CS6502, Assignment 1

Prof. M. Schellekens
UCC, Department of Computer Science

Submit your work via email to: m.schellekens@cs.ucc.ie

Make sure your email-subject reads *exactly* as follows:

CS6502 assignment 1

Attach your python code solutions in one executable Python file ending in .py called:

CS6502assignment1studentID.py

Include your student ID as part of this file name (replace studentID by your actual student ID in the templates given above).
Include your full name as part of the file at the top of the file

Comment your name out with a # sign:

Jimini Cricket

Make sure your code executes correctly and test your code on various examples (this does not need to be done as part of the assignment. Simply submit the relevant code).

Properly comment your code to explain what each part does. Use only the names for programs given in the assignment. Using other names than given one will result in a deduction of marks.

Due date:

Friday, Feb 18, 2022, 5pm.

Total marks:

35

Late assignments won't be accepted/graded.

Exercise 1 [10 marks]
Use the given Python program choose(n,k) which computes binomial coefficients recursively. Adapt the code to produce a new Python program ptriangle(n) that prints the n-the line in Pascale's triangle. The output is given as a list of integers. For instance, a call to ptriangle(6) should return the list $[1, 5, 10, 10, 5, 1]$.

Recall that binomial coefficients, "n choose k" (i.e. the number of ways that k objects can be chosen from n given objects) are defined as follows:

Choose(n,0) = 1
Choose(n,n) = 1
Choose(n,k) = Choose(n-1,k-1) + Choose(n-1,k)

The recursive Python program is given by:

```
def choose(n,k):
    if k == 0:
        return 1
    if k == n:
        return 1
    else:
        return choose(n-1,k-1) + choose(n-1,k)
```

Exercise 2 [15 marks]
Implement the InsertionSort algorithm given below.

```
def InsertionSort(lis):
    listemp = lis.copy()
    for i in range(1,len(listemp)):
        while (i >= 1 and (listemp[i-1] > listemp[i])):
            swap(listemp,i-1,i)
            i = i-1
    return listemp
```

Adapt the code as follows:

1) Define a new function called InsertionSortCount(lis)

InsertionSortCount(lis) returns the number of comparisons (between list elements only) made by the algorithm InsertionSort on the input list.

Note: This is slightly more tricky than it looks.
InsertionSortCount([4,1,3,2]) should return the value 6.

Your first step is to include a counter *count* which is set to 0 at the start.

Then insert a command line count = count + 1 inside the body of the while loop just after the while loop starts. This on its own will not produce the correct answer. The reason is that the condition of the while loop is:

($i >= 1$ **and** ($listemp[i-1] > listemp[i]$))

In cases where listemp[i-1] has a value less than or equal to listemp[i] the while loop will not execute and the counter *count* will not be increased by 1. However the execution of the algorithm did compare listtemp[i-1] with listemp[i] so count *should* be increased by 1.

Address this problem by inserting appropriate if-then-else (conditional) statements where they are needed to catch the cases where the while loop exits and the counter still needs updating.

2) Define a new function called InsertionSortTime(n)
The input is a positive integer n (determining the sizes of the input lists for InsertionSortCount(lis)).

InsertionSortTime(n) runs InsertionSortCount(lis) on all input lists of size n. These input lists form exactly all permutations of the first n positive integers. For instance, for inputs of size $n = 3$ the algorithm InsertionSortCount(lis) must be executed on the 3! = 6 input lists: [1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1].

InsertionSortTime(n) must print the following two strings and associated values:

"The worst-case time of InsertionSort on lists of size n is k"

In the string output: n must be the input number given to InsertionSortCount(n)) and k must be the maximum number of comparisons returned by InsertionSortCount(lis) on the input lists of size n. For instance: "The worst-case time of InsertionSort on lists of size 1 is: 0"

"The average-case time of InsertionSort on lists of size n is k:"

In the string output: n must be the input number given to InsertionSortCount(n)) and k must be the average number of comparisons returned by InsertionSortCount(lis) on the input lists of size n, i.e. the total number of comparisons made by InsertionSort(lis) on all input lists of size n divided by $n!$.

Note: due to the fact that $n!$ grows so fast, you can only test the algorithm up to about size 8 or 9.

Exercise 3 [10 marks]
Write Python code for a recursive version of the BubbleSort algorithm use the function BubbleSortRec(lis). The algorithm should recursively apply the "bubble-up" phase of the algorithm.