1. What will the following code print:

```
islands = "Hawai'i;Kaua'i;Mau'i;O'ahu"
words = islands.split(";")
a = words[0]
b = words[1].lower()
c = words[2].upper()
print(a,b,c)
wo = a[:5] + a[6:]
lava = c[1] + a[1]
print("{0} is found on {1}!".format(lava,wo))
last = b[-1] + c[-1]
print(words[3],last)
```

2. Write a **function** that takes as a parameter a list of numbers and returns a list with each number replaced by the corresponding string representation of that number.

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)
    p2 = Point(5,3)
    Circle(p1, 1).draw(win)
    Circle(p2, 1).draw(win)
    c3 = Circle(Point(5,5), 1)
    l = Line(p1,p2)
    c3.draw(win)
    l.draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following code print:

```
def prob4():
    verse = "jam tomorrow and jam yesterday,"
    print("The rule is,")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

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

```
def prob5():
    infile = open("in.txt", "r")
    x1 = eval(infile.readline())
    x2 = eval(infile.readline())
    x3 = eval(infile.readline())
    if x1 > x2:
        print(x3)
    else:
        print(infile.readline())
    infile.close()
prob5()
```

in.txt

```
40
8
21
30
2011
230
8
1959
10
```

6. Write a **program** that reads in a text file, `infile.txt`, and writes out the negative numbers to another file, `outfile.txt`. You may assume that every line of the input file consists of exactly one number.

7. What will the following code print:

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

8. Write a **function** that takes height as a parameter and prints out a message based on its value. If height is shorter than 4 feet, your function should print out a message saying that the visitor is too short to ride the amusement park rides. If height is taller than 6 feet, your function should print out a message that the user is too tall to ride the amusement park rides. For any other height, your function should print out a message that the visitor can ride the amusement park rides.

9. Write a **program** that asks the user to enter a string. If the user enters an empty string, your program should continue prompting the user for a new string until they enter a non-empty string. Your program should then print out the string entered.

10. Write a simulation to calculate how far a random walker will travel after n steps. At each step, the walker will walk either to the left or right with some probability provided by the user. You are to simulate the walk and report the ending distance from your starting point. Your code should explain to the user what is going on; ask for the probability of walking to the left and the number of steps the walker will make; simulate the walk; and report the results. *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. What will the following code print:

```
islands = "Governors-Randall-City-Roosevelt"
words = islands.split("-")
print(words[3])
a = words[0]
b = words[1].lower()
c = words[2].upper()
print(a,b,c)
who = b[4:]
where = a[4].upper() + c[-1].upper()
print("{0} are found in {1} {2}!".format(who,where,c))
```

2. Write a function that takes as a parameter a list of strings and returns a list with each string in upper case.

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(1,1)
    p2 = Point(3,3)
    p3 = Point(5,5)
    r1 = Rectangle(p1, p3)
    Circle(p2, 2).draw(win)
    l = Line(p1,p3)
    r1.draw(win)
    l.draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following code print:

```
def prob4():
    verse = "Take care of"
    w,c = mystery(verse)
    print("the sense and the sounds will")
    print(w, "themselves")
    enigma(verse,c)
def mystery(v):
    print(v)
    w = v.lower()
    c = 4
    return(w,c)
def enigma(v,c):
    for i in range(c):
        print(":) ")
prob4()
```

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

```
def prob5():
    infile = open("in.txt", "r")
    x1 = eval(infile.readline())
    x2 = eval(infile.readline())
    x3 = eval(infile.readline())
    if x1 > 100:
        print(x2)
    else:
        print(x3)
    infile.close()
prob5()
```

in.txt

```
1966
326
21
30
2011
230
8
1959
10
```

6. Write a **program** that reads in a text file, `infile.txt`, and writes out all the words with less than 10 characters to another file, `outfile.txt`. You may assume that every line of the input file consists of exactly one word.

7. What will the following code print:

```
nums = [1,4,5,10,5,2,1,9,8,12]
pr = 1
i = 0
while pr < 20 and i < len(nums):
    pr = pr*nums[i]
    i = i + 1
print(i, pr)
```

8. Write a **function** that takes height as a parameter and prints out a message based on its value. If height is shorter than 36 inches, your function should print out a message saying that the customer is too short to ride the bus alone. If height is taller than 44 inches, your function should print out a message that the user must pay $2.25 to ride the bus. For any other height, your function should print out a message saying the user may ride the bus for free.

9. Write a **program** that asks the user to enter a positive number. If the user enters a negative number or zero, your program should continue prompting the user for a new number until they enter a positive one. Your program should then print out the positive number entered.

10. Write a simulation to calculate how far a random walker will travel after n steps. At each step, the walker will walk either to the left or right with equal probability. You are to simulate the walk and report the ending distance from your starting point. Your code should explain to the user what is going on; ask for the number of steps the walker will make; simulate the walk; and report the results. 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. What will the following code print:

```
islands = "Anguila%Dominican Republic%Haiti%Jamaica"
words = islands.split("%")
print(words[3])
a = words[0].lower()
b = words[1]
c = words[2].upper()
print(a,b,c)
names = b.split(" ")
d = names[0]
e = names[1]
print(d,e)
hi = c[0] + b[1] + a[-2:]
print("{0} in {1} {2}!".format(hi,d[0],e[0]))
```

2. Write a function that takes as a parameter a list of strings and returns a list containing the first letter of each string in the list.

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)
    p3 = Point(5,5)
    c1 = Circle(Point(5,1), 1)
    c2 = Circle(Point(5,3), 1)
    Circle(p3, 1).draw(win)
    c1.draw(win)
    c2.draw(win)
    win.getMouse()
    win.close()
main()
```

4. What will the following program print:

```
def prob4():
    verse = "Times"
    w,c = mystery(verse)
    print("what I tell you")
    enigma(w,c)
    print("is true.")
def mystery(v):
    print(v)
    w = v.lower()
    c = 3
    return(w,c)
def enigma(v,c):
    for i in range(c):
        print(v)
prob4()
```

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

```
def prob5():
    infile = open("in.txt", "r")
    x1 = eval(infile.readline())
    x2 = eval(infile.readline())
    if x1 > 0:
        print(x2)
    else:
        print(infile.readline())
    infile.close()
prob5()
```

in.txt

```
10
-21
30
2011
-230
8
1959
10
```

6. Write a **program** that reads in a text file, `infile.txt`, and writes out all numbers between -10 and 10 that are in infile.txt to another file, `outfile.txt`. You may assume that every line of the input file consists of exactly one number.

7. What will the following code print:

```
nums = [1,4,-3,6,-10,2,-1,9,8,12]
sum = 0
i = 0
while sum > 0 and i < len(nums):
    sum = sum + nums[i]
    i = i + 1
print(i, sum)
```

8. Write a **function** that takes age as a parameter and prints out a message based on its value. If age is less than 16 years, your function should print out a message saying that the applicant is too young for a drivers licence. If age is greater than 18 years, your function should print out a message that the applicant can apply for a regular drivers license. For any other age, your function should print out a message saying the user qualifies for a junior licence only.

9. Write a **program** that asks the user to enter a number larger than or equal to 100. If the user enters a number smaller than 100, your program should continue prompting the user for a new number until they enter a larger than or equal to 100. Your program should then print out the positive number entered.

10. Write a simulation to calculate the farthest a random walker will travel after n steps. At each step, the walker will walk either to the left or right with equal probability. You are to simulate the walk and report the farthest distance from your starting point. Your code should explain to the user what is going on; ask for the number of steps the walker will make; simulate the walk; and report the results. 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()`