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, Spring 2013

1. What will the following code print:

```
s = "Football*Basketball*Hockey*Baseball*Soccer"
num = s.count("*") + 1
sports = s.split("*")
print("There are", num, "sports")
print("Two of them are",sports[0], sports[-2])
mess = "Poaoxci BnletRc caGnqu oaUxr Fd"
result = ""
for i in range(len(mess)):
    if i % 3 == 0:
        print(mess[i])
        result = result + mess[i]
print(sports[1], result)
```

Output:

2. Write a **program** that asks the user for a number of hours. Your program should then compute and print out the equivalent number of days and hours. For example, if the user entered `50 hours`, your program would output `2 days and 2 hours`.

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

```
def main():
    w = setUp()         # Creates and returns a graphics window

    c = userInput()     # Asks user for a color (either red, green, or blue)
                        # and returns color entered. userInput() must verify
                        # that the input is valid

    displayCircle(w,c) # Draws a circle of color c in the center of 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 name of a country (one of Japan, Indonesia, or Hungary) and an amount of U.S. currency. Your function should convert the amount of U.S. dollars into the currency of the given country. If the country passed to the function is not on the list, return -1. The conversion rates for these countries are $1 US = 83.85 Japanese Yen, 9639.99 Indonesian Rupiahs, and 0.00456 Hungarian Forints.

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

```
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, sum(a,b)
    return a
```

```
def sum(a,b):
    return a+b
```

(a) `fib(0)` **Return:**

(b) `fib(3)` **Return:**

(c) `fib(5)` **Return:**

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

```
def sixV1():
   infile=open("music.txt","r")
   for line in infile.readlines():
      if len(line)>8:
         print(line.upper())
      else:
         print(line)

sixV1()
```

music.txt

```
Samba, Brazil
Cumbia, Colombia
Syrtaki, Greece
Hip-hop
```

Output:

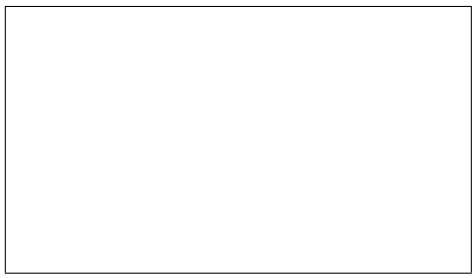
7. Write a **program** that reads in a text file, `infile.txt`, and prints out the line numbers of the lines containing the word `Python` in the file.

8. What is the graphical output:

(a)
```
from turtle import *
def once(t, x, r):
    for i in range(3):
        if r:
            t.right(120)
        else:
            t.left(120)
        t.forward(x)

t = Turtle()
once(t, 40, True)
once(t, 40, False)
```

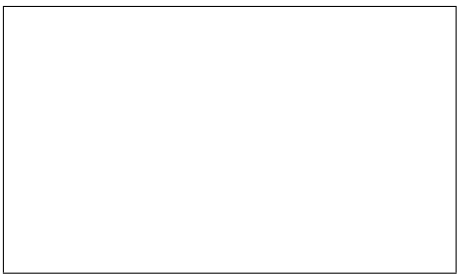
Output:



(b)
```
from turtle import *
def mystery(t, x, r):
    for i in range(3):
        if r:
            t.right(120)
        else:
            t.left(120)
        t.forward(x)
    if x > 20:
        mystery(t, x/2, r)

t = Turtle()
mystery(t, 40, True)
mystery(t, 40, False)
```

Output:



9. Write the python code for the algorithm below:

```
pow2(number)
    if number is less than or equal to 1
        return 2
    otherwise
        return 2 * pow2(number - 1)
```

10. Write a simulation for a *Dice* game between two players. In each round, both players roll a single six-sided die. The winner of the round is the player with the higher number. They will play a best of a series of rounds (3, 5, 7, etc...). A player wins the game when they win more than half the rounds. You are to simulate a version of the game. Your code should explain to the user what is going on; ask for the number of rounds; 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, Spring 2013

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
s = "Field Hockey+Swimming+Water Polo+Football+Basketball"
num = s.count("+") + 1
sports = s.split("+")
print("There are", num, "sports")
print("Two of them are",sports[1], sports[-1])
mess = "Qoauxca BrletRce crcx qvBnqa ocUxk"
result = ""
for i in range(len(mess)):
    if i % 3 == 0:
        print(mess[i])
        result = result + mess[i]
print(sports[1], result)
```

Output:

2. Write a **program** that asks the user for a number of days. Your program should then compute and print out the equivalent number of weeks and days. For example, if the user entered `25 days`, your program would output `3 weeks and 4 days`.

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

```
def main():
    w = setUp()         # Creates and returns a graphics window

    c = userInput()     # Asks user for a color (either red, green, or blue)
                        # and returns color entered. userInput() must verify
                        # that the input is valid

    displayCircle(w,c) # Sets the background of w to color c and draws a
                        # circle in the center of 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 name of a country (one of Guatemala, Denmark, or Costa Rica) and an amount of U.S. currency. Your function should convert the amount of U.S. dollars into the currency of the given country. If the country passed to the function is not on the list, return -1. The conversion rates are $1 US dollar = 0.13 Guatemalan Quetzals, 0.62 Danish Krone, 499.38 Costa Rican Colones.

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

```
def fct(x):
    f = 1
    while (x > 0):
        f = f * x
        x = decr(x)
    return f
```

```
def decr(x):
    return x-1
```

(a) `fct(3)` **Return:** _____

(b) `fct(4)` **Return:** _____

(c) `fct(5)` **Return:** _____

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

```
def sixV2():
   infile=open("music.txt","r")
   for line in infile.readlines():
      if (line.count("mbi"))>0:
         print line.replace("mb","ls")
      else:
         print(line)

sixV2()
```

music.txt

```
Samba, Brazil
Cumbia, Colombia
Syrtaki, Greece
Hip-hop
```

Output:

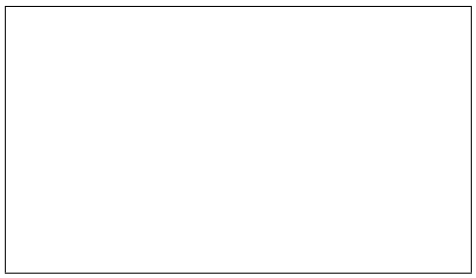
7. Write a **program** that reads in a text file, `infile.txt`, and prints out the line numbers of the lines containing the word `Prolog` in the file.

8. What is the graphical output:

(a)
```
from turtle import *
def once(t, x, r):
    for i in range(4):
        if r:
            t.right(90)
        else:
            t.left(90)
        t.forward(x)

t = Turtle()
once(t, 40, True)
once(t, 40, False)
```

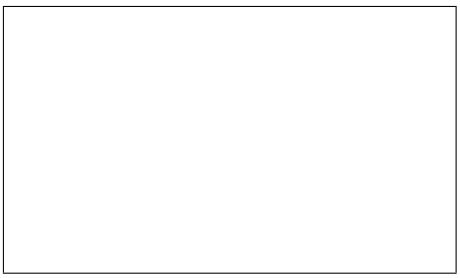
Output:



(b)
```
from turtle import *
def mystery(t, x, r):
    for i in range(4):
        if r:
            t.right(90)
        else:
            t.left(90)
        t.forward(x)
    if x > 20:
        mystery(t, x/2, r)

t = Turtle()
mystery(t, 40, True)
mystery(t, 40, False)
```

Output:



9. Write the python code for the algorithm below:

```
fact(number)
    if number is 1
        return 1
    otherwise
        return number * fact(number - 1)
```

10. Write a simulation for a simplified *Blackjack* game, where a player plays against the house. The player and the house each pick a number between 1 and 30. Whoever is closer to 21 without going over wins. If the player wins, they earn $5. If they lose, they pay $5. The game is over when either a) the player is out of money or b) they double their money. You are to simulate a version of the game with a single player. Your code should explain to the user what is going on; ask for the amount of money that the player starts with; simulate the game; and report whether the player lost their money or doubled it. *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, Spring 2013

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
s = "Ultimate Frisbee&Boxing&Wrestling&Soccer&Baseball"
num = s.count("&") + 1
sports = s.split("&")
print("There are", num, "sports")
print("Two of them are",sports[0],sports[-3])
mess = "Loaexcf Btle R FaGequioalxrd"
result = ""
for i in range(len(mess)):
    if i % 3 == 0:
        print(mess[i])
        result = result + mess[i]
print(sports[-1], result)
```

Output:

2. Write a **program** that asks the user for a number of weeks. Your program should then compute and print out the equivalent number of years and weeks. For example, if the user entered `58 weeks`, your program would output `1 year and 6 weeks`.

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

```
def main():
    w = setUp()         # Creates and returns a graphics window

    c = userInput()     # Asks user for a color (either red, green, or blue)
                        # and returns color entered. userInput() must verify
                        # that the input is valid

    displayCircle(w,c) # Draws a circle of color c in the center of 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 name of a country (one of Samoa, Venezuela, or Ukraine) and an amount of U.S. currency. Your function should convert the amount of U.S. dollars into the currency of the given country. If the country passed to the function is not on the list, return -1. The conversion rates are $1 US Dollar = 2.2 Samoan Talas, 4.30 Venezuelan Bolivars, and 8.10 Ukrainian Hryvnas.

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

```
def pos(list):
    allpos=True
    for elem in list:
        allpos= allpos and pos_help(elem)
    return allpos
```

```
def pos_help(m):
    return m>0
```

(a) `pos([1,2,3])`

Return:

(b) `pos([2,3,-1,7])`

Return:

(c) `pos([ ])`

Return:

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

```
def sixV3():
   infile=open("music.txt","r")
   for line in infile.readlines():
      if (line.count("mba"))>0:
         print line.replace("mba","lsa")
      else:
         print(line)

sixV3()
```

music.txt

```
Samba, Brazil
Cumbia, Colombia
Syrtaki, Greece
Hip-hop
```

Output:

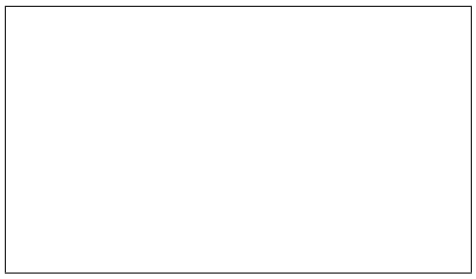
7. Write a **program** that reads in a text file, `infile.txt`, and prints out the line numbers of the lines containing the word `Java` in the file.

8. What is the graphical output:

(a)
```
from turtle import *
def once(t, x, r):
    for i in range(3):
        if r:
            t.right(120)
        else:
            t.left(120)
        t.forward(x)

t = Turtle()
t.left(90)
once(t, 40, True)
once(t, 40, False)
```

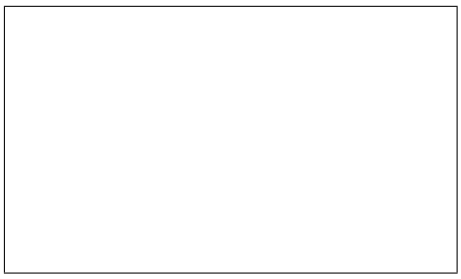
Output:



(b)
```
from turtle import *
def mystery(t, x, r):
    for i in range(3):
        if r:
            t.right(120)
        else:
            t.left(120)
        t.forward(x)
    if x > 20:
        mystery(t, x/2, r)

t = Turtle()
t.left(90)
mystery(t, 40, True)
mystery(t, 40, False)
```

Output:



9. Write the python code for the algorithm below:

```
add(number)
    if number is less than 1
        return 0
    otherwise
        return number + add(number - 1)
```

10. Write a simulation for a *lottery* game, where a player takes turns trying to pick the right number. A player wins if they correctly guess the number. A player will continue to play until either a) they run out of money, or b) they win the lottery. At each round the player pays $1 and picks a number at random from numbers between 1 and 20,000. At the same time, a lottery ball is picked with a number between 1 and 20,000. If the player's number and the ball match, the player wins $1,000,000. Otherwise, the player plays again. You are to simulate a version of the game with a single player. Your code should explain to the user what is going on; ask how much money the player has to start; simulate the game; and report how much money the player had at the end of the game. *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, Spring 2013

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. What will the following code print:

```
s = "Racket Ball,Gymnastics,Soccer,Hand Ball,Fencing"
num = s.count(",") + 1
sports = s.split(",")
print("There are", num, "sports")
print("Two of them are",sports[1],sports[-2])
mess = "Gpqorkaqxlzc ozKc erhewkpazefsr"
result = ""
for i in range(len(mess)):
    if i % 3 == 0:
        print(mess[i])
        result = result + mess[i]
print(sports[-3], result)
```

Output:

2. Write a **program** that asks the user for a number of minutes. Your program should then compute and print out the equivalent number of hours and minutes. For example, if the user entered `250 minutes`, your program would output `4 hours and 10 minutes`.

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

```
def main():
    w = setUp()         # Creates and returns a graphics window

    c = userInput()     # Asks user for color (either red, green, or blue)
                        # and returns color entered. userInput() must verify
                        # that the input is valid

    displayCircle(w,c) # Sets the background of w to color c and
                        # draws a circle in the center of 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 name of a country (one of Russia, Qatar, or Poland) and an amount of U.S. currency. Your function should convert the amount of U.S. dollars into the currency of the given country. If the country passed to the function is not on the list, return -1. The conversion rates are $1 US Dollar = 30.80 Russian Rubles, 0.275 Qatari Riyals, and 3.11 Polish Zlotys.

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

```
def filterpos(l):
   newlist=[]
   for el in l:
      if el>0:
         newlist=app(el,newlist)
   return newlist
```

```
def app(el,newlist):
   return newlist+[el]
```

(a) `filterpos([-1,-3,-9])`

Return: _____

(b) `filterpos([1,-1,2])`

Return: _____

(c) `filterpos([2,3,4,7])`

Return: _____

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

```
def sixV4():
   infile=open("music.txt","r")
   for line in infile.readlines():
      if (line.count("a"))>0:
         print line.replace("a","i")
      else:
         print(line)

sixV4()
```

music.txt

```
Samba, Brazil
Cumbia, Colombia
Syrtaki, Greece
Hip-hop
```

Output:

7. Write a **program** that reads in a text file, `infile.txt`, and prints out the line numbers of the lines containing the word `Fortran` in the file.

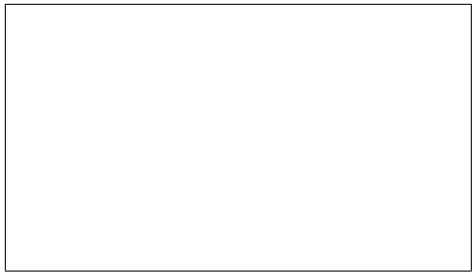
8. What is the graphical output:

(a)
```
from turtle import *

def once(t, x, r):
    for i in range(4):
        if r:
            t.right(90)
        else:
            t.left(90)
        t.forward(x)

t = Turtle()
t.left(90)
once(t, 40, True)
once(t, 40, False)
```

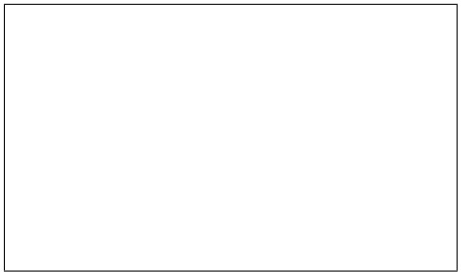
Output:



(b)
```
from turtle import *
def mystery(t, x, r):
    for i in range(4):
        if r:
            t.right(90)
        else:
            t.left(90)
        t.forward(x)
    if x > 20:
        mystery(t, x/2, r)

t = Turtle()
t.left(90)
mystery(t, 40, True)
mystery(t, 40, False)
```

Output:



9. Write the python code for the algorithm below:

```
fact100(number)
    if number equals 100
        return 100
    otherwise
        return number * fact100(number + 1)
```

10. Write a simulation for a *dating* game, where a guy takes his girlfriend to the movies every week. Each week they flip a coin. If he wins, he picks the movie; otherwise she chooses. There is a certain percent chance that she'll pick a chick flick. (So there are two probabilities: 50% that she'll get to pick, and then another chance that she'll pick a chick flick.) They'll go out for a year (so 52 dates). As soon as she picks more than 10 chick flicks, he'll break up with her, and if they don't break up they'll get married. You are to simulate a version of the game. Your code should explain to the user what is going on; ask for the probability of the woman picking a chick flick; simulate the game; and report whether they broke up or got married. *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