Name : _____

1. *(2 points)*
Grading TA: _____

- INTEGRITY: By taking this exam, you pledge that this is your work and you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech. Do NOT sign nor take this exam if you do not agree with the honor code.
- DEVICES: If your cell phone, pager, PDA, beeper, iPod, or similar item goes off during the exam, you will lose 10 points on this exam. Turn all such devices off and put them away now. You cannot have them on your desk.
- ACADEMIC MISCONDUCT: Academic misconduct will not be tolerated. You are to uphold the honor and integrity bestowed upon you by the Georgia Institute of Technology.
  - Keep your eyes on your own paper.
  - Do your best to prevent anyone else from seeing your work.
  - Do NOT communicate with anyone other than a proctor for ANY reason in ANY language in ANY manner.
  - Do NOT share ANYTHING during the exam. (This includes no sharing of pencils, paper, erasers).
  - Follow directions given by the proctor(s).
  - Stop all writing when told to stop. Failure to stop writing on this exam when told to do so is academic misconduct.
  - Do not use notes, books, calculators, etc during the exam.
- TIME: Don't get bogged down by any one question. If you get stuck, move on to the next problem and come back once you have completed all of the other problems. This exam has 5 questions on 9 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Georgia Tech community. I have also read and understand the requirements outlined above.

Signature: _____

Question	Points	Score
1. Grading TA Name	2	
2. Multiple Choice	7	
3. Robot Drawing	10	
4. Player Stats	14	
5. Food Tracker	9	
Total:	42	

2. *(7 points)*
For each of the following multiple choice questions, indicate the most correct answer by circling it!

(a) [1 pt] What is stored in aTuple after this code is run:

```
aTuple = (1,2,3,['hello','world'], True)
aTuple[3][1] = False
```

A. Error: You can't index into an int
B. `(1,2,3,['hello',False],True)`
C. `(1,2,3,[False,'world'],True)`
D. `(1,2,3,['hello','world'],False)`
E. Error: You can't change that element in a tuple

(b) [1 pt] Order the following items from earliest (older) to latest (newer):

1. The Vacuum Tube
2. The Integrated Circuit
3. The Transistor
4. The Relay

A. 2,4,3,1 B. 1,4,2,3 C. 4,2,1,3 **D. 4,1,3,2** E. None of these.

(c) [1 pt] In 1954, John Backus, developed what for IBM? A. COBAL **B. FORTRAN** C. Ethernet D. Integrated Circuit E. Solid Sate Memory

(d) [1 pt] Which of the following is true about the keys in a dictionary?

A. An integer can be a key.

B. A list can be a key.

C. A dictionary can be a key.

D. All keys must be mutable.

E. All of the above are False.

(e) [1 pt] Assume you have an empty dictionary created as follows:

```
aDict = {}
```

Which one of the following statements would **not** raise an exception?

A. `aDict[ ["1"] ]= "one"`

B. `aDict.items()`

C. `aDict[aDict]="two"`

D. `aDict={1;1}`

(f) [1 pt] What is the **last** thing printed by this code?

```
g = [1,2,3,4]
g.append("the")
print(g)
g = g.append("quick brown fox")
print(g)
x = ("a", "b", "c")
x.append("d")
print(x)
```

Select the valid option:

A. `[1, 2, 3, 4, 'the']`

B. `"quick brown fox"`

C. `None`

D. `("a", "b", "c", "d")`

E. `[1,2,3,4,"the", "quick brown fox"]`

(g) [1 pt] Which of the following is valid for the second parameter of open():

A. `"r"`

B. `"a"`

C. leave it blank

D. `"w"`

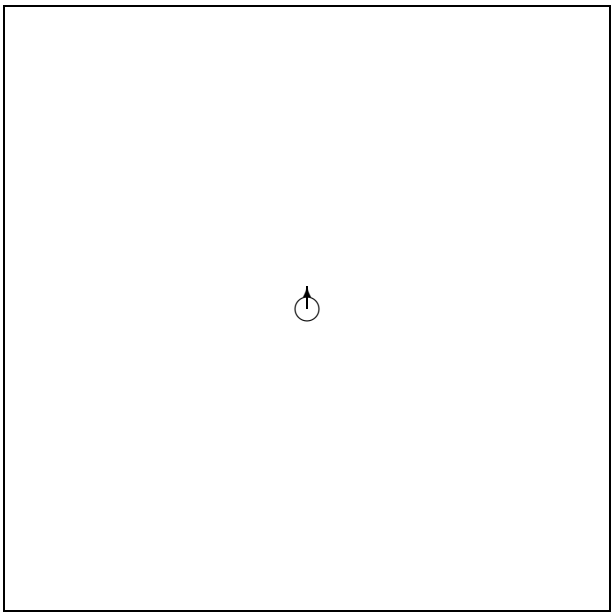
E. all choices are correct

3. *(10 points)*
Draw the robot's trajectory when the following code is executed. Draw the final position of the robot with a circle, and use an arrow to indicate it's final "forward" direction. Assume that the robot starts in the middle facing towards the top of the paper ("up", or "north"). Assume that if you turn for one second the robot turns exactly 90 degrees. If the program prints anything to the shell, write exactly what is printed below the box.

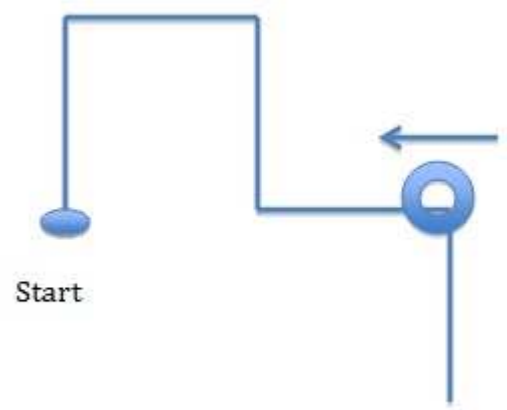
```
from Myro import *
init()

a = 4
b = 2
while a >=0:
   if a%4==0:
      forward(1,1)
   if a%2==0:
      turnLeft(1,1)
   else:
      turnRight(1,5)
   try:
      b = 4 / a
      backward(1,1)
   except:
      print("Exception!" * 2)

   a = a-1
```



Solution:



Grading:
1 point for each correct turn. (4 total)
3 points for having the 5 line segments, or 2 points for six segments.
1 point for the correct robot position.
1 point for the correct facing arrow (left).
1 point for printing Exception!Exception!
-1 point for including quote marks around the printout!

4. *(14 points)*

Write a function called `playerStats` that accepts two parameters. The first parameter will be a string containing a name. The second parameter will be a list of strings. Each string in the list will contain three pieces of information: The name of a player, the number of wins they have, and the number of losses they have, separated by commas.

Your function should find the record for the player named in the first parameter, and calculate their win percentage. The win percentage is calculated as: $\frac{wins}{(wins+losses)}$

Your function should return a tuple that consists of a boolean that represents if the player is a "winner" (has a win percentage equal to or above 0.5) and the win percentage (as a float). You may assume that all wins and loses will be represented by strings (digits) that can be converted into valid integers. Hint: The string.split() method will be useful to separate the three items in each string.

Note: If a record with a matching name is NOT found, your function should return a tuple with None in each position, e.g. `(None, None)`. You may assume that all players in the input list will have played at least one game (i.e. wins and losses will not BOTH be zero, although you may see 0,1 or 1,0 scores).

Example run:

```
>>> recordList = ["Ann, 10, 5", "Mary, 4, 12", "Robert, 1, 6" ]
>>> result = playerStats("Mary", recordList)
>>> print( result )
(False, 0.25)
```

Solution:

```
def playerStats (aName, aList):
   for record in aList:
      recordData = record.split(",")
      if aName == recordData[0]:
         wins = int( recordData[1])
         loss = int( recordData[2])
         winPct = wins / (wins+loss)
         return (winPct >= 0.5 , winPct)

    #If no matching record found:
    return (None, None)
```

Grading:
+1 Correct function header.

+2 correctly iterates through the strings in the input list
+3 Correctly isolates the name, wins, and losses substrings
+1 Matches the inputName to the record's name correctly.
+2 correctly Calculates the win percentage
+2 Correctly compares the win percentage to 0.5 (greater or equal!)
+2 Returns correct data in the tuple.
+1 Returns a (None,None) tuple in the event of a non-match.

5. *(9 points)*

You've decided to keep track of what you eat. You have a pre-existing file named food.txt that contains everything you've eaten and how many calories each item contains. Each line of the file consists of the name of the food, a tab, and the number of calories. Write a function called `foodTracker` that opens food.txt, asks the user (via the input() function) for the name of the food item, and the calorie cost, and then writes the food and calories to the file separated by a tab character. Keep doing this until the user types quit for the name of a piece of food. For example, after this interaction:

```
>>>foodTracker()
Enter a food name: Grapes
How many calories was it? 70
Enter a food name: candy
How many calories was it? 250
Enter a food name: quit
>>>
```

The last two lines of the "food.txt" file would look like this:

```
Grapes          70
candy           250
```

Solution:

```
def foodTracker():
    f = open("food.txt", "a")

    food = input("Enter a food name: ")

    while food != 'quit':
        calorie = input("How many calories was it?")

        f.write("{}\t{}\n".format(food, calorie))

        food = input("Enter a food: ")

    f.close()
```

Grading:
1 point for correct header
2 points for opening file in append mode. (1 point for write mode)

2 points for getting the food/calorie input from the user correctly.
2 points for writing the food/calorie info with a tab between it and a newline after it. (1 pt for correct writes that are missing tab, and/or newline)
2 points for correctly repeating until the user typed 'quit' as a food name.
-1 point if they forgot to close the file.