

Assignment 3

Due: Dec 16, 2017

Subject: Algorithms

- 1) (Binary Search) Run the following **guess.py** program on your computer a few times:

```
# This is a guess the number game.
import random
guessesTaken = 0
print('Hello! What is your name?')
myName = input()
number = random.randint(1, 10000)
print('Well, ' + myName + ', I am thinking of a number
between 1 and 10000.')
while guessesTaken < 20:
    print('Take a guess.')
    guess = input()
    guess = int(guess)

    guessesTaken = guessesTaken + 1
    if guess < number:
        print('Your guess is too low.')
    if guess > number:
        print('Your guess is too high.')
    if guess == number:
        break

if guess == number:
    guessesTaken = str(guessesTaken)
    print('Good job, ' + myName + '! You guessed my number in
' + guessesTaken + ' guesses!')
if guess != number:
    number = str(number)
    print('Nope. The number I was thinking of was ' + number)
```

What is your best guess? How many times it took for you to guess the number? Send the output of the program with your answers.

What is the optimal number of guesses for this case? How did you calculate?
(Hint: Calculate for the binary search in the range from 1-10000)

What is the best case (the number of times it takes to guess the number right)?

- 2) (ch5.29) What problems do you expect to arise if the following program is implemented in python? (Hint: The problem of round-off errors associated with floating-point arithmetic.). Correct the program and make sure that it halts.

```

cnt = 0.1
repeat:
    print(cnt)
    cnt = cnt + 0.1
until (cnt == 1)

```

- 3) (ch5.2, question 3) The Euclidean algorithm finds the greatest common divisor of two positive integers X and Y by the following process: As long as the value of neither X nor Y is zero, assign the larger the remainder of dividing the larger by the smaller. The greatest common divisor, if it exists, will be the remaining non-zero value. Express this algorithm in our pseudocode. And write a python code to implement.
- 4) (ch5.22) The following algorithm is designed to print the beginning of what is known as the Fibonacci sequence. Write the algorithm in python (version 1).
BONUS: Reimplement the python program **recursively** (version 2) [Hint: $fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)$]. Discuss if the algorithm is efficient (runtime).

```

Last = 0
Current = 1
while (Current < 100):
    print(Current)
    Temp = Last
    Last = Current
    Current = Last + Temp

```

Program should work with the following command:

```

>> print(fibonacci(20))
20.Fibonacci number is 10946

```

- 5) (ch5.social issues.1) Because it is currently impossible to verify completely the accuracy of complex programs, under what circumstances, if any, should the creator of a program be liable for errors?
- 6) (ch5.social issues.5) Some people feel that new algorithms are discovered, whereas others feel that new algorithms are created. To which philosophy do you subscribe? Would the different points of view lead to different conclusions regarding ownership of algorithms and ownership rights? Can algorithms be **patented** (in the USA, in Turkey)?

BONUS Questions (10 points each)

- 7) (ch5.4, question 6) A variation of the insertion sort algorithm is the “**selection sort**”. It begins by selecting the smallest entry in the list and moving it to the front. It then selects the smallest entry from the remaining entries in the list and moves it to the second position in the list. By repeatedly selecting the smallest entry from the remaining portion of the list and moving that entry forward, the sorted version of the list grows from the front of the list, while the back portion of the list consisting of the remaining unsorted entries shrinks. Use our pseudocode to express a function similar to that in Figure 5.11 for sorting a list using the selection sort algorithm and write a python program to implement.

- 8) (*ch5.4, question 7*) Another well-known sorting algorithm is the “**bubble sort**”. It is based on the process of repeatedly comparing two adjacent names and interchanging them if they are not in the correct order relative to each other. Let us suppose that the list in question has n entries. The bubble sort would begin by comparing (and possibly interchanging) the entries in positions n and $n - 1$. Then, it would consider the entries in positions $n - 1$ and $n - 2$, and continue moving forward in the list until the first and second entries in the list had been compared (and possibly interchanged). Observe that this pass through the list will pull the smallest entry to the front of the list. Likewise, another such pass will ensure that the next to the smallest entry will be pulled to the second position in the list. Thus, by making a total of $n - 1$ passes through the list, the entire list will be sorted. (If one watches the algorithm at work, one sees the small entries bubble to the top of the list—an observation from which the algorithm gets its name.) Use our pseudocode to express a function similar to that in Figure 5.11 for sorting a list using the bubble sort algorithm and write a python program to implement.

Submit your work as a zip/rar file (**asg3-your-name.zip**) to weonline. Include a *pdf* report with your answers to all questions (1-5, and if possible 6 to 7), and the python programs (count.py, euclidian.py, and more).

Note: No late assignments.