

Show Your Work! The point values for each problem are in square brackets. There are 35 points possible.

1. We looked at several methods of searching: we searched lists using linear search, we searched sorted lists using binary search, and we searched dictionaries for keys using lookup. Suppose a sorted list and a dictionary store the same information. As an example, the following list and dictionary `d` store the same information:

```
s1 = [1, 5, 9, 17, 31]
d = {(1:0), (5:1), (9:2), (17:3), (31:4)}
```

If we search a sorted list with 1,000,000 elements using linear search and binary search, and we search a large dictionary storing 1,000,000 items using lookup, which of the three searches is likely to be fastest? Which is likely to be slowest? [2 points]

In class we saw that dictionary lookup was fastest and linear search was slowest.

2. Under what circumstances would it be better to use the `xrange` function instead of the `range` function? (Two sentences or less.) [1 point]

`range(n)` builds a list with elements $0, 1, 2, \dots, n - 1$. If n is large, this is going to use a lot of memory. So if you're implementing a loop such as

```
for i in range(n):
    . . .
```

and never using the list — just the values of `i` — then `xrange` is a better choice since it just assigns the values $0, 1, \dots, n - 1$ to `i` without storing them all at once. So it will use considerably less memory.

3. In two sentences or less explain why the internet can be viewed as a directed graph. [1 point]

4. Find the output of each of the following print statements: [2]

(a)

```
s = "hello"
t = "b" + s[1:5] + "w"
print t
```

(b)

```
l = [1,2,3,5]
del l[2]
print l
```

(a) bellow

(b) [1, 2, 5]

5. Find the output of the following Python program. [3]

```
def foo(s, l1, l2, d):
    s = "Hello!" # Assignment to parameter doesn't change arguments
    l1 = ['b', 'o', 'g'] # Ditto
    l2[0] = 'b' # Assignment to element of mutable parameter changes
                # argument
    d['s'] = 'h' # Ditto

s = "Greetings!"
l1 = ['l', 'o', 'g']
l2 = ['c', 'a', 't']
d = {'p': 'o', 's': 't', 'e': 'r'}

foo(s, l1, l2, d)

print s
print l1
print l2
print d
```

The output is

```
Greetings!
['l', 'o', 'g']
['b', 'a', 't']
{'p': 'o', 's': 'h', 'e': 'r'}
```

6. Find the output of the following Python program. [3]

```
def s(l, i, j):
    """Swap the elements in locations i and j in the list l"""
    t = l[i]
    l[i] = l[j]
    l[j] = t

l = [1, 3, 6, 0, 9]
s(l, 1, 3)
print l
```

The output is:

```
[1, 0, 6, 3, 9]
```

7. Find the output of the following Python program. [3]

```
def cs(l):
    n = len(l)
    ls = [l[n-1]] # ls has one element, the last element in l
    for i in range(0, n-1):
        # Append the elements in locations 0, 1, . . . , n-2 to ls
        ls.append(l[i])
    return ls

l = [4, 2, 9, 3, 5]
ls = cs(l)
print ls
```

The output is:

```
[5, 4, 2, 9, 3]
```

8. Find the output of the following Python program. [3]

```
def wi(s,c):
    """s: a string whose characters are sorted in increasing order
       c: a single character
       Find the smallest subscript i such that c < s[i]
       i.e., c alphabetically precedes s[i]"""
    for i in range(len(s)):
        if c < s[i]:
            return i
    return len(s)

s = "bdqsvx"
p = wi(s, 'u')
print p
```

The output is:

```
4
```

9. What's the output of the following Python program if the input is

(a) 2

(b) 4

[4 points]

```
def recurse(n):
    """Compute 2**n using recursion"""
    if n == 0:
        result = 1
    else:
        result = 2*recurse(n-1)
    print "n =", n, ", result =", result
    return result

n = int(raw_input("What's n?\n "))
result = recurse(n)
print "result =", result
```

Output:

a) What's n?
2
n = 0 , result = 1
n = 1 , result = 2
n = 2 , result = 4
result = 4

b) What's n?
4
n = 0 , result = 1
n = 1 , result = 2
n = 2 , result = 4
n = 3 , result = 8
n = 4 , result = 16
result = 16

10. Find the output of the following Python program. [3]

```
def build_matrix(n):
    """Create a matrix (or list of lists) with n rows and
       n columns. The element in row i and row j is i+j"""
    matrix = []
    for i in range(n):
        row = []
        for j in range(n):
            val = i+j
            row.append(val)
        matrix.append(row)
    return matrix

def print_matrix(mat, n):
    """Print a matrix with n rows and n columns with one row
       on each line of output"""
    for i in range(n):
        for j in range(n):
            print mat[i][j],
        print

n = 3
mat = build_matrix(n)
print "The matrix is"
print_matrix(mat, n)
```

Output:

```
The matrix is
0 1 2
1 2 3
2 3 4
```

11. A Python program is making use of a dictionary whose keys are ints and whose values are lists of ints. For example,

```
d = {0: [1,2,3], 3: [2,3,4,0], -1: [-1,3,4]}
```

Write a Python function `sum_lists` that takes a dictionary argument such as `d` and returns a new dictionary with the same keys, but the values are the sums of the elements in the lists. For example, if `d` is the argument, `sum_lists` should return something like

```
dsl = {0:6, 3:9, -1:6}
```

A header for the function is

```
def sum_lists(d):
```

[5 points]

```
def sum_lists(d):
    dsl = dict()
    for key in d:
        sum = 0
        for val in d[key]:
            sum = sum + val
        dsl[key] = sum
    return dsl
```

12. Write a Python function `max_loc`. It's passed a list of ints. It returns a list whose first element is the largest item in the list, and whose remaining elements are the locations in the list where the largest item is stored. For example, suppose the list is

```
l = [5, 2, 1, 5, 3, 4, 5]
```

Then the function should return the list

```
r = [5, 0, 3, 6]
```

A header for the function is

```
def max_loc(l):
```

[5 points]

```
def max_loc(l):
    biggest = l[0]
    locs = [0]
    for i in range(1, len(l)):
        if l[i] > biggest:
            biggest = l[i]
            locs = [i]
        elif l[i] == biggest:
            locs.append(i)
    return [biggest]+locs
```