Lab 2: 2.1

```
a = input('Enter the first stick"s length (assumed positive): ')
b = input('Enter the second stick"s length (assumed positive): ')
c = input('Enter the third stick"s length (assumed positive): ')

If a>b+c  or  b>a+c  or c > a+b:

       print 'Cannot be arranged to make a triangle!'
else:
       print 'Can be arranged to make a triangle!'

if a<=b+c  and b <= a+c  and  c<=a+b:

       print 'Can be arranged to make a triangle!'
else:
       print 'Cannot be arranged to make a triangle!'
```

Lab 2: 2.2

```
s = raw_input('Enter a length-3 string: ')

if s[0]==s[1] or s[0]==s[2]  or s[1]==s[2]:

      print 'There is a repeat'
else:
      print 'All different'

if  s[0] != s[1] and s[0] !=s[2] and s[1]!=s[2]:

      print 'All different'
else:
      print 'There is a repeat'
```

Lab 2: 2.3

```
if (1<=x<=2) and (1<=y<=2):
      print 'A'
elif  (x<1) or (x>2):
      print 'B'
elif (y<1) or (y>2):
      print 'C'
else:
      print 'D'
```

X = 1.5, y = 1.5 prints A
X = 0, y = anything prints B
X = 1.5, y = 3 prints C

No choice for x and y prints D. That is because if the if condition is False then one of the elsif conditions have to be true. An instructive typo

Lab 3: 2 (7)

```
X = float(x)
if x==0:
      return 0.0
L = x
L = (L + x/L)/2
etc
```

Lab 3: 3.1

```
# DemoGraphics.py
""" Draws a design with squares and a design
with rings."""

from simpleGraphics import *

# First Figure
MakeWindow(6,bgcolor=DARKGRAY,labels=False)
DrawRect(0,0,6,6,color=CYAN,stroke=5,rotate=0)
# Add more squares...

DrawRect(0,0,6,6,color=ORANGE,stroke=5,rotate=15)
DrawRect(0,0,6,6,color=PURPLE,stroke=5,rotate=30)
DrawRect(0,0,6,6,color=PINK,stroke=5,rotate=45)

# Second Figure
MakeWindow(10,bgcolor=DARKGRAY,labels=False)
# Rings
DrawDisk(0,1,2,stroke=10)
# Add more rings...

DrawDisk(5,1,2,stroke=10)
DrawDisk(-5,1,2,stroke=10)
DrawDisk(2.5,-1,2,stroke=10)
DrawDisk(-2.5,-1,2,stroke=10)

ShowWindow()
```

Lab 3: 3.2

```
def DrawTurkey(x,y,W):
    """ Draws the Turkish Flag.  W is the vertical dimension
    The center of the large red rectangle is at (x,y).

    Precondition: x,y, and W are numbers and W>0.
    """
    W = float(W)
    # Important measurements
    L = 1.5*W
    A = W/2
    B = W/2
    C = W/16
    D = 2*W/5
    E = W/3
    F = W/4
    M = L/30;
    # It is not quite red...
    R = [.89,.039,.09]
    DrawRect(x,y,L,W,color=R,stroke=0)

    # Draw the white rectangle on the edge
    DrawRect(x-L/2-M/2,y,M,W,color=WHITE,stroke=0)

    # Draw the white disk
    DrawDisk(x-L/2+A,y,B/2,color=WHITE,stroke=0)

    # Take a "bite"out of the white disk  by drawing a well placed red disk
    DrawDisk(x-L/2+A+C,y,D/2,color=R,stroke=0)

    # Draw the tilted star
    DrawStar(x-L/2+A+C-D/2+E+F/2,y,F/2,color=WHITE,rotate=18,stroke=0)
```

Lab 4: 4.1

```
# Application Script
if __name__ == '__main__':
    """Find the minimum of dist(t) where t is an
    integer that satisfies L<=t<=R."""

    L = input('Enter initial time (integer): ')
    R = input('Enter final time (integer): ')

    # At any stage of the search, d_min is the smallest value of dis(t)
    # found thus far and t_min is the time associated with that minimum.
    d_max = dist(L)
    t_max = L
    for t in range(L+1,R+1):
        d_current = dist(t)
        if d_current > d_max:
            # A new minimum has been found.
            d_max = d_current
            t_max = t
    print t_max,d_max
```

Lab 4: 4.2

```
# Application Script
if __name__ == '__main__':
    """The number of bad days"""

    L = input('Enter initial time (integer): ')
    R = input('Enter final time (integer): ')

    d_past = dist(L)
    badCount = 0
    for t in range(L+1,R+1):
        d_current = dist(t)
        if d_current > d_past:
            badCount  +=1
        d_past = d_current
    print badCount
```

Lab 4: 5 (First graphic)

```
for k in range(n):
      # Draw the kth row
      if k%2==0 and k>=1:
            DrawRow(x0,y,s,k,c1,c2)
      elif k>=1:
            DrawRow(x0,y,s,k,c2,c1)
      # The next row is up s units
      y = y+s
```

Lab 4: 5 (Second graphic)

```
for k in range(n):
    # Draw the kth row
    if k%2==0:
        DrawRow(x0,y,s,n-k,c1,c2)
    else:
        DrawRow(x0,y,s,n-k,c2,c1)
    # The next row is up s units
    y = y+s
```

Lab 4: 5 (Third graphic)

```
for k in range(n):
    # Draw the kth row
    if k%2==0:
        DrawRow(x0,y,s,n-k,c1,c1)
    else:
        DrawRow(x0,y,s,n-k,c2,c1)
    # The next row is up s units
    y = y+s
```

Lab 4: 6 (First Graphic)

```
# Draw x and y axes
n = 8
MakeWindow(n,bgcolor=PINK,labels=True)
DrawLineSeg(-n,0,2*n,0,linecolor=BLACK)
DrawLineSeg(0,-n,2*n,90,linecolor=BLACK)
```

Lab 4: 6 (Second Graphic)

```
# Draw a pizza with 4 toppings: CYAN, PINK, PURPLE, YELLOW
# Proceed by drawing 360 colored ``spokes''
n = 4
MakeWindow(n,bgcolor=BLACK)
for k in range(0,360):
    if k<=90:
        DrawLineSeg(0,0,n,k,linecolor=CYAN)
    elif k<=180:
        DrawLineSeg(0,0,n,k,linecolor=PINK)
    elif k<=270:
        DrawLineSeg(0,0,n,k,linecolor=PURPLE)
    else:
        DrawLineSeg(0,0,n,k,linecolor=YELLOW)
```

Lab 4: 6 (Third Graphic)

```
# A Rhombus is a quadrilateral with four equal sides.
# In a Rhombus, opposite angles are equal
n = 8
MakeWindow(n,bgcolor=WHITE,labels=True)
L = 7
theta = 55
x0 = -4
y0 = -2
# Here are two of the sides
DrawLineSeg(x0,y0,L,0,linecolor=BLACK)
DrawLineSeg(x0,y0,L,theta,linecolor=BLACK)
# The vertex opposite (x0,y0) is (x1,y1) where
x1 = x0+L+L*math.cos(theta*math.pi/180)
y1 = y0+L*math.sin(theta*math.pi/180)
# Here are the other two sides
DrawLineSeg(x1,y1,L,180,linecolor=BLACK)
DrawLineSeg(x1,y1,L,180+theta,linecolor=BLACK)
```

Lab 5: 2.2 No cut-off Question

```
r =  .5
MakeWindow(n,bgcolor=BLACK,labels=True)
for k in range(500):
    # Throw the k-th paint ball
    x = randu(-n+r,n-r)
    y = randu(-n+r,n-r)
    etc
```

Lab 5: 2.2 Probability Question

```
rc = randi(1,6)
    if rc==1 or rc==2 or rc==3:
        c = MAGENTA
    elif rc==4 or rc==5:
        c = CYAN
    else:
        c = BLUE
    DrawDisk(x,y,r,color=c)
```

Lab 5: 3.1

```
for t in range(L+1,R+1):
        d_current = dist(t)
        if d_current < d_min:
            # A new minimum has been found.
            d_min = d_current
            t_min = t
t = L+1
while t<R+1:
        d_current = dist(t)
        if d_current < d_min:
            # A new minimum has been found.
            d_min = d_current
            t_min = t
        t+=1
```

Lab 5: 3.2 First Part

```
n = input('Enter a positive integer: ')
m = n
steps=0
mMax = m
while m>1:
    if m%2 == 0:
        # m is even
        m = m/2
    else:
        # m is odd
        m = 3*m+1
    steps+=1
    print steps,m
    if m>mMax:
        mMax = m
print mMax
```

Lab 5: 3.2 Second part

```
n = input('Enter a positive integer: ')
m = n
steps=0
mMax = m
while m>1 and steps<100:
    if m%2 == 0:
        # m is even
        m = m/2
    else:
        # m is odd
        m = 3*m+1
    steps+=1
    print steps,m
```

Lab 5: 4

```
# L is even and x is closer to it than R…
B1 = (L%2==0) and (x-L < R-x)
# R is even and x is closer to it than L
B2 = (R%2==1) and (R-x < x-L)
return B1 or B2
```

Lab 6: 2.3

```
while True:
    N = raw_input('Enter a nonnegative integer:  ')
    try:
        # Convert the input string to an int.
        N = int(N)
        # Valid input. Terminate the loop.
        if N>0:
           break
    except ValueError:
        # Invalid input. Inform the user and the
        # iteration continues.
        print 'N must have type  int and N > 0'
```

Lab 6: 4.2.

```
x.sort()
m = n/2
return x[m]


x.sort()
Return x[lenx(x)-10:]
```

Lab 6: 4.3.(a,b)

```
n = 1000000
x = randlist(0,1,N)
A = x[:N-1]
B = x[1:]
C = Add(A,B)
n = n - C.count(1)



A = randlist(1,6,10000)
B = randlist(1,6,10000)
C = randlist(1,6,10000)
D = Add(A,Add(B,C))
Prob = D.count(7)/10000.
```

Lab 7: 2(d)

```
b = []
for s in a:
    B1 = s.count(s[0])==1
    B2 = s.count(s[1])==1
    B3 = s.count(s[2])==1
    if B1 and B2 and B3:
        b.appppend(s)
```

Lab 7: 3(b)

```
for k in range(28):
    dk = sqrt((x[k]-300)**2+(y[k]-200)**2)
    if dk<=400:
        print C[k]
```

Lab 7: 5(a)(b)

Any list a for which a[2] is the smallest value in a[2:]

```
a = [10,20,30,90,100]
```

```
def SelectionSort(a):
    n = len(a)
    b = list(a)
    for k in range(n):
        Select(b,k)
        # b[0:(k+1)] is sorted
    return b
```

Lab 8: 4(a)(b)

```
3**L
```

```
ShowTriPartition([-5,5,5],[-5,-5,5],3)
ShowTriPartition([-5,5,-5],[-5,5,5],3)
```

Lab 9: 2.2

```
count = 0
For k in range(100):
    D= RandomDisk(10)
    ShowDisk(D)
    d = Dist(Point(0,0),D)
    if d<=5:
        count+=1
```

Lab 9: 2.3

```
L = []
for k in range(500):
    D = RandomDisk(10)
    L.append(D)
For D in L:
    C = D.center
    r = D.radius
    if abs(C.x)+r<=10 and abs(C.y)+r<=10:
        ShowDisk(D)
```

Lab 9: 2.4

```
L = []
for k in range(50):
    D = RandomDisk(10)
    L.append(D)
for k in range(50:
    D = L[k]
    if outsideAll(D,L[:k]+L[k+1:]):
        ShowDisk(D)
```

Lab 9: 2.5

```
n = 10
L = []
TotalA = 0
While TotalA < 2*n**2:
    D = RandomDisk(n)
    if outsideAll(D,L):
        L.append(D)
        TotalA += D.area()
        ShowDisk(D)
```

Lab 9: 4(a)(b)(c) -Corrected Versions

```
k=0
for d in D1:
    if d in D2:
        k+=1
```

```
k=0
for d in D1:
    if d not in D2:
        k+=1
For d in D1
    if d not in D1
        k+=1
```

```
k=0
for d in D1:
    if d in D2 and D1[d]==D2[d]
        k+=1
```

Lab 9: 5(a)(b)(c}(d)

(a)All the words that occur 100 or more times
(b) The total number of words in the sonnet colletion
(c)The number of words that occur 100 or more times
(d)Prints all the words that have length 10 or greater

Lab 9: 4(a)(b)(c) -Corrected Versions

```
k=0
for d in D1:
    if d in D2:
        k+=1
```

```
k=0
for d in D1:
    if d not in D2:
        k+=1
For d in D1
    if d not in D1
        k+=1
```

```
k=0
for d in D1:
    if d in D2 and D1[d]==D2[d]
        k+=1
```