Algorithms and Data Structures: Week 2 (SOLUTIONS)

Johnson rev. Bell (Michaelmas 2023)

Question 1

(a) The algorithm calculates in p the product of all odd numbers between L and H.

(b) Rewritten with a **for** loop.

Input: integer L, integer H
Output: ?
  integer $p = 1$
  for $x = L$ to H **do**
    if x is odd **then**
      $p = p \times x$
    end if
  end for
  print "result is ", p

Question 2

Recall the algorithm to find the smallest item in a list.

The basic approach is to look at each of the a_i in turn and remember which is the smallest we have seen so far. To achieve this, we create a variable called smallest and set it initially to be a_1. Then we look at the other a_i in turn and whenever we see one that is smaller than smallest we reset smallest to be that value.

To find the second smallest we just create a second variable called second_smallest and use the same basic approach. You may assume there are at least two items in the list to be processed.

It is also acceptable to avoid the use of the **else** clause. You may want to explore the suitability of nested **if** statements were the algorithm to be generalised to return the smallest k items in a list.

Input: integer n, a collection of numbers $A = \{a_0, \ldots, a_{n-1}\}$
Output: the second smallest number in A

```
if a_0 < a_1 then
  smallest = a_0
  second_smallest = a_1
else
  smallest = a_1
  second_smallest = a_0
end if
for a_i in {a_2, ..., a_n} do
  if a_i < smallest then
    second_smallest = smallest
    smallest = a_i
  else
    if a_i < second_smallest then
      second_smallest = a_i
    end if
  end if
end for
print "the second smallest value is ", second_smallest
```

Question 3

Primality can be tested as follows:

Input: integer k (the number to be tested)

```
for i = 2 to square root of k do
  if i divides k then
    print "not prime"
    exit
  end if
end for
print "is prime"
```

Question 4

Let us think through an example. Suppose it is 12.10.2014 (and we live in the universe where the Earth rotates around the Sun in precisely 360 days, a period that in modern society is divided into twelve equal months). What will be the date in, say, 860 days? First note that 860/360 is 2 remainder 140, i.e we are looking to know the date in 2 years and 140 days time. Or, in other words, 140 days after 12.10.2016. Now as 140/30 is 4 remainder 20 this is 4 months and 20 days. Adding the 4 to the months' value would give us 12.14.2016 but, of course, there are only 12 months in a year so in fact we mean 12.2.2017 (notice what we did – subtracted 12 from the month and added 1 to the year). So we want the date 20 days after 12.2.2017 which, adding 20 to the days' value, gives us 32.2.2017 and, similar to before, we note what we actually mean is 2.3.2017 .

Here is the pseudocode:

```
Input: year, month, day, k
Output: date k days later
  year = year + k//360
  k = k % 360
  month = month + k//30
  if month > 12 then
    month = month − 12
    year = year + 1
  end if
  k = k % 30
  day = day + k
  if day > 30 then
    day = day − 30
    month = month + 1
    if month > 12 then
      month = month − 12
      year = year + 1
    end if
  end if
  print "date will be ", day, month, year
```

Some of you may use the % (modulo) operator more liberally than others, so think about how it's justifiable to use subtraction to deal with the day/month overflow situations. You should also attempt to identify patterns of pseudocode reuse in your solutions, but we will stop short of defining subroutines in pseudocode (at least in Part I).