Name: Kirti Joyejob #6046002

COMP 352 Assignment#2

Q1.1

(i)  
Algorithm findPair(x,y)

Input n elements in linked list

Output pair such that x<y where x is node1 and y is node1.next

x← head

while pointer is not null

if (x<x.next())

return (x,x.next())

x←x.next()

else

x=x.next()

(ii) x is a temporary variable in which the current head is stored in. I compared that value with the one following it, and if the current value is smaller than the one next to it, then that pair is output, else x stores the second value. The loop goes on and on again until pointer head points at null, meaning that there is no value left in the linked list to compare with.

(iii) Complexity of my algorithm is O(n). The loop runs in O(n) times if isn't smaller than x.next(), which is the worst case here. The rest of the code can be ignored as it runs in constant time.

Q.2.

(i)

Algorithm outputPair

Input n elements from linked list to queue Q

Output pair such that x<y where x is node1 and y is node1.next

While pointer != null

enqueue(head)

head← head+1

While head is not null do

a←dequeue()

head←head+1

b←head  
 if a<b

return (a,b)

endif

(N.B: the assignment has not mentioned anything about storing the elements of the linked list somewhere else so that they are not lost).

(ii) I enqueued all the elements from the non-sorted linked list to a queue, then each time i dequeue, the head is points to the next value. If the element that has been dequeued is less than the element at which the head is now pointing at, then that pair is output, else each element is dequeued one by one and compared with its consequent one.

(iii) It's a wastage of space since each time an element is removed from the queue, the head moves to the next element, hence making the previous one null. It sure takes a twice as much time since first all the n elements have to be added to the queue, which is O(n). Then each element is removed from the queue and compared with the consequent one which is again O(n).

The complexity is linear, i.e O(n). The algorithm outputPair runs in 2n times.

Q.3.(i)  
(a) *f*(*n*) = *n4* – 3000000 *n* log *n* + 100000

Complexity: O(n4). Complexity is a polynomial with n to the power of 4.

(b) *f*(*n*) = 2*n* – 2*n2*

Complexity: O(n). Linear

(c) *f*(*n*) = *n6* + 3000000000 /*n*

Complexity: O(n6). Polynomial with n to the power of 6.

(d) *f*(*n*) = ((*n6* + 1) log*n* + log(*n* + 3) – (*n* + 1) + 2) / *n3*

= ((n6-1)logn)/n3 + (log(n+3))/n3 - (n+1)/n3 + 2/n3  
 = (n6logn - logn)/n3 + (log(n+3))/n3 - (n+1)/n3 + 2/n3

= n3logn - logn/n3 + (log(n+3))/n3 - (n+1)/n3 + 2/n3  
  
Complexity: O(n3). Cubic

(e) *f*(*n*) = *n4* – 3/*n6*

Complexity: O(n4). Complexity is a polynomial with n to the power of 4, since n6 is in the denomenator which when n tends to positive infinite, that value tends to 0.

(N.B: Since it is said "Estimate" in the assignment, it is understood that no calculation is to be made)

(ii)

(a) f(n) is O(n4) and Ω(n).

(b) f(n) is O(n) and Ω(n4). Therefore, f(n) is Θ(n4).