Name:

Introduction to Programming in Python, I Semester, 2014–2015

Quiz 3, 3 September 2014

Answer all questions in the space provided. There are two questions on two pages.
Don't forget to fill your name!

1. Consider the following function.

```
def f(l1,l2):
  for i in range(len(l1)):
    for j in range(len(l2)):
      if l1[i] == l2[j]:
         return False
  else:
    return True
```

   (a) What does `f(l1,l2)` compute?

   Returns `True` *if* `l1` *and* `l2` *are disjoint (no common elements),* `False` *otherwise.*

   (b) What is the worst-case complexity of `f(l1,l2)`?

   $O(m \cdot n)$ *where* m *is* `len(l1)` *and* n *is* `len(l2)`.

(5 marks)

Rough Work:

. . . Question 2 on the back

2. Binary search is how we find words in dictionaries and other sorted lists. To search for k in a list l, compare x with the middle position mid and then inductively search in the first half or second half depending on whether $k < l[mid]$ or $k > l[mid]$.

   Write a recurrence for $T(n)$ and compute the worst-time complexity of binary search.

```
def binarysearch(l,k):  # Search for k in l, l sorted ascending
  if l == []:
    return False
  mid = len(l) // 2
  if l[mid] == k:
    return True
  elif k < l[mid]:
    return (binarysearch(l[:mid],k))
  else:
    return (binarysearch(l[mid+1:],k))
```

(5 marks)

$$
\begin{aligned}
f(1) &= 1 \\
f(n) &= f(\frac{n}{2}) + 1
\end{aligned}
$$

Expanding $f(n)$ we get:

$$
\begin{aligned}
f(n) &= f(\frac{n}{2}) + 1 = f(\frac{n}{4}) + 2 = \cdots = f(\frac{n}{2^k}) + k = \cdots \\
&= f(\frac{n}{2^{\log_2 n}}) + \log_2 n \\
&= f(1) + \log_2 n \\
&= O(\log_2 n)
\end{aligned}
$$

Rough Work: