is the graphical output:

(a)
```
from turtle import *
def once3(t,x):
    for i in range(4):
        forward(x)
        right(90)
t = Turtle()
once3(t,50)
```

Output:

(b)
```
from turtle import *
def mystery3(t,x):
    for i in range(4):
        forward(x)
        right(90)
    if x < 100:
        mystery3(t,x+10)
t = Turtle()
mystery3(t,50)
```

Output:

9. Write the python code for the algorithm below: (from `http://en.wikipedia.org/wiki/Catalan_number`)

```
algorithm catalan(n)
     if n is 0 then
         sum = 1
     else
        sum = 0
        for i from 0 to n
            sum = sum + catalan(i)*catalan(n-i)
    return sum
```

10. Write a simulation for a *numbers guessing* game, where players take turns trying to guess the secret number. A player wins if they correctly guess the number. You are to simulate a version of the game with multiple players. Your code should explain to the user what is going on; ask for the probability of the player hitting the target on any given turn and the number of players; simulate the game; and report the winner. *You are not to write the entire program, instead, do the following*:

    (a) Design the top level (main). At this first level of abstraction there should be no details pertaining to how the work is done, just function calls and assignments that tell us what will be done in terms of functions that read input, functions that do calculations, and functions that write output. Think of the racquetball program as a guide.

    (b) Refine the top level by writing code with two more levels of abstraction. More specifically, write out one of the functions that main calls on, and write out one of the functions that is called by that function.

Useful String Methods: (from p 140 of textbook)

Function	Meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` is centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of `sub` in `s`.
`s.find(sub)`	Find the first position where `sub` occurs in `s`.
`s.join(list)`	Concatenate `list` into a string using `s` as a separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` with all characters converted to lowercase.
`s.lstrip()`	Copy of `s` with leading whitespace removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns rightmost position.
`s.rjust(sub)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing whitespace removed.
`s.split()`	Split `s` into a list of substrings.
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

Useful Turtle Methods: (from `http://docs.python.org/3.0/library/turtle.html`)

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving

NAME:
EMAIL:
SIGNATURE:
CIRCLE COURSE SECTION: TTH 11-1 MW 1-3 TTH 4-6 MW 6-8
MW 4-6 MW 11-1 MW 9-11

Lehman College, CUNY
CIS 166 & CMP 230 Final Exam, Version 4, Fall 2012

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
letterStr = "AXBXCXDXEXFXGXHXIXJ"
num = letterStr.count("X")
letters = letterStr.split("X")
print("There are", num, "letters")
print(letters[0], letters[-1])
mess = "h?o?ola?te ?hi?p ?ook?ies"
ends = mess.split("?")
print(letters[-2], "love")
for i in range(len(ends)):
    if i % 2 == 0:
        print(letters[2], end="")
    print(ends[i],end="")
```

Output:

2. Write a **program** that asks the user to enter the number of kilograms, and prints out the equivalent number of pounds.
Helpful Fact: 1 kilogram = 2.2056 pounds.

3. Fill in the missing function definitions for this program:

```
def main():
    w = setUp()                 #Creates and returns a graphics window
    x1,y1,r = userInput()       #Asks user for 3 inputs and returns numbers entered
    displayCircle(w,x1,y1,r)    #Draws a circle at (x1,y1) with radius r on window w
    conclusion(w)               #Gets a mouse click and closes window w
main()
```

(That is, write the functions `setUp()`, `userInput()`, `displayCircle()` and `conclusion()`.)

4. Write a **function** that takes as two parameters: the number of stations and the ticket type, and returns the Bangkok metro fare.

   If the number of stations is 1 or smaller and the ticket type is "senior," the fare is 8. If the number of stations is 1 or smaller and the ticket type is "adult," the fare is 16. If the number of stations is 2 and the ticket type is "senior," the fare is 9. If the number of stations is 2 and the ticket type is "adult," the fare is 18. If the number of stations is greater than or equal to 3, return a negative number (since your calculator does not handle inputs that high).

5. What is returned when the function is invoked on the inputs below:

```
def enigma4(x,y,z):
    if x < y:
        return(z)
    elif x == y:
        return(z+z)
    else:
        s = cont4(z)
        return(s)
```

```
def cont4(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(a) `enigma4(3,4,"blondie")`

Return: _____

(b) `enigma4(50,50,"Yum")`

Return: _____

(c) `enigma4(10,0,"einworb")`

Return: _____

6. Given the following program and input file, what is printed:

```
def sixV4():
    infile = open("in4.txt", "r")
    for line in infile.readlines():
        if line.find("Apple") == 0:
            print(line)
    infile.close()
sixV4()
```

in4.txt

```
Ambrosia (fruit salad)
Angel food cake
Apple crisp
Apple dumpling
Apple pie
```

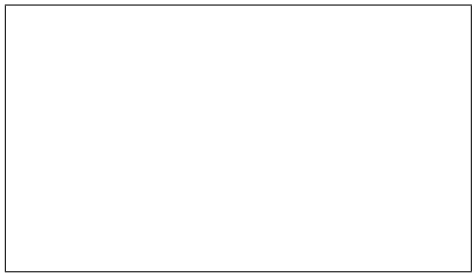
Output:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the number of times the letter `E` occurs in the file.

8. What is the graphical output:

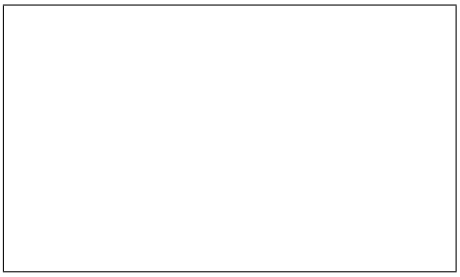
(a)
```
from turtle import *
def once4(t,x):
    for i in range(3):
        forward(x)
        right(120)
t = Turtle()
once4(t,30)
```

Output:



(b)
```
from turtle import *
def mystery4(t,x):
    for i in range(3):
        forward(x)
        right(120)
    if x > 0:
        mystery4(t,x-10)
t = Turtle()
mystery4(t,30)
```

Output:



9. Write the python code for the algorithm below: (from `http://en.wikipedia.org/wiki/Fibonacci_number`)

```
algorithm fibonacci(n)
    if n is 0 or a negative number
        return 0
    else if n < 2
        return 1
    else
        return fibonacci(n-1)+fibonacci(n-2)
```

10. Write a simulation for a *darts* game, where players take turns trying to hit the target with a dart. A player wins if they come within 5 centimeters of the target. You are to simulate a version of the game with multiple players. Your code should explain to the user what is going on; ask for the probability of the player hitting the target on any given turn and the number of players; simulate the game; and report the winner. *You are not to write the entire program, instead, do the following*:

    (a) Design the top level (main). At this first level of abstraction there should be no details pertaining to how the work is done, just function calls and assignments that tell us what will be done in terms of functions that read input, functions that do calculations, and functions that write output. Think of the racquetball program as a guide.

    (b) Refine the top level by writing code with two more levels of abstraction. More specifically, write out one of the functions that main calls on, and write out one of the functions that is called by that function.

Useful String Methods: (from p 140 of textbook)

Function	Meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` is centered in a field of given width.
`s.count(sub)`	Count the number of occurrences of `sub` in `s`.
`s.find(sub)`	Find the first position where `sub` occurs in `s`.
`s.join(list)`	Concatenate `list` into a string using `s` as a separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` with all characters converted to lowercase.
`s.lstrip()`	Copy of `s` with leading whitespace removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns rightmost position.
`s.rjust(sub)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing whitespace removed.
`s.split()`	Split `s` into a list of substrings.
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Graphics Reference: (from p 108-111 of the textbook)

GraphWin Objects
`GraphWin(title, width, height)`
`plot(x,y,color)`
`plotPixel(x,y,color)`
`setBackground(color)`
`close()`
`getMouse()`
`checkMouse()`
`setCoords(xll,yll,xur,yur)`

Graphics Objects
`setFill(color)`
`setOutline(color)`
`setWidth(pixels)`
`draw(aGraphWin)`
`undraw()`
`move(dx,dy)`
`clone()`

Text Methods
`Text(anchorPoint, string)`
`setText(string)`
`getText()`
`getAnchor()`
`setFace(family)`
`setSize(point)`
`setStyle(style)`
`setTextColor(color)`

Point Methods
`Point(x,y)`
`getX()`
`getY()`

Line Methods
`Line(point1, point2)`
`setArrow(string)`
`getCenter()`
`getP1(), getP2()`

Circle Methods
`Circle(centerPoint, radius)`
`getCenter()`
`getRadius()`
`getP1(), getP2()`

Rectangle Methods
`Rectangle(point1,point2)`
`getCenter()`
`getP1(), getP2()`

Oval Methods
`Oval(point1, point2)`
`getCenter()`
`getP1(), getP2()`

Polygon Methods
`Polygon(P1, P2, P3,...)`
`getPoints()`

Useful Turtle Methods: (from `http://docs.python.org/3.0/library/turtle.html`)

Function	Meaning
`forward(d)`	Move turtle forward `d` steps
`backward(d)`	Move turtle backward `d` steps
`right(angle)`	Turn turtle `angle` degrees to the right
`left(angle)`	Turn turtle `angle` degrees to the left
`up()`	Pull the pen up no drawing when moving
`down()`	Pull the pen down drawing when moving