**SI@UCF Introduction to Programming in Python Test #2 Solutions**
**6/22/2022**

1) (5 pts) Write a program that reads in the number of seconds a game is played and the number of milliseconds a single frame of the game is shown (so for example, if a frame is shown for 25 milliseconds, in 100 milliseconds, four frames are shown), and calculates the number of frames shown over the course of the game. Have your program print this number out.

```
secondsPlay = int(input("How many seconds will you play?\n"))
oneFrameMS = int(input("How many milliseconds does one frame show?\n"))

# All answers below are worth full credit.
print( (1000*secondsPlay)//oneFrameMS )
print( (1000*secondsPlay)/oneFrameMS )
print( (1000*secondsPlay+oneFrameMS-1)//oneFrameMS)
```

**Grading: 1 pt print, 2 pts mult 1000 by secondsPlay,**
**2 pts divide oneFrameMS**

2) (15 pts) In an art factory, balls of clay are made for sale. The packaging says that the ball of clay must weigh in between 45 grams and 55 grams. Write a program that reads in the current weight of a ball of clay and determines what action (if any) to take to make sure the ball of clay is ready for packaging. Your program must print out a message of one of the three forms: (a) You must add X grams of clay before packaging, (b) You must remove X grams of clay before packaging, or (c) Your clay is ready to be packages. If adding or removing clay, make sure to add or remove the minimal amount of clay necessary to get ready for packaging.

```
claySize = int(input("How many grams is the ball of clay?\n"))

if claySize < 45:
    print("Please add",45-claySize,"grams of clay.")
elif claySize > 55:
    print("Please remove",claySize-55,"grams of clay.")
else:
    print("Clay is ready for packaging!")
```

**Grading: 5 pts total per each option.**

3) (10 pts) Complete the program below so that it reads in a positive integer n from the user and then prints out the first n powers of 2, starting with 2. If the user enters n = 10, your program should print out

```
2 4 8 16 32 64 128 256 512 1024
```

```
n = int(input("Please enter n.\n"))

cur = 2
for i in range(n):
    print(cur)
    cur = 2*cur
```

**Grading: 2 pts loop runs right # times, 2 pts print in loop**
**4 pts update var to print correctly,**
**2 pts rest**

4) (10 pts) Write a segment of code that reads in integers from the user until they enter -1 and stores each of those integers in the list values. Do NOT add -1 to the list.

```
values = []

val = int(input())

while val != -1:
    values.append(val)
    val = int(input())
```

**Grading: 2 pts read val, 3 pts loop, 3 pts append, 2 pts read**

5) (16 pts) Consider writing a pygame program where a sprite starts in the middle of the screen. Whenever the user presses the down arrow key (K_DOWN), the sprite moves to a random location that is below the current location of the sprite (with a new Y value in between the current one and 400) Whenever the user presses the up arrow key (K_UP), the sprite moves to a random location that is above the current location of the sprite (with a new Y value in between 0 and the current Y value). Similarly, when the user presses the left arrow key (K_LEFT), the sprite should move to a random location to its left (but with the same y value) and when the user presses the right arrow key (K_RIGHT), the sprite should move to a random location to its right.

```
import pygame, sys
from pygame.locals import *
import random

pygame.init()
DISPLAYSURF = pygame.display.set_mode((1000, 600))
sprite = pygame.image.load("sprite.gif")
sprite = pygame.transform.scale(sprite, (64, 64))

width  = 1000
height = 600
posX = 500
posY = 300

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

        if event.type == KEYDOWN:

            if event.key == K_DOWN and posY < 400: # Grading -
                                                   # 2 pts per line
                posY = random.randint(posY+1, 400) # No need for and
                                                   # check, I just did
            if event.key == K_UP and posY > 0:     # that to prevent
                                                   # a random number
                posY = random.randint(0, posY-1)   # error

            if event.key == K_LEFT and posX > 0:

                posX = random.randint(0, posX-1)

            if event.key == K_RIGHT and posX < 800:

                posX = random.randint(posX+1, 800)

    DISPLAYSURF.fill(pygame.Color(0, 0, 0))
    DISPLAYSURF.blit(sprite, (posX, posY))
    pygame.display.update()
```

6) (12 pts) Complete the code below so that it displays a red ball bouncing up and down at a constant rate of 5 pixels per frame. The ball should "hit the bottom of the screen" when the y-value of its center is equal to 590 and it should start coming back down when the y-value of its center is equal to 10. The x-value of the center should always remain at 400.

```
import pygame, sys
from pygame.locals import *

pygame.init()

width  = 800
height = 600

DISPLAYSURF = pygame.display.set_mode((width, height))
clock = pygame.time.Clock()

posX = 400
posY = 10
dY = 5
radius = 10

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    # FILL IN ALL CODE HERE!!!


    posY += dY                                  # Grading: 3 pts

    if posY >= 590 or posY <= 10:               # Grading: 6 pts
        dY = -dY                                # Grading: 3 pts



    DISPLAYSURF.fill(pygame.Color(0, 0, 0))
    pygame.draw.circle(DISPLAYSURF, pygame.Color("red"), (posX,
posY), radius)

    pygame.display.update()
    clock.tick(50)
```

7) (15 pts) Write a function that takes in a list of integers, vals, and returns True if it contains any duplicate values, and False if all the values in the list are unique. Don't worry about the efficiency of your method. As long as you solve the problem correctly, you'll get full credit.

```
def containsDuplicates(vals):

    for i in range(len(vals)):
        for j in range(i+1, len(vals)):
            if vals[i] == vals[j]:
                return True

    return False
```

**Grading: Trying each unique pair (or equivalent) – 5 pts**
**Immediately returning True upon discovery – 5 pts**
**Waiting till end to return False – 5 pts**

8) (15 pts) Write a segment of code that reads in 100,000 words from the user (not necessarily distinct) and the prints out a chart showing how many times each word appeared in the list. The chart should have one line per word, with each line having a word, followed by a space, followed by the number of times that word appeared.

```
words = {}                                  # Grading – 2 pts

for i in range(100000):                     # Grading – 2 pts

    word = input()                          # Grading – 1 pt

    if word in words:                       # Grading – 2 pts
        words[word] = words[word] + 1       # Grading – 2 pts
    else:                                   # Grading – 1 pt
        words[word] = 1                     # Grading – 2 pts

for word in words:                          # Grading – 2 pts
    print(word,words[word])                 # Grading – 1 pt
```

9) (2 pt) Topper's Craft Creamery sells $1 cones on campus on Wednesdays (today). What are you allowed to add to the cone if you pay 50 cents more for each one? **Toppings! (Give to All)**