Dartmouth College
Computer Science 1, Spring 2014
Exam 1

Thursday, April 17

Print your name: ____

Circle your recitation section leader's name.

Xiaole (Caleb) An	Eric Chen	Tyler Crowe	Max Deibel
Derek DeWitt	Lotanna Ezenwa	Emily Greene	Nicole Hedley
Naho Kitade	Anna Knowles	Matthew Krantz	Jai Lakhanpal
Joshua Lang	Justine Lee	Christopher Leech	Randy Li
Bridget Melvin	Taylor Neely	Mehdi Oulmakki	Chander Ramesh
Arun Reddy	Benjamin Ribovich	Benjamin Rush	Nicholas Schwartz
Kameko Winborn	Garrett Watumull	Guanyang (Ethan) Yu	

- If you need more space to answer a question than we give you, you may use the additional blank sheet of paper attached to your exam. Make sure that we know where to look for your answer.
- Read each question carefully and make sure that you answer everything asked for. Write legibly so that we can read your solutions. Please do not write anything in red.
- We suggest that for solutions that require you to write Python code, you include comments. They will help your grader understand what you intend, which can help you get partial credit.
- You have until 9:00 pm to complete this exam.
- You do not need to submit your crib sheet.

Question	Value	Score
1	10	
2	20	
3	5	
4	5	
5	10	
6	10	
7	15	
8	5	
9	20	
Total	100	

Question 1 10 points

For each part of the question, answer True or False. Use the space below each question for your answer.

(a) (2 points)
The Python expression `7 % 4` evaluates to `2`.

FALSE: the correct answer is 3

(b) (2 points)
The cs1lib function `request_redraw` tells Python to copy a 'hidden image' into the graphics window that we can see.

TRUE

(c) (2 points)
In order to alter the value of a global variable `x` in a function, the function must contain the line of code `global x`.

TRUE

(d) (2 points)
An `if` statement with one or more `elif` clauses must have an `else` clause.

FALSE

(e) (2 points)
This while loop will never terminate.

```
i = 0
while( i != 11 ):
     i = i + 2
```

TRUE, because the counter increments by 2 (even numbers) and the stopping condition only checks against 11 (odd)

Question 2 20 points

For each part of the question, I will provide some Python code. Tell me what will be printed on the console. If you think that there is an error in the code that would prevent output, briefly describe the error. Use the space to the right of each question for your answer.

(a) (2 points)

```
x = 5
y = 3
y = x + y
x = x
print x
```

5

(b) (2 points)

```
def do_something(x):
     x = x + 1
print 2*do_something(10)
```

ERROR because do_something doesn't return anything and we are multiplying "nothing" by 2

(c) (2 points)

```
print  not("a" > "b") and (False or 3 > -5)
```

TRUE

(d) (2 points)

```
def add_one(x):
    x = x + 1
y = 10
add_one(y)
print(y)
```

10 (if they said ERROR because of the parentheses in print, we gave them credit – this is legal syntax, but they have never seen it.)

(e) (2 points)

```
x = 1
y = 4
z = 8
if( x + y == 3 ):
     print "step 1"
elif( x + y == 5 ):
     print "step 2"
else:
     print "step 3"
if( x == 1 ):
    print "step 4"
```

step 2 `\n` **step 4**

(f) (2 points)

```
s = "vengence"
i = len(s)-1
while( i >= 0 ):
     print s[i]
     i = i - 1
```

ecnegnev (with each character on a separate line)

(g) (2 points)

```
def my_function(p, q, r):
     if( q > 0 ):
          return r
     else:
          return p
print my_function(1, 2, 3) + my_function(3, -2, 1)
```

6

(h) (2 points)

```
def f(y):
     global x
     x = x + y

x = "abc"
f("d")
f("e")
f("f")
print x
```

abcdef

(i) (2 points)

```
x = [10, 9, 8, 7, 6, 5]
y = x
y[0] = 20
x[0] = 10
if( x[0] == y[0] ):
     print "same"
else:
     print "different"
```

same

(j) (2 points)

```
def f(x):
     return x(100)
def g(x):
     return 2*x
print f(g)
```

200. Many people got this wrong... we pass g, as function, to f – this calls g(100) which returns 200, and therefore 200 is returned and printed.

Question 3 5 points

Which of the four blocks of code numbered 1–4 generates exactly the same output as block 0, regardless of the values of the boolean variables `A`, `B`, and `C` (assume that these variables take on the same value in each block of code). Briefly explain your answer.

```
# block 0
if( A or B or C ):
     print True
```

```
# block 1
if( A and B and C ):
     print True

# block 2
if A:
     print True
if B:
     print True
if C:
     print True

# block 3
if A:
     print True
elif B:
     print True
elif C:
     print True
else:
     print True

# block 4
if A:
     print True
elif B:
     print True
elif C:
     print True
```

BLOCK 4 is equivalent to block 1 because only a single "True" is printed when A or B or C is true and nothing is printed otherwise. Block 2 prints "True" three times if A=B=C=True and Block 3 prints "True" if A=B=C=False – therefore neither of these can be correct.

Question 4 5 points

You decide to implement a tic-tac-toe game with 25 squares (a 5 x 5 board). The goal of the game is the same as in 3 x 3 tic-tac-toe, but this time you have to get 5 in a row to win. Your Artificial Intelligence system for playing against the human has some simple rules early in the game. As the game nears its end, you decide to explore the entire space of possible outcomes. In particular, when there are six remaining squares, you build a "tree" with all possible game outcomes and pick the branch that optimizes your chance of winning. At this stage in the game, how many possible outcomes are there? Briefly explain your answer.

6! = 6 x 5 x 4 x 3 x 2 x 1 = 720. If there 6 empty squares on the board then the first can be selected in any of 6 ways, the second in any of 5 ways, etc. Some students noticed that some of these 720 boards will look identical, even though they were reached in different ways. Removing these same boards leaves 20 unique outcomes. This answer was also acceptable.

Question 5 10 points

Consider the Python code below. Describe the output after this code is executed. Briefly explain what the function `mystery` does.

```
def mystery( S ):
    y = 0
    x = S[y]
    for i in range(1,len(S)):
        if( S[i] >= x ):
            x = S[i]
            y = i
    return y

print mystery( [1,9,7,3,9,2,-4,3,1] )
```

OUTPUT: 4 (the index of the last occurrence of the largest number).

DESCRIPTION: this function returns the index of the largest number. If this largest number is not unique, then this function returns the index to the last occurrence.

Question 6 10 points

Consider the Python code below. Briefly describe what is drawn after this code is executed.

```
from cs1lib import *

def mystery():
    clear()
    s = 20
    flip = 1
    for x in range(0,11):
        for y in range(0,11):
            if( flip == 1 ):
                set_fill_color(1,0,0)
                flip = 0
            else:
                set_fill_color(0,0,0)
                flip = 1

            draw_rectangle(x*s, y*s, s, s)
    request_redraw()

start_graphics( mystery, "mystery", 220, 220 )
```

This function draws a checkboard with alternating red/black squares. There are a total of 11x11 squares on a window of size 220 x 200. The window has the name "mystery".

Question 7 15 points

Write a Python function `perfect_square` that takes a single parameter and returns `True` if this parameter is a perfect square and `False` otherwise (e.g., 16 is a perfect square because $4 \times 4 = 16$). If the parameter passed to this function is not an integer and is not greater than or equal to 0, then you should immediately return `False`. There are several ways to write this function, but your solution must use a `while` or `for` loop to determine if the number is a perfect square.

```
# SOLUTION
def perfect_square( x ):
    if( not( type(x)==int and x > 0) ):
        return False # make sure that x is an integer and greater than 0

    for i in range(0,x+1): # have to search up to x for the case when x=0 or 1
        if( i*i == x ):
            return True # found  perfect square

    return False # not a perfect square
```

Question 8 5 points

Rewrite `perfect_square` from the previous problem but this time you should not use a `while` or `for` loop. In addition you should not use an `if` statement. Unlike the previous question, you need not worry about checking if the input is a positive integer. (Hint: in python `x**2` computes the square of `x` and you are going to want to import the `sqrt` function from the `math` module).

```
# SOLUTION
def perfect_square( x ):
   return( int(sqrt(x))**2 == x )
```

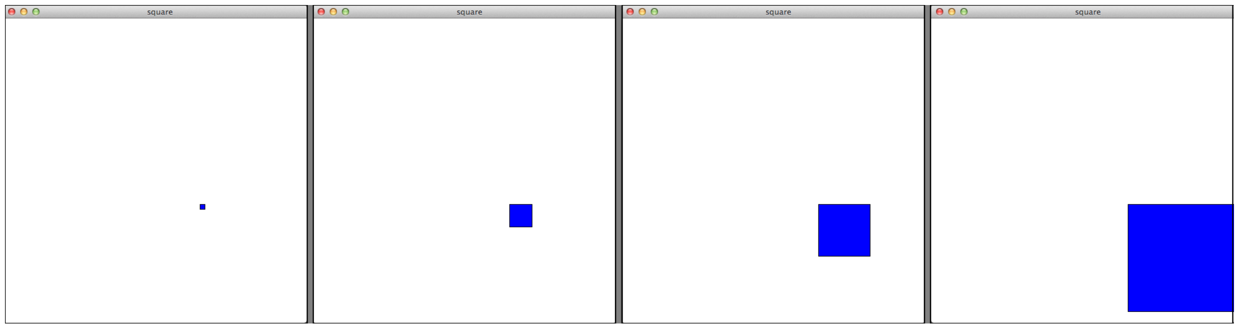
This works because if x is a perfect square, then casting its square root to an int has no impact. Therefore when we square that number, we get x back. There were many creative solutions to this, which were also acceptable.

Question 9 20 points

Write a Python program that creates a 500 x 500 graphics window with the title 'square'. When a user clicks in this window, draw a blue 5 $\times$ 5 pixel square. The square's upper left hand corner should be where the user clicked. Then, repeatedly increase the width and height of the square by 1 pixel until any part of the square reaches the right or bottom of the graphics window. Your square should grow rightward and downward as its size increases (as shown below).

When the square hits any edge of the graphics window the user can click again in the window and the entire process should repeat itself. If the user clicks the mouse while the square is still growing, then you should ignore their mouse click.

Your code should consist of a single function and the appropriate call to `start_graphics`. You may write your solution below and/or on the next blank page.



```
# SOLUTION
from cs1lib import *

def grow_square():
    while not window_closed():
        s = 1
        if mouse_down():
            x = mouse_x() # get mouse position
            y = mouse_y()
            while( x+s <= 500 and y+s <= 500 ): # animate until we reach edge
                clear()
                set_fill_color(0,0,1)
                draw_rectangle( x, y, s, s)
                s = s + 1; # increase square size
                request_redraw()
                sleep(0.02)

start_graphics( grow_square, "square", 500, 500 )
```