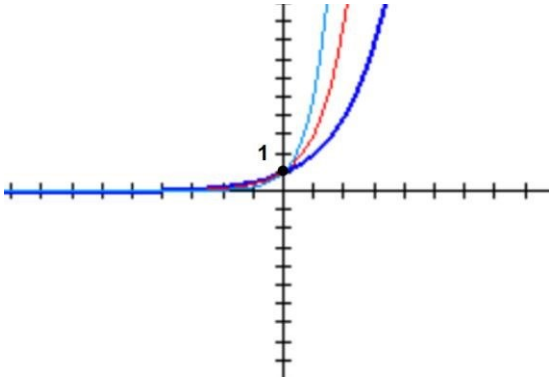
SELECTED SOLUTIONS

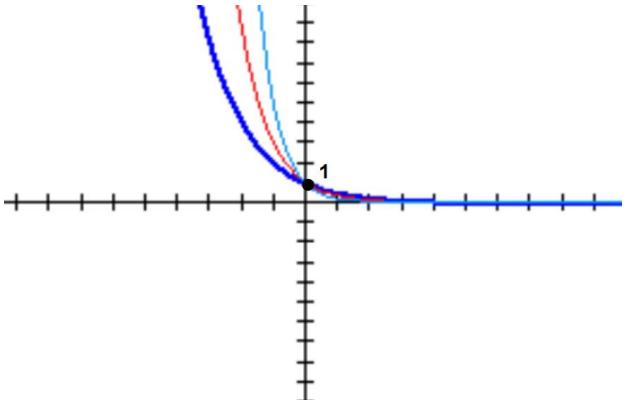
CHAPTER 10

Exercise Set 10.1.1

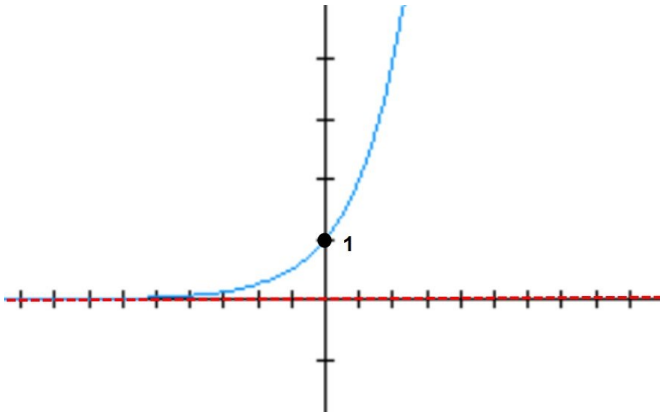
1. The y-intercept of all three functions is (0,1).



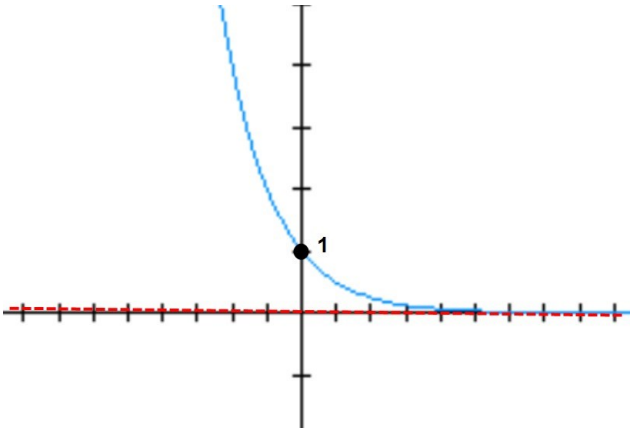
2. The y-intercept of all three functions is (0,1).



3. Domain: $(-\infty,\infty)$; Range: $(0,\infty)$; y-intercept: (0,1); Vertical Asymptote: $y = 0$

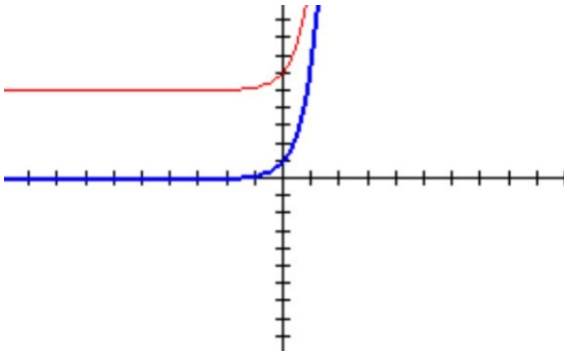


4. Domain: $(-\infty,\infty)$; Range: $(0,\infty)$; y-intercept: (0,1); Vertical Asymptote: y = 0

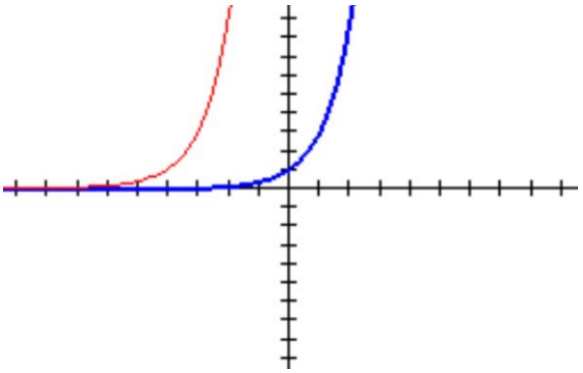


5. If b = 0, then $y = a \bullet b^x$ becomes $y = a \bullet 0^x = 0$ which is the zero function (i.e. the x-axis), not an exponential function.
If b = 1, then $y = a \bullet b^x$ becomes $y = a \bullet 1^x = a$ which is a constant function (i.e. a horizontal line), not an exponential function.

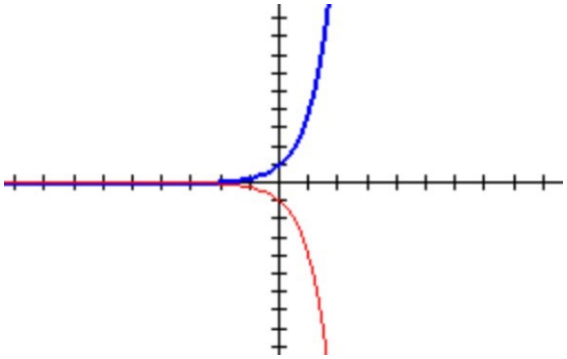
6. $y_1 = 7^x$ and $y_2 = 7^x + 5$



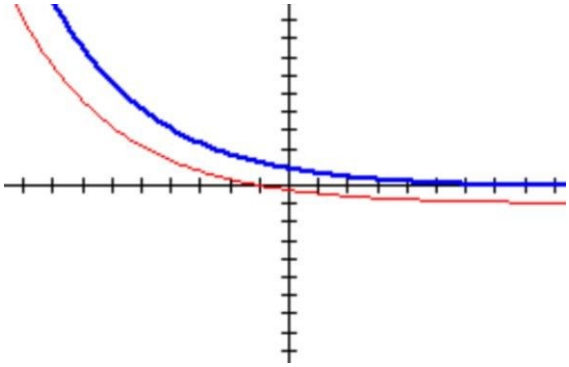
7. $y_1 = 3^x$ and $y_2 = 3^{x+4}$



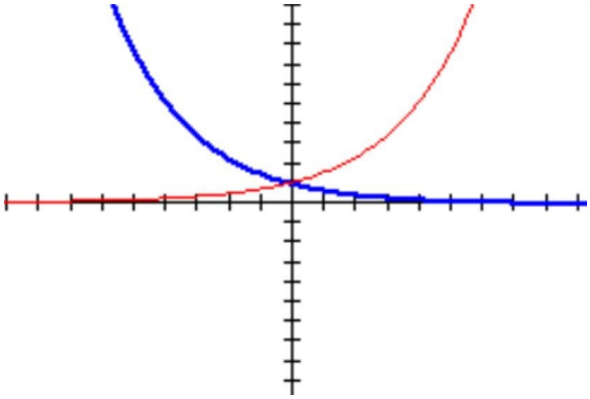
8. $y_1 = 4^x$ and $y_2 = -4^x$



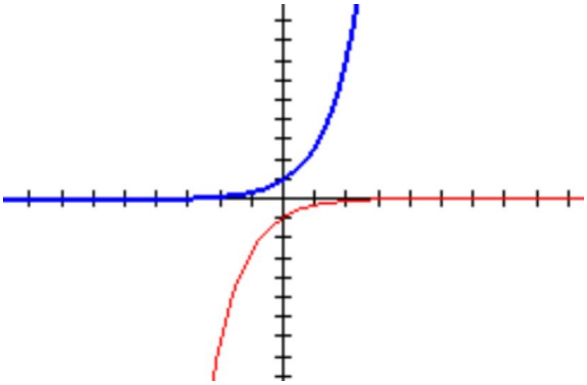
9. $y_1 = \left(\frac{3}{4}\right)^x$ and $y_2 = \left(\frac{3}{4}\right)^{x+1} - 1$



10. $y_1 = \left(\frac{2}{3}\right)^x$ and $y_2 = \left(\frac{2}{3}\right)^{-x}$



11. $y_1 = e^x$ and $y_2 = -e^{-x}$



12. f(-3) = 48

13. h(2) = 3/50

14. $138,403.06

15. Using $f(x) = .01(2)^{x-1}$, $f(30) = .01(2)^{29}$ = $5,368,709.12.

16. Write a program that asks the user to input values for a, b, and x. Calculate and output $a \cdot b^x$.

```
from fractions import Fraction

a = float(input('a? '))
b = float(input('b? '))
x = float(input('x? '))
if a==0:
   print('a cannot equal 0')
elif b==0:
   print('b cannot equal 0')
elif b==1:
  print('b cannot equal 1')
else:
    y = a*b**x
    print(Fraction(y))
```

17.

```
from matplotlib import pyplot as plt
from sympy import Symbol,sympify
x = Symbol('x')
x_list = [i/10 for i in range (-100,100)]

a = float(input('a? '))
b = float(input('b? '))
if a==0:
   print('a cannot equal 0')
elif b==0:
   print('b cannot equal 0')
elif b==1:
  print('b cannot equal 1')
else:
   f = str(a)+'*'+str(b)+'**x'
   f = sympify(f)
   y_list=[f.subs({x:x_value}) for x_value in x_list]
   plt.plot(x_list,y_list)
```

Exercise Set 10.1.2

1. $6^3 = 216$
2. $2^{-7} = \frac{1}{2^7} = \frac{1}{128}$
3. $10^{1/2} = \sqrt{10}$
4. $\frac{y^{21}}{x^{14}}$
5. w
6. $a^{12} + a^{12} = 2a^{12}$
7. $\frac{4}{s^8}$
8. $\frac{2b^4}{5n^8}$

Exercise Set 10.1.3

1. Modify program add_exponents so that if the result is a**1, it just outputs a.

```
m = int(input('m? '))
n = int(input('n? '))
p = m + n
print('(a**', m, ')*(a**', n, ')')
if p==1:
  print('= a')
else:
  print('= a**', p)
```

2. Modify the program so that if the result is a negative exponent, it outputs the result as a fraction, and if the result is a 0 exponent, then the output is simply 1.

```
m = int(input('m? '))
n = int(input('n? '))
p = m + n
print('(a**', m, ')*(a**', n, ')')
if p==1:
  print('= a')
elif p==-1:
   print('=1/a')
elif p<0:
   print('=1/a**',abs(p))
elif p==0:
   print('= 1')
else:
  print('= a**', p)
```

3. Modify the program so that the output is formatted using SymPy's **pprint()** function.

```
from sympy import sympify,pprint

m = int(input('m? '))
n = int(input('n? '))
p = m + n
print('(a**', m, ')*(a**', n, ')=')
if p==1:
   expr = 'a'
elif p==-1:
   expr = '1/a'
elif p<0:
  expr = '1/a**'+str(abs(p))
```

```
elif p==0:
   expr='1'
else:
  expr = 'a**'+str(p)

expr = sympify(expr)
pprint(expr)
```

Exercise Set 10.1.4

1. a**(m + n)
2. a**(m - n)
3. (a*b)**n
4. a**3*b**3
5. a**(m+n)
6. a**m*a**n
7. 1/a
8. a**5
9. a**9.0
10. a**m/a**n

Exercise Set 10.2.1

1. x = 2
2. x = 4
3. x = 15
4. x = ½
5. x = 3
6. x = -3, x = 1
7. x = 2

Exercise Set 10.2.2

1. A. left = '2**x', right = '64'

   B. exp_eqn2 = '4**(x+1) – (64)'

2. The real solution is 3. The complex solution is (log(8) + I*pi)/log(2).

3. The result is that only real solutions are output, not complex solutions.

   4. Yes the program handles square root expressions using the function sqrt().

Exercise Set 10.3.1

1. 11.4 lbs per square inch

2. $16,231

3. 3.35 mg

4. 362 students

5. P(1) = 5 prime
   P(2) = 17 prime
   P(3) = 257 prime
   P(4) = 65537 prime
   P(5) = 42994967297 not prime

Exercise Set 10.3.2

1. The y-intercept is $(0, P_0)$

2. 0

3. $Y_{max} = P_0 e^{25k}$

4. $Y_{max} = P_0$

Exercise Set 10.3.3

1. $P_0 = 100$, $k = 0.045$

A. About 125.2 g

B. About 127.5 g

C. About 250.5 g

2.A. $P_0 = 500$, k = .02

B. About 611

C. About 916

D. About 901

3A. 1.67% of the original amount.

B. About 5275 g

4. Write a program that asks the user to input the initial population, time t, and growth rate k. Calculate and output P(t) rounded to the nearest integer.

```
import math
p0 = int(input('initial population? '))
r = float(input('growth rate? '))
t = int(input('time? '))
pt = p0*math.exp(r*t)
pt = int(pt)
print('P(',t,')=',pt)
```

5. Write a program that asks the user to input the initial population P_0, half-life h, and amount of time t. Calculate and output P(t) rounded to the nearest integer.

```
p0 = int(input('initial population? '))
h = float(input('half life? '))
t = int(input('time? '))
pt = p0*(1/2)**(t/h)
pt = int(pt)
print('P(',t,')=',pt)
```

6. Write a program that asks that user to input the annual interest rate *r* and amount of initial investment *A*. Calculate the doubling time and the amount.

```
r = float(input('annual interest rate? '))
p0 = int(input('amount invested? '))
t = round(72/r,2)
A = round(2*p0,2)
print('You will have', A, 'in about', t, 'years')
```

Exercise Set 10.4.1

1. A. undefined because g(0) is undefined

B. undefined because f(-3) is undefined

C. 8

D. 2/5

E. undefined because f(-3) is undefined

F. $\frac{2}{2+3x}$, x ≠ 0, x≠ -2/3

G. $\frac{2(x+3)}{x}$, x ≠ 0, x≠ -2/3

2. A. undefined because g(-1) is not a real number.

B. 1

C. undefined because f(0) undefined

D. 1/8

E. undefined because f(0) undefined

F. $\frac{1}{(\sqrt{x})^3}$ *or* $\frac{1}{x\sqrt{x}}$

G. $\sqrt{\frac{1}{x^3}}$

H. x ≠ -1, x ≠ 0

I. x > 0 or (0, ∞)

3. A. undefined because g(-1) not a real number

B. 1

C. 0

D. 4

E. |x|

F. x

G. |x|

H. x ≥ 0 or [0, ∞)

I. (-∞, ∞)

4. Possible answers:

A. f(x) = $\frac{5+x}{4-x}$, g(x) = x^2

B. $f(x) = 2 + \sqrt{x}$, $g(x) = x^3 - 4$

C. $f(x) = e^x$, $g(x) = \sqrt{x}$

Exercise Set 10.4.2

1. Fill in yellow highlighted code to calculate and output f(g(x)).

```
result = 'undefined'                # default value
print ("g(f(",x1,"))=")
if f(x1)!=float('inf'):
    if g(f(x1))!=float('inf'):
        result =
Fraction(g(f(x1))).limit_denominator(100)
print(result)
```

2. Modify the program so that the user can input a list of values.

```
def f(a):
if a!=-3:
      return a/(a+3)
else:
       return float('inf')

def g(a):
    if a!=0:
        return 2/a
    else:
        return float('inf')

# main program
x_str = input("x list? ")
x_str_list=x_str.split(',')
# convert from string to float
x_list=[float(x_value) for x_value in x_str_list]
fg = []
gf=[]

for x1 in x_list:
    result = 'undefined'                # default value
    if g(x1)!=float('inf'):
        if f(g(x1))!=float('inf'):
            result = round(f(g(x1)),2)
   fg.append(result)
```

```
    result = 'undefined'                # default value
    if f(x1)!=float('inf'):
        if g(f(x1))!=float('inf'):
            result = round(g(f(x1)),2)
    gf.append(result)

print("f(g(x))=",fg) print("g(f(x))=",gf)
```

3. Replace functions f(a) and g(a) with the following:

```
def f(a):
if a != 0:
        return 1/a**3
else:
       return float('inf')

def g(a):

    from math import sqrt
if a >= 0:
       return sqrt(a)
else:

       return float('inf')
```

Exercise Set 10.4.3

1. Fill in yellow highlighted code to input function g and lambdify it.

```
g_str=input("function g? ")
g = lambdify(x,g_str)
```

2. Fill in the aqua highlighted code to calculate and output g(f(x1)).

```
print("g(f(", x1, "))=")
result = Fraction(g(f(x1))).limit_denominator(100)
print(result)
```

3. A. f(g(1)) = 1/10, g(f(1)) = -4
   B. f(g(5)) = -96, g(f(5)) = 0
   C. f(g(-2)) = 3/20, g(f(-2)) = 17/15

Exercise Set 10.4.4

1. Incorporate exception handling.

```
from fractions import Fraction
from sympy import Symbol,lambdify

# main program
x = Symbol('x')
f_str=input("function                f?                ")
f = lambdify(x, f_str)

g_str=input("function g? ")
g = lambdify(x,g_str)

x1 = Fraction(input("x? "))

print("f(g(",x1,"))=")
try:

result=Fraction(f(g(x1))).limit_denominator(100)

except:
    result = "Undefined"
print(result)


print("g(f(",x1,"))=")
try:
     result=Fraction(g(f(x1))).limit_denominator(100)
except:
    result="Undefined"
print(result)
```

2. Modify program so that user can input a list of values.

```
from sympy import Symbol,lambdify

x = Symbol('x')
f_str=input("function f? ")
f=lambdify(x,f_str)
g_str=input("function g? ")
g = lambdify(x,g_str)

x_str = input("x list? ")
x_str_list=x_str.split(',')
x_list=[float(x_value) for x_value in x_str_list]
fg = [ ]
gf=[ ]
for x1 in x_list:     # find f(g(x))
try:
    result = round(f(g(x1)),2)
except:
     result="Undefined"
fg.append(result)

# find g(f(x))
try:
    result = round(g(f(x1)),2)
except:
    result="Undefined"
    gf.append(result)

print("f(g(x))=",fg) print("g(f(x))=",gf)
```

3. You will end up with either an error or complex number because you are trying to raise a negative number to a fractional exponent.

Exercise Set 10.5.1

1. F, O
2. F
3. F, O
4. F, O
5. F
6. F

7.F

8. F, O

9. R

10. R

11. F, O

12. F, O

13. R

14. F

15. An even function cannot be one-to-one by its very definition: f(-x) = f(x) for any x.

Exercise Set 10.5.2

1. Replace the yellow highlighted code with the following to check if the function is one-to-one.

```
one2one = True
for y in y_values:
    if y_values.count(y) > 1:
        one2one = False

    if one2one==False:
        print("It is not one-to-one")
    else:
        print("It is one-to-one")
```

2. Modify the program so that it outputs the set of ordered pairs. Insert the following after the first **else:**

```
pairs = []
for k in range(len(x_values)):
   pairs.append((x_values[k],y_values[k]))
print(pairs)
```

Exercise Set 10.6.1

1. $f^{-1}(x) = ½ x$

2. $f^{-1}(x) = \frac{x-1}{3}$

3. $f^{-1}(x) = \frac{1}{x}$

4. $f^{-1}(x) = 3x + 2$

5. $f^{-1}(11) = 3$

6. $f(17) = 24$

7. $f^{-1}(x) = \frac{x+2}{5}$ so $f^{-1}(8) = 2$

8. $f^{-1}(x) = \frac{\sqrt{x}}{2}$

so $f^{-1}(64) = 4$

9. 4

10. 0

11. -3

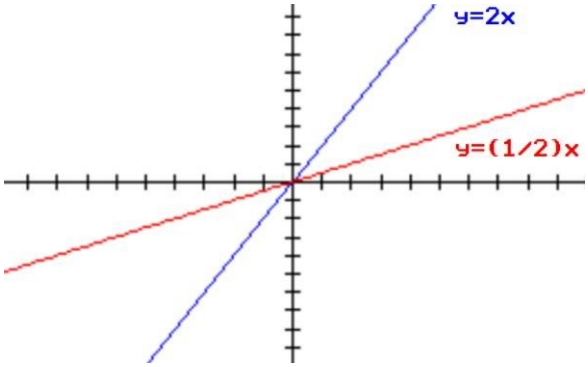
12. { (-2, -4), (0, -3), (1, -1), (3, 0), (4,4) }

13. Write a program that calculates f(a) and the value of f^{-1} at that value.

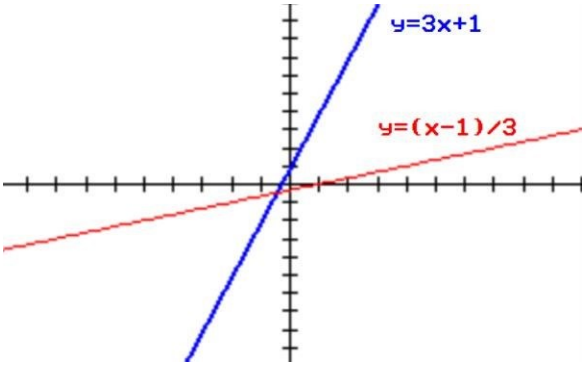
```
from sympy import *
x=  Symbol('x')
fcn_str = input("Function? ")
a = int(input("a? "))
fcn_x = sympify(fcn_str)
b = fcn_x.subs({x:a})
print('f(', a, ') = ', b)
print('f -1 (', b, ') = ',a)
```

Exercise Set 10.6.2

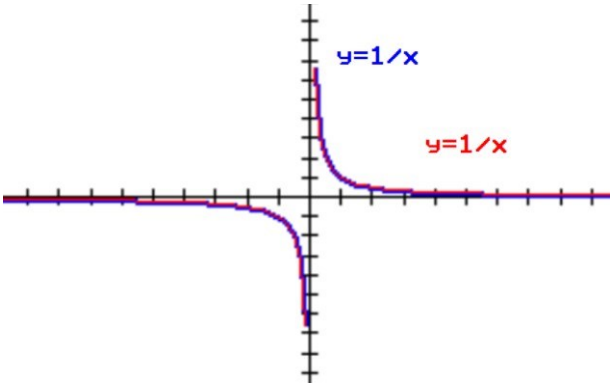
1.



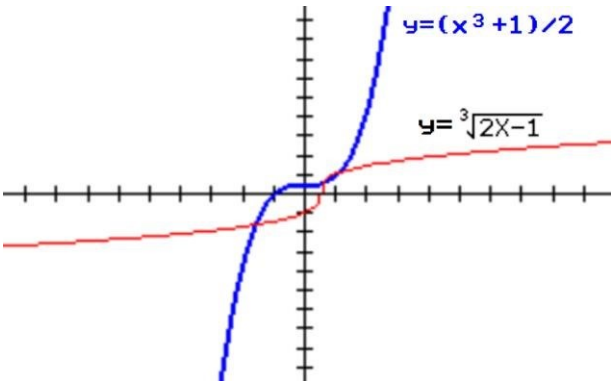
2.



3.



4.



Exercise Set 10.6.3

1. $f^{-1}(x) = \sqrt[3]{x+1}$

2. $f^{-1}(x) = \frac{-4+2x}{x}$

3. $f^{-1}(x) = -\frac{1}{x+3}$

4. $f^{-1}(x) = -\frac{4x+3}{x-2}$

2. $f^{-1}(x) = x^2 + 4$ for $x \geq 0$

Exercise Set 10.6.4

1. Replace aqua code (code to remove x_value from x_list) with:

```
x_list.remove(x_value)
```

Replace pink code (code to append y_value to y_list) with:

```
y_list.append(y_value)
```

2. Replace yellow code (code to draw line of symmetry y=x) with:

```
x_list=range(-20,20)
y_list=[x_value for x_value in x_list]
plt.plot(x_list,y_list,color='red')
```

3. Modify the program so that it draws the graph of the inverse by simply reversing every (*x*, *y*) coordinates in *f*.

```
from sympy import *
from matplotlib import pyplot as plt
from numpy import arange

x=Symbol('x')

fun=input("Function? ")

# initialize subplot and set the axes
ax = plt.subplot()
plt.axis([-20,20, -20,20])

f_exp = sympify(fun)                # convert to SymPy expression

#x_list1 and y_list1 are the domain and range of the function fun
y_list = [ ]
x_list = [ ]
```

```
# initialize x_list1 as the interval [−20,20] with increment 0.2
x_list = list(arange(-20,20,0.2))

for x_value in x_list:
   y_value = f_exp.subs({x:x_value})
   if not y_value.is_real
      # remove x_value from x_list if y_value is not real
      x_list.remove(x_value)
   else:
      # add y_value to y_list if y_value is real
      y_list.append(y_value)

plot_label='$y='+str(fun)+'$'
plt.plot(x_list,y_list,label=plot_label)

# draw the inverse by swapping the x- and y-values
plt.plot(y_list,x_list)
ax.legend()

#Fill in code to draw line of symmetry y = x
x_list=range(-20,20)
y_list=[x_value for x_value in x_list]
plt.plot(x_list,y_list,color='red')
```

Exercise Set 10.6.5

1. Insert the following code to solve and print:

```
inverse = solve(expr,y)
print("inverse fcn is g(x) = ")
pprint(inverse)
```

2. The inverse functions returned are:

   A. $x^2 - 4$
   B. $(x - 4)^2$
   C. $[-\sqrt{x+3}, \sqrt{x+3}]$
   D. log(x) + 8

3. The variable type is a list.

4. Add the following lines to the beginning of the program:

```
from matplotlib import pyplot as plt
from numpy import arange
import draw_rational as dr
```

Add the following lines to the end of the program:

```
plt.clf() ax = plt.subplot()
plt.axis([-10,10,-10,10])

function=[ ]
functions.append(fcn_x)
functions.append(inverse[0])
if len(inverse)==2:
    functions.append(inverse[1])

# Graph each function in functions[]

for f in functions:
    f_exp = sympify(f)
    y_list = [ ]
    x_list = [ ]
    x_list.extend(arrange(-10, 10, 0.2))
    x_list2 = [k for k in x_list]
    for x_value in x_list2:
        y_value =f_exp.subs({x:x_value})
        if not y_value.is_real:
            x_list.remove(x_value)
        else:
            y_list.append(y_value)
    plot_label='$y='+str(f)+'$'
    plt.plot(x_list,y_list,label=plot_label)

ax.legend()

# line of symmetry
x_list=range(-10,10)
y_list=[x_value for x_value in x_list]
plt.plot(x_list,y_list,color='red')
```

Exercise Set 10.6.6

1. Domain of f: $(-\infty,\infty)$; Range of f: $(-\infty,\infty)$

   Domain of f^{-1}: $(-\infty,\infty)$; Range of f^{-1}: $(-\infty,\infty)$

2. Domain of f: $x \neq 2$; Range of f: $x \neq 0$

   Domain of f^{-1}: $x \neq 0$; Range of f^{-1}: $x \neq 2$

3. Domain of f: $x \neq 0$; Range of f^{-1}: $x \neq -3$

   Domain of f^{-1}: $x \neq -3$; Range of f^{-1}: $x \neq 0$

4. Domain of f: $x \neq -4$; Range of f^{-1}: $x \neq 2$

Domain of f^{-1}: $x \neq 2$; Range of f^{-1}: $x \neq -4$

Exercise Set 10.6.7

1. $f(f^{-1}(x)) = f\left(\frac{1}{3}(x-4)\right) = 3\left(\frac{1}{3}(x-4)\right) + 4) = (x-4)+4 = x$

   $f^{-1}(f(x)) = f^{-1}(3x+4) = \frac{1}{3}(3x+4-4) = \frac{1}{3}(3x) = x$

2. $f(f^{-1}(x)) = f\left(\sqrt[3]{x+8}\right) = \left(\sqrt[3]{x+8}\right)^3 - 8 = x+8-8 = x$

   $f^{-1}(f(x)) = f^{-1}(x^3-8) = \sqrt[3]{x^3-8+8} = \sqrt[3]{x^3} = x$

3. $f(f^{-1}(x)) = f\left(\sqrt{x}+2\right) = \left(\sqrt{x}+2-2\right)^2 = \left(\sqrt{x}\right)^2 = x$

   $f^{-1}(f(x)) = f^{-1}((x-2)^2) = \sqrt{(x-2)^2}+2 = (x-2)+2 = x$

$\frac{2x+3}{13}$ $\overline{2x+3}$ $\overline{13}$

4. $f(f^{-1}(x)) = f\left(\frac{3x+5}{x}\right) = \frac{\left(\frac{3x+5}{1-2x}\right)-5}{2\left(\frac{3x+5}{1-2x}\right)+3} = \frac{\frac{3x+5}{1-2x}-5\left(\frac{1-2x}{1-2x}\right)}{\frac{6x+10}{1-2x}+3\left(\frac{1-2x}{1-2x}\right)} = \frac{\frac{3x+5-5+10x}{1-2x}}{\frac{6x+10+3-6x}{1-2x}} = \frac{\frac{13x}{1-2x}}{\frac{13}{1-2x}} = \frac{13x}{13} = x$

   $f^{-1}(f(x)) = f^{-1}\left(\frac{x-5}{2x+3}\right) = \frac{3\left(\frac{x-5}{2x+3}\right)+5}{1-2\left(\frac{x-5}{2x+3}\right)} = \frac{\frac{3x-15}{2x+3}+5\left(\frac{2x+3}{2x+3}\right)}{\left(\frac{2x+3}{2x+3}\right)-2\left(\frac{x-5}{2x+3}\right)} = \frac{\frac{3x-15}{2x+3}+\frac{10x+15}{2x+3}}{\frac{2x+3}{2x+3}-\frac{2x-10}{2x+3}} = \frac{\frac{3x-15+10x+15}{2x+3}}{\frac{2x+3-2x+10}{2x+3}} =$

$\frac{13x}{1-2} = 13x = x$

5. Sample answers: f(x) = $\frac{1}{x}$ or f(x) = x