Name:
Email:
Signature:
Circle course section: MW 11-1 TTh 1-3 TTh 4-6 TTh 6-8

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 1, Spring 2012

1. What will the following code print:

```
hiddenMission = "AXpXoXlXlXoX X1"
s = hiddenMission.split("X")
print(s[0])
year = 1967
months = ["january","february","march"]
m1 = months[1].capitalize()
m2 = months[2].capitalize()
print("Take-off: ", m1,21,year)
print("Landing: ", m2,7,year)
print("Due to fire, {0}'s mission aborted".format("".join(s)))
```

2. Write a **function** that takes as a parameter a list of strings and returns a list that contains the last letter of each string.

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
def main():
    win = GraphWin("What's displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(5,1)
    for i in range(5):
        p1.move(0,1)
        Circle(p1,1).draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following code print:

```
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(s)*2+1
    c = cities[len(s)%4]
    return(num,c)
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we've had a problem here")
def prob4_1():
    n,m = calculate("Vers 1")
    mystery(n,m)
prob4_1()
```

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

```
def main():
    welcome()             #Prints "Welcome" to the screen
    age = userInput()     #Continues to prompt until user enters a positive
                          #number and returns that number
    y = calculate(age)    #Using age, calculates year born
    displayResults(age,y) #Prints age and birth year
main()
```

(That is, write the functions `welcome()`, `userInput()`, `calculate()` and `displayResults()`.)

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(4,2,-1):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes out only the largest number from the file. You may assume that every line of the input file consists of exactly one number.

8. What will the following code print:

```
nums = [1,4,0,6,5,2,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

9. Write a **function** that takes number between 1 and 12 as a parameter and prints out the corresponding month as a string. For example, if the parameter is `1`, your function should print out `January`. If the parameter is `2`, your function should print out `February`, etc.

10. Write a simulation for the game *Jenga*, where players take turns removing blocks from a large tower. A player loses if the tower collapses after he removes a block. 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 tower collapsing after a given move and the number of blocks in the initial tower; 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()`

Name:
Email:
Signature:
Circle course section: MW 11-1 TTh 1-3 TTh 4-6 TTh 6-8

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Lehman College, CUNY
CMP 230 Final Exam, Version 2, Spring 2012

1. What will the following code print:

```
hiddenMission = "AZpZoZlZlZoZ Z11"
s = hiddenMission.split("Z")
print(s[7])
year = 1969
days = [16,20,24]
print("Take-off: July {0}, {2}\nLanding: July {1}, {2}".format(days[0], days[2], year))
cmd = s[0] + "rmstrong"
print("Commander: ", cmd, "".join(s))
```

2. Write a function that takes as a parameter a list of strings and returns a list containing each string twice.

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
def main():
    win = GraphWin("What's displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(1,5)
    for i in range(5):
        p1.move(1,0)
        Circle(p1,1).draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following code print:

```
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we've had a problem here")
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(s)+2
    c = cities[int(s[-1])]
    return(num,c)
def prob4_2():
    n,m = calculate("Version 2")
    mystery(n,m)
prob4_2()
```

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

```
def main():
    welcome()            #Prints "Welcome" to the screen
    y = userInput()      #Continues to prompt until user enters a positive
                         #number and returns that number
    age = calculate(y)   #Using year born, calculates age
    displayResults(age,y) #Prints age and birth year
main()
```

(That is, write the functions `welcome()`, `userInput()`, `calculate()` and `displayResults()`.)

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(0,4,2):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes only the smallest number from the file. You may assume that every line of the input file consists of exactly one word.

8. What will the following code print:

```
nums = [10,4,1,6,5,2,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] > nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

9. Write a **function** that takes number between 1 and 7 as a parameter and prints out the corresponding day as a string. For example, if the parameter is `1`, your function should print out `Monday`. If the parameter is `2`, your function should print out `Tuesday`, etc.

10. Write a simulation for the game *Sorry*, where players take turns rolling a 6-sided dice and moving their pieces around the game board. A player wins if they are the first player to have all their pieces back to their starting point. The game board has 15 squares on each side, for a total of 60 squares the pieces must travel. If you land on a square with the other player, his piece returns to his home. You are to simulate a version of the game with two players and one game piece each. Your code should explain to the user what is going on; 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()`

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

Name:
Email:
Signature:
Circle course section: MW 11-1 TTh 1-3 TTh 4-6 TTh 6-8

Lehman College, CUNY
CMP 230 Final Exam, Version 3, Spring 2012

1. What will the following code print:

```
hiddenMission = "A#p#o#l#l#o# #17"
s = hiddenMission.split("#")
print(s[7])
days = [7,11,19]
year = 1972
print("Take-off: Dec {0}, {2}\nLanding: Dec {1}, {2}".format(days[0], days[2],year))
cmd = s[0] + "merica"
print("Command Module: ", cmd, "".join(s))
```

2. Write a function that takes as a parameter a list of strings and returns a list containing each string in uppercase letters.

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
def main():
    win = GraphWin("What's displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(5,5)
    for i in range(5):
        Circle(p1,i).draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following code print:

```
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we've had a problem here")
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(cities)-10
    c = cities[len(s)]
    return(num,c)
def prob4_3():
    n,m = calculate("V3")
    mystery(n,m)
prob4_3()
```

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

```
def main():
    welcome()            #Prints "Welcome" to the screen
    age = userInput()    #Continues to prompt until user enters a positive
                         #number and returns that number
    y = calculate(age)   #Using age, calculates retirement year (year turns 65)
    displayResults(age,y) #Prints age and retirement year
main()
```

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(2,4,1):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes out only the largest number from the file. You may assume that every line of the input file consists of exactly one number.

8. What will the following code print:

```
nums = [2,4,3,6,1,20,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] <= nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

9. Write a **function** that takes number between 1 and 9 as a parameter and prints out the corresponding string. For example, if the parameter is `1`, your function should print out `one`. If the parameter is `2`, your function should print out `two`, etc.

10. Write a simulation for the game *Parcheesi*, where players take turns rolling two 6-sided dice and moving their pieces around the game board. A player wins if they are the first player to have all their pieces back to their starting point. The game board has 72 squares the pieces must travel. If you land on a square with the other player, his piece returns to his home. You are to simulate a version of the game with two players and one game piece each. Your code should explain to the user what is going on; 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()`

ANSWER KEY for CMP 230 Final Exam, Version 1 Spring 2012
Lehman College, CUNY

1. What will the following code print:

```
hiddenMission = "AXpXoXlXlXoX X1"
s = hiddenMission.split("X")
print(s[0])
year = 1967
months = ["january","february","march"]
m1 = months[1].capitalize()
m2 = months[2].capitalize()
print("Take-off: ", m1,21,year)
print("Landing: ", m2,7,year)
print("Due to fire, {0}'s mission aborted".format("".join(s)))
```

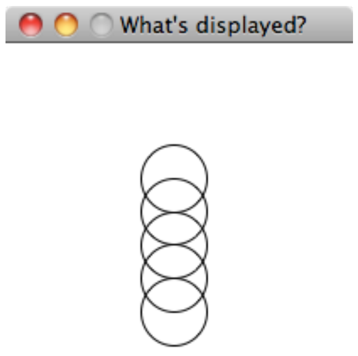
```
A
Take-off:  February 21 1967
Landing:  March 7 1967
Due to fire, Apollo 1's mission aborted
```

2. Write a **function** that takes as a parameter a list of strings and returns a list that contains the last letter of each string.

```
def last(myList):
    s = []
    for l in myList:
        s.append(l[-1])
    return(s)
```

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
def main():
    win = GraphWin("What's displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(5,1)
    for i in range(5):
        p1.move(0,1)
        Circle(p1,1).draw(win)
    win.getMouse()
    win.close()
main()
```



4. What will the following code print:

```
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(s)*2+1
    c = cities[len(s)%4]
    return(num,c)
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we've had a problem here")
def prob4_1():
    n,m = calculate("Vers 1")
    mystery(n,m)
prob4_1()
```

```
Vers 1
Houston we've had a problem here
```

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

```
def main():
    welcome()              #Prints "Welcome" to the screen
    age = userInput()      #Continues to prompt until user enters a positive
                           #number and returns that number
    y = calculate(age)     #Using age, calculates year born
    displayResults(age,y)  #Prints age and birth year
main()
```

(That is, write the functions `welcome()`, `userInput()`, `calculate()` and `displayResults()`.)

```
def welcome():
    print("Welcome")

def userInput():
    a = -1
    while a < 0:
        a = eval(input('Enter age: '))
    return a

def calculate(a):
    return(2012-a)

def displayResults(a,y):
    print('Age: {0}, Year: {1}'.format(a,y))

def main():
    welcome()             #Prints "Welcome" to the screen
    age = userInput()     #Continues to prompt until user enters a positive
                          #number and returns that number
    y = calculate(age)    #Using age, calculates year born
    displayResults(age,y) #Prints age and birth year
main()
```

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(5,3,-1):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

```
constellation

skylab
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes out only the largest number from the file. You may assume that every line of the input file consists of exactly one number.

```
def main():
    infile = open("infile.txt", "r")
    max = eval(infile.readline())
    for l in infile.readlines():
        if eval(l) > max:
            max = eval(l)
    print(max)
```

8. What will the following code print:

```
nums = [1,4,0,6,5,2,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

```
[4, 1, 6, 5, 2, 9, 8, 12, 0]
```

9. Write a **function** that takes number between 1 and 12 as a parameter and prints out the corresponding month as a string. For example, if the parameter is `1`, your function should print out `January`. If the parameter is `2`, your function should print out `February`, etc.

```
def numMonth(n):
    if n == 1:
        print("January")
    elif n == 2:
        print("February")
    elif n == 3:
        print("March")
    elif n == 4:
        print("April")
    elif n == 5:
        print("May")
    elif n == 6:
        print("June")
    elif n == 7:
        print("July")
    elif n == 8:
        print("August")
    elif n == 9:
        print("September")
    elif n == 10:
        print("October")
    elif n == 11:
        print("November")
    elif n == 12:
        print("December")
```

or

```
def numMonth(n):
    months = ["January","February","March","April","May","June","July","August","September","Octobe
    print(months[n-1])
```

10. Write a simulation for the game *Jenga*, where players take turns removing blocks from a large tower. A player loses if the tower collapses after he removes a block. 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 tower collapsing after a given move and the number of blocks in the initial tower; 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.

a) Design of top level:

```
def main():
    introduction()
    pr, blocks = getInputs()
    winner = simGame(pr,blocks)
    print("The winner was player", winner)
```

b) Many ways to do this, here’s one possibility of a refinement with two more levels:

```
def simGame(pr,blocks):
    #Keep track of who turn it is and the winner
    turn = 1
    winner = 0    #(if 0, no winner yet)
    #while there are still blocks and no winner:
    while turn < blocks and winner == 0:
        winner = simOneTurn(pr, turn)
        turn = turn+1
    return winner

def simOneTurn(pr, turn):
    #If the blocks fall, then check whose turn it was:
    if random() < pr:
        if turn % 2 == 0:
            return(2)    #Player 2 won!
        else:
            return(1)    #Player 1 won!
    return(0)
```

ANSWER KEY for CMP 230 Final Exam, Version 2 Spring 2012
Lehman College, CUNY

1. What will the following code print:

```
hiddenMission = "AZpZoZlZlZoZ Z11"
s = hiddenMission.split("Z")
print(s[7])
year = 1969
days = [16,20,24]
print("Take-off: July {0}, {2}\nLanding: July {1}, {2}".format(days[0], days[2], year))
cmd = s[0] + "rmstrong"
print("Commander: ", cmd, "".join(s))
```

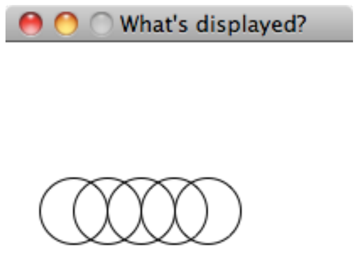
```
11
Take-off: July 16, 1969
Landing: July 24, 1969
Commander:  Armstrong Apollo 11
```

2. Write a function that takes as a parameter a list of strings and returns a list containing each string twice.

```
def double(myList):
    s = []
    for l in myList:
        s.append(l)
        s.append(l)
    return(s)
```

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
from graphics import *
def main():
    win = GraphWin("What’s displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(1,5)
    for i in range(5):
        p1.move(1,0)
        Circle(p1,1).draw(win)
    win.getMouse()
    win.close()
main()
```



4. What will the following code print:

```
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we've had a problem here")
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(s)+2
    c = cities[int(s[-1])]
    return(num,c)
def prob4_2():
    n,m = calculate("Version 2")
    mystery(n,m)
```

```
Version 2
Houston Tranquility Base here
```

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

```
def main():
    welcome()             #Prints "Welcome" to the screen
    y = userInput()       #Continues to prompt until user enters a positive
                          #number and returns that number
    age = calculate(y)    #Using year born, calculates age
    displayResults(age,y) #Prints age and birth year
main()
```

(That is, write the functions `welcome()`, `userInput()`, `calculate()` and `displayResults()`.)

```
def welcome():
    print("Welcome")

def userInput():
    y = -1
    while y < 0:
        y = eval(input('Enter year: '))
    return y

def calculate(y):
    return(2012-y)

def main():
    welcome()             #Prints "Welcome" to the screen
    y = userInput()       #Continues to prompt until user enters a positive
                          #number and returns that number
    age = calculate(y)    #Using year born, calculates age
    displayResults(age,y) #Prints age and birth year
main()
```

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(0,4,2):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

```
mercury

apollo
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes out only the smallest number from the file. You may assume that every line of the input file consists of exactly one number.

```
def main():
    infile = open("infile.txt", "r")
    min = eval(infile.readline())
    for l in infile.readlines():
        if eval(l) < min:
            min = eval(l)
    print(min)
```

8. What will the following code print:

```
nums = [10,4,1,6,5,2,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] > nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

```
[4, 1, 6, 5, 2, 9, 8, 10, 12]
```

9. Write a **function** that takes number between 1 and 7 as a parameter and prints out the corresponding day as a string. For example, if the parameter is `1`, your function should print out `Monday`. If the parameter is `2`, your function should print out `Tuesday`, etc.

```
def numDay(n):
    if n == 1:
        print("Monday")
    elif n == 2:
        print("Tuesday")
    elif n == 3:
        print("Wednesday")
    elif n == 4:
        print("Thursday")
    elif n == 5:
        print("Friday")
    elif n == 6:
        print("Saturday")
    elif n == 7:
        print("Sunday")
```

or

```
def numDay(n):
    days = ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
    print(days[n-1])
```

10. Write a simulation for the game *Sorry*, where players take turns rolling a 6-sided dice and moving their pieces around the game board. A player wins if they are the first player to have all their pieces back to their starting point. The game board has 15 squares on each side, for a total of 60 squares the pieces must travel. If you land on a square with the other player, his piece returns to his home. You are to simulate a version of the game with two players and one game piece each. Your code should explain to the user what is going on; 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.

```
a)  Design of top level:

def main():
    introduction()
    winner = simGame()
    print("The winner was player", winner)

b)  Many ways to do this, here's one possibility of a refinement with two
more levels:
```

```
def simGame():
    #Starting positions:
    pos1 = 0
    pos2 = 0
    #Keep going until someone travels all 60 squares
    while pos1 < 60 or pos2 < 60:
        #Player 1 goes first
        pos1 = simOneTurn(pos1)
        #If first player lands on second, second returns home:
        if pos1 == pos2:
            pos2 = 0
        #Second player’s turn:
        pos2 = simOneTurn(pos2)
        if pos1 == pos2:
            pos1 = 0
    #Return the winner:
    if pos1 >= 60:
        return 1
    else:
        return 2


def simOneTurn(pos):
    #Roll 2 random dice (lowest possible is 2; highest is 12)
    roll = randInt(11)+2
    return(pos+roll)
```

ANSWER KEY for CMP 230 Final Exam, Version 3 **Spring 2012**
Lehman College, CUNY

1. What will the following code print:

```
hiddenMission = "A#p#o#l#l#o# #17"
s = hiddenMission.split("#")
print(s[7])
days = [7,11,19]
year = 1972
print("Take-off: Dec {0}, {2}\nLanding: Dec {1}, {2}".format(days[0], days[2],year))
cmd = s[0] + "merica"
print("Command Module: ", cmd, "".join(s))
```

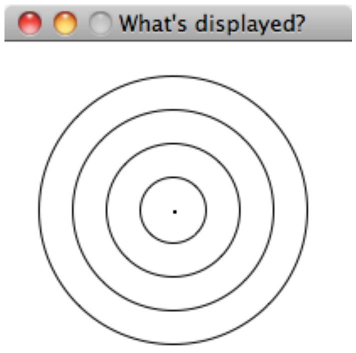
```
17
Take-off: Dec 7, 1972
Landing: Dec 19, 1972
Command Module:  America Apollo 17
```

2. Write a function that takes as a parameter a list of strings and returns a list containing each string in uppercase letters.

```
def upper(myList):
    s = []
    for l in myList:
        s.append(l.upper())
    return(s)
```

3. Draw what is displayed in the graphics window when the following program is executed:

```
from graphics import *
def main():
    win = GraphWin("What's displayed?")
    win.setCoords(0.0,0.0,10.0,10.0)
    p1 = Point(5,5)
    for i in range(5):
        Circle(p1,i).draw(win)
    win.getMouse()
    win.close()
main()
```



4. What will the following code print:

```
def mystery(n,m):
    if n < 0:
        print("But why, some say, the moon? JFK, 1962", m)
    elif n == 11:
        print(m, "Tranquility Base here")
    elif n == 13:
        print(m, "we’ve had a problem here")
def calculate(s):
    print(s)
    cities = ["NYC", "Cape Canaveral", "Houston", "D.C."]
    num = len(cities)-10
    c = cities[len(s)]
    return(num,c)
def prob4_3():
    n,m = calculate("V3")
    mystery(n,m)
prob4_3()
```

```
V3
But why, some say, the moon? JFK, 1962 Houston
```

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

```
def main():
    welcome()             #Prints "Welcome" to the screen
    age = userInput()     #Continues to prompt until user enters a positive
                          #number and returns that number
    y = calculate(age)    #Using age, calculates retirement year (year turns 65)
    displayResults(age,y) #Prints age and retirement year
main()

def welcome():
    print("Welcome")

def userInput():
    a = -1
    while a < 0:
        a = eval(input(’Enter age: ’))
    return a

def calculate(a):
    return(65-a)

def displayResults(a,y):
    print(’Age: {0}, Retire in {1} years’.format(a,y))


def main():
    welcome()             #Prints "Welcome" to the screen
    age = userInput()     #Continues to prompt until user enters a positive
                          #number and returns that number
    y = calculate(age)    #Using age, calculates retirement year (year turns 65)
    displayResults(age,y) #Prints age and retirement year
main()
```

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

```
def main():
    infile = open("in6.txt", "r")
    lines = infile.readlines()
    for i in range(0,4,2):
        print(lines[i])
main()
```

in6.txt

```
mercury
gemini
apollo
skylab
constellation
```

```
apollo

skylab
```

7. Write a **program** that reads in a text file, `infile.txt`, and writes out only the largest number from the file. You may assume that every line of the input file consists of exactly one number.

```
def main():
    infile = open("infile.txt", "r")
    max = eval(infile.readline())
    for l in infile.readlines():
        if eval(l) > max:
            max = eval(l)
    print(max)
```

8. What will the following code print:

```
nums = [2,4,3,6,1,20,9,8,12]
i = 0
while i < len(nums)-1:
    if nums[i] <= nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i + 1
print(nums)
```

```
[4, 3, 6, 2, 20, 9, 8, 12, 1]
```

9. Write a **function** that takes number between 1 and 9 as a parameter and prints out the corresponding string. For example, if the parameter is `1`, your function should print out `one`. If the parameter is `2`, your function should print out `two`, etc.

```
def numString(n):
    if n == 1:
        print("one")
    elif n == 2:
        print("two")
    elif n == 3:
        print("three")
    elif n == 4:
        print("four")
    elif n == 5:
        print("five")
    elif n == 6:
        print("six")
    elif n == 7:
        print("seven")
    elif n == 8:
        print("eight")
    elif n == 9:
        print("nine")
```

or

```
def numString(n):
    nums = ["one","two","three","four","five","six","seven","eight","nine"]
    print(nums[n-1])
```

10. Write a simulation for the game *Parcheesi*, where players take turns rolling two 6-sided dice and moving their pieces around the game board. A player wins if they are the first player to have all their pieces back to their starting point. The game board has 72 squares the pieces must travel. If you land on a square with the other player, his piece returns to his home. You are to simulate a version of the game with two players and one game piece each. Your code should explain to the user what is going on; 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.

```
a)  Design of top level:

def main():
    introduction()
    winner = simGame()
    print("The winner was player", winner)

b)  Many ways to do this, here's one possibility of a refinement with two
more levels:

def simGame():
    #Starting positions:
    pos1 = 0
    pos2 = 0
    #Keep going until someone travels all 72 squares
    while pos1 < 72 or pos2 < 72:
        #Player 1 goes first
        pos1 = simOneTurn(pos1)
        #If first player lands on second, second returns home:
        if pos1 == pos2:
            pos2 = 0
        #Second player's turn:
        pos2 = simOneTurn(pos2)
        if pos1 == pos2:
            pos1 = 0
    #Return the winner:
    if pos1 >= 72:
        return 1
    else:
        return 2


def simOneTurn(pos):
    #Roll 2 random dice (lowest possible is 2; highest is 12)
    roll = randInt(11)+2
    return(pos+roll)
```