University of Aberdeen, Department of Computing Science

CS1028 Practical 8

Programming for Sciences and Engineering

8 November, 2019

Questions:

1. What happens if you run `factorial()` (considered in class) with negative value of n? With a large value, say 35?

2. Compose a recursive program that computes the value of `ln(n!)`.

3. Consider the following recursive function

```
def someFun(n):
    if n <= 0:
        return
    print(n)
    someFun(n-2)
    someFun(n-3)
    print(n)
```

   Write down the sequence of integers written by a call to `someFun(6)`. AFTERWARDS check your answer by running it.

4. Consider the following recursive function

```
def moreFun(n):
    if n <= 0:
        return ''
    return moreFun(n-3) + str(n) + moreFun(n-2) + str(n)
```

What is the return value of `moreFun(6)`?

5. Consider the following recursive function:

```
def mystery(a, b):
    if b == 0:
        return 0
    if b % 2 == 0:
        return mystery(a+a, b//2)
    return mystery(a+a, b//2) + a
```

a) What does `mystery(25, 2)` return?

b) What does `mystery(3, 11)` return?

c) Given positive integers a and b, describe what value `mystery(a, b)` computes.

Advanced questions:

Once you have completed the questions above and feel sufficiently confident, feel free to try these harder questions.

1. Write a recursive function that takes a positive integer as input and returns its prime decomposition as a list of numbers.

   Sounds a lot like the first Advanced question from Practical 7, but this time you are explicitely asked to write a recursive function, i.e, one that is calling itself. Make use of the fact that, if m is a prime divisor of n, it suffices to compute the prime decomposition of n/m.

2. If you still do not feel sufficiently challenged: Permutations. Compose a program that takes a command-line argument n and writes all $n!$ permutations of the n letters starting at a (assume that n is no greater than 26). A permutation of n elements is one of the $n!$ possible orderings of the elements. As an example, when $n = 3$ you should get the following output. Do not worry about the order in which you enumerate them.

```
bca cba cab acb bac abc
```