

Second Homework Exam: Study Guide

1. Write a program that translates a number between 0 and 4 into the closest letter grade. For example the number 2.8 (which might have been the average of several grades) would be converted to B-. Break ties in favor of the better grade; for example 1.85 should be a C. There's more than one way to do this.

```
grade = float(raw_input("Please enter grade: "))

if grade > 4:
    print "Number too big."
elif grade >= 3.85:
    print "A"
elif grade >= 3.5:
    print "A-"
elif grade >= 3.15:
    print "B+"
elif grade >= 2.85:
    print "B"
elif grade >= 2.5:
    print "B-"
elif grade >= 2.15:
    print "C+"
elif grade >= 1.85:
    print "C"
elif grade >= 1.5:
    print "C-"
elif grade >= 1.15:
    print "D+"
elif grade >= 0.85:
    print "D"
elif grade >= 0.35:
    print "D-"
elif grade >= 0:
    print "F"
else:
    print "Number too small."
```

Other ways would be to nest the if statements or to ask context-free questions.

```
grade = float(raw_input("Please enter grade: "))
```

```
if grade > 4:
    print "Number too big."
else:
    if grade >= 3.85:
        print "A"
    else:
        if grade >= 3.5:
            print "A-"
        else:
            if grade >= 3.15:
                print "B+"
            else:
                if grade >= 2.85:
                    print "B"
                else:
                    ...
```

```
if grade > 4:
    print "Number too big."
if 4 >= grade >= 3.85:
    print "A"
if 3.85 > grade >= 3.5:
    print "A-"
if 3.5 > grade >= 3.15:
    print "B+"
if 3.15 > grade >= 2.85:
    print "B"
...
```

2. Write a program that produces scalable patterns. The program asks the user for a size then prints a pattern of that size. Here are two examples: one program prints scalable Z's and the other one prints scalable 4's. Other patterns are possible: A, T, E, L, F, R, Q.

```
size = int(raw_input("Please enter the size: "))

for line in range(size):
    for column in range(size):
        if line + column == size - 1 or \
            line == 0 or line == size-1 or \
            line == size/2 and size/3 <= column <= 2 * size / 3:
            print "*",
        else:
            print " ",
    print
```

```
>>> ===== RESTART =====
>>>
Please enter the size: 13
* * * * *
          *
         *
        *
       *
      * * * *
     *
    *
   *
  *
 *
* * * * *
>>> ===== RESTART =====
>>>
Please enter the size: 18
* * * * * * * * * * * * * * * *
          *
         *
        *
       *
      *
     *
    * * * *
   *
  *
 *
*
* * * * * * * * * * * * * * * *
```

Note how long lines in Python can be split using a backslash for continuation.

Next page has the second program that prints 4's.

3. Write a program that reads two times in military format (0900, 1730) and prints the number of hours and minutes between the two times. Here is a sample run:

```
Please enter the first time: 0900
Please enter the second time: 1730
8 hours 30 minutes
```

Be sure that your program is able to deal with the case where the second time is earlier than the first, in which case the second time must be considered to be part of "the next day":

```
Please enter the first time: 1730
Please enter the second time: 0900
15 hours 30 minutes
```

This has been discussed in class at least twice (once in lab and once in lecture).

```
timeOne = raw_input("Enter first time: ")
timeTwo = raw_input("Enter second time: ")

t1 = int(timeOne[0:2]) * 60 + int(timeOne[2:])
t2 = int(timeTwo[0:2]) * 60 + int(timeTwo[2:])

if (t2 >= t1):
    pass
else:
    t2 = t2 + 24 * 60

d = t2 - t1
print d / 60, "hours and", d % 60, "minutes"
```

```
>>> ===== RESTART =====
>>>
Enter first time: 0900
Enter second time: 1730
8 hours and 30 minutes
>>> ===== RESTART =====
>>>
Enter first time: 1730
Enter second time: 0900
15 hours and 30 minutes
>>> ===== RESTART =====
>>>
Enter first time: 0900
Enter second time: 0859
23 hours and 59 minutes
>>> ===== RESTART =====
>>>
Enter first time: 0859
Enter second time: 1001
1 hours and 2 minutes
>>>
```

Notice how the use of if statements eliminates the need to be extremely clever.

4. Write a program that transforms numbers

1, 2, 3, ..., 12

into the corresponding month names

January, February, March, ..., December

This is, again, straightforward – now that we can use if statements.

```
month = int(raw_input("Enter month number: "))

if month == 1:
    print "January"
if month == 2:
    print "February"
if month == 3:
    print "March"
if month == 4:
    print "April"
if month == 5:
    print "May"
if month == 6:
    print "June"
if month == 7:
    print "July"
if month == 8:
    print "August"
if month == 9:
    print "September"
if month == 10:
    print "October"
if month == 11:
    print "November"
if month == 12:
    print "December"
if month < 1 or month > 12:
    print "Invalid month number."
```

5. Write a program that asks the user to specify the radii and the coordinates of two circles. Check to see if the circles intersect. If they intersect, then display a message "Circles intersect." Otherwise, display "Circles don't intersect." Hint: Compute the distance between the centers and compare it to the sum of the two radii. The distance between two points is the square root of the sum of squared differences between their corresponding coordinates x and y.

```
print "We are going to collect information about the first circle."
x1 = float(raw_input("Please enter the x coordinate of its center: "))
y1 = float(raw_input("Please enter the y coordinated of its center: "))
r1 = float(raw_input("Now enter the radius of this circle: "))

print "Now we're going to ask you about the second circle."
x2 = float(raw_input("Please enter the x coordinate of its center: "))
y2 = float(raw_input("Please enter the y coordinated of its center: "))
r2 = float(raw_input("Now enter the radius of this circle: "))

d = pow((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2), 0.5)

if d <= r1 + r2:
    print "The two circles intersect."
else:
    print "The two circles don't intersect."
```

```
>>> ===== RESTART =====
>>>
We are going to collect information about the first circle.
Please enter the x coordinate of its center: 0
Please enter the y coordinated of its center: 0
Now enter the radius of this circle: 1
Now we're going to ask you about the second circle.
Please enter the x coordinate of its center: 1
Please enter the y coordinated of its center: 1
Now enter the radius of this circle: 0.4
The two circles don't intersect.
>>> ===== RESTART =====
>>>
We are going to collect information about the first circle.
Please enter the x coordinate of its center: 0
Please enter the y coordinated of its center: 0
Now enter the radius of this circle: 1
Now we're going to ask you about the second circle.
Please enter the x coordinate of its center: 1
Please enter the y coordinated of its center: 1
Now enter the radius of this circle: 0.42
The two circles intersect.
>>>
```

Imagine yourself driving a helicopter. Your location is the center of the circle, the radius is the size of your helicopter's rotor blades. Your friend is driving a slightly bigger helicopter. Assuming air currents are irrelevant, how close would the two of you be able to get without your rotor blades hitting? You will immediately agree that if you stay away from each other at a distance bigger than the two blades put together you would both be safe.

6. A year with 366 days is called a leap year. A year is a leap year if it is divisible by 4 (for example, the year 1980), except it is not a leap year if it is divisible by 100 (for example, the year 1900); however, it is a leap year if it is divisible by 400 (for example, the year 2000). There were no exceptions before the introduction of the Gregorian calendar on October 15, 1582 (for example, the year 1500 was a leap year). Write a program that asks the user for a year and computes whether that year is a leap year or not. Try solving this problem by using only one if statement.

```
year = int(raw_input("Enter year: "))

if year < 1582:
    if year % 4 == 0:
        print "Leap year."
    else:
        print "Not a leap year."
else:
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                print "Leap year."
            else:
                print "Not a leap year."
        else:
            print "Leap year."
    else:
        print "Not a leap year."
```

```
>>>
Enter year: 1500
Leap year.
>>>
Enter year: 1980
Leap year.
>>>
Enter year: 2000
Leap year.
>>>
Enter year: 1988
Leap year.
>>>
Enter year: 1991
Not a leap year.
>>>
Enter year: 1900
Not a leap year.
```

The alternative is this (simply state when it should be a leap year – three cases):

```
year = int(raw_input("Enter year: "))

if year < 1582 and year % 4 == 0 or year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):
    print "Leap year."
else:
    print "Not leap year."
```

7. Write a program that starts by choosing a random integer between 0 and 100 and asks you to guess it. Every time it collects your guess it counts it. If your guess is correct it says so and ends, reporting the number of guesses you needed to find out. If your guess is not correct the program tells you if you should try higher or lower with respect to the guess. (Let's call this the "number guessing game").

```
import random

num = random.randrange(100)

print "I have chosen a number between 0 and 100."
guess = int(raw_input("Please try to guess it: "))

count = 1

while guess != num:
    if guess > num:
        print "Try lower."
    else:
        print "Try higher."
    guess = int(raw_input("Please try again: "))
    count += 1

print "Congratulations, you only took", count, "attempts."
```

```
>>>
I have chosen a number between 0 and 100.
Please try to guess it: 50
Try lower.
Please try again: 25
Try higher.
Please try again: 37
Try higher.
Please try again: 42
Try higher.
Please try again: 47
Try lower.
Please try again: 45
Try lower.
Please try again: 44
Try lower.
Please try again: 43
Congratulations, you only took 8 attempts.
>>>
```

As we discussed in class yesterday (Wed, May 20, in lab) the best strategy (on average) is to always halve the range of potential answers. This way your search is balanced. We can write a program that implements our thinking, and that's the purpose of the last problem in this batch.

8. Write a program where the player and the computer trade places in the number guessing game. That is, the player picks a random number between 1 and 100 that the computer has to guess. Before you start, think about how you guess.

```
raw_input("Please choose an integer between 0 and 100, then press Enter.")
print "\nI will guess your number in no more than 8 tries."
print "When asked please answer with \"yes\", \"higher\" or \"lower\" only."
print "Now let's get started.\n"
low = 0
high = 100
guess = (low + high) / 2
answer = raw_input("Is it " + str(guess) + "? ")
while answer != "yes":
    if answer == "higher":
        low = guess
        if low + 1 == high:
            guess = high
        else:
            guess = (high + low) / 2
    else:
        high = guess
        guess = (low + high) / 2
    answer = raw_input("Is it " + str(guess) + "? ")
print "I knew it."
```

This program will be discussed in class today.

It's relatively simple only requires care if the user chooses 100 or 0 (one of the boundary values).

Here's more advanced code that prints for all numbers between 0 and 100 the number of questions asked until that number is guessed. You're only responsible for the code above. But you can run the code below to double-check the behavior of the program above.

```
def determine(secret):
    low = 0
    high = 100
    guess = (low + high) / 2
    count = 1
    while guess != secret:
        if guess < secret:
            low = guess
            if low + 1 == high:
                guess = high
            else:
                guess = (high + low) / 2
        else:
            high = guess
            guess = (low + high) / 2
        count += 1
    return count

for i in range(101):
    print i, " will be found in", determine(i), "questions."
```