Here is the code from the slides, with the corresponding primitive operations:

```python
def lookup(str_, list_):
    for i in range(len(list_)):
        if str_ == list_[i]:
            return i
    return -1
```

The worse-case running time for a list of length $n$ is $9n + 1$.

1. a) What is the total worst-case running time of the following code fragment expressed in terms of $n$?

```python
n = len(vals)
sum = 0
for i in range(n):
    sum = sum + vals[i]
```

b) Why can we ignore the constants and lower-order terms?
2. Express the worst case complexity of the code in problem 1 in terms of big-O notation.

3. For each code fragment, state its big-O complexity.
   a)

   ```python
   for x in numlist1:
       for y in numlist2:
           print(x + y)
   ```
   b)

   ```python
   if i%2 == 0:
       for j in range(0,len(numlist),2):
           print(numlist[j])
   else:
       for j in range(len(numlist)):
           print(numlist[j])
   ```
   c)

   ```python
   x = 0
   n = 50
   for i in range(n):
       x = x + i
   ```