Problem Set 1

Name: Zhang Wenhao

Collaborators:

Ω

Problem 1-1.

(a) we can simplify thest to $f_1 = \Theta(n \log n)$, $f_2 = \Theta((\log n)^n)$, $f_3 = \Theta(\log(n^6 006))f_4 = \Theta(6006 \log n)$, $f_5 = \theta(\log\log(6006n))$, so the sequence is $(f_5, f_3, f_4, f_1, f_2)$.
correct

(b) we take logrithm to every functions, then $f_1 = n \log 2$, $f_2 = n \log 6006$, $f_3 = 6006^(n) \log 2$, $f_4 = 2^n \log 6006$, $f_5 = n^2 \log 6006$, so the sequence of functions is $(f_1, f_2, f_5, f_4, f_3)$.
correct

(c) $(\{f_2, f_5\}, f_4, f_1, f_3)$
correct

(d) take the logrithm to functions,$(\{f_5, f_2\}, f_1, f_3, f_4)$
correct

Problem 1-2.

(a)

```
    # problem1-2 a
    def reverse(D, i, k):
        if k < 2:
            return
        for j in range(k/2):
            x1 = D.delete_at(i-1+j)
            x2 = D.delete_at(i+k-1-j)
            D.insert_at(i-1+j,x2)
            D.insert_at(i+k-1-j,x1)
    # answer: use recursive
    def reverse(D, i, k):
        if k < 2:
            return # base case
        x2 = D.delete_at(i+k-1)
        x1 = D.delete_at(i)
        D.insert_at(i, x2)
        D.insert_at(i + k - 1, x1)
        reverse(D, i+1, k - 2)
```

(b)

```
# problem1-2 b
def move(D, i, k, j):
    if k < 1:   # base case
        return
    x1 = D.delete_at(i)
    if j > i:
        j - 1
    D.insert_at(j,x1)
    if i > j:
        i = i + 1
    move(D, i, k-1, j)
```

Problem 1-3. skip

Problem 1-4.

(a) 1.insert_ first(x): Construct a new node as a storing x. If the double link D is empty,then link both D.head/D.tail to a.Otherwise set b equal to L.head,then set a's next pointer point to b and b.pre pointer point to a,then set L.head point to a.

2.insert_ last(x): Same as insert_ first, Construct a new node as a sotring x, if the double link D is empty, set D.head and D.tail point to a. otherwise set b equal to L.tail, a.pre point to b, b.next point to a, and L.tail ponit to a.

3.delete_ first(): First, if linklist is empty, reutrn.If L.head.next equal None, set L.head equal None, return. Otherwise set a equal to D.head.next, b equal to D.head, set b.next point to None, D.head point to a.

4.delet_ last(): First, if linklist is empty, reutrn. If L.Tail.pre equal None, set L.Tail. equal None, return. Otherwise set a equal to D.Tail.pre, b equal to D.Tail, set b.pre point to None, D.tail point to a.

(b) First, constuct a new empty double Link list L_1, set L1.head point to x1, L1.tail point to x2, then L1 is the double Link list we need to return.
Now, remove x1 to x2 from L:
if x1 is L's head and x2 is not L's tail, set L.head point to x2.next, x2.next.pre point to None.
if x1 is L's head and x2 is L's tail, set L.head and L.tail all point to None
if x1 is not L's head and x2 is L's tail, set x1.pre.next point to Null, and L's tail point to x1.pre.
otherwise set x1.pre.next point to x2.next & x2.next.pre point to x1.pre.

(c) if L2.head = None, it means L2 is empty, reutrn.
if x.next = None, x.next = L2.head, L2.head.pre = x, then set L2.head & L2.tail all point to None.
if x.next != None, set p = x.next, x.next = L2.head, L2.head.pre = x, L2.head point to None. p.pre = L.tail, L.tail.next = p, then L.tail point to None.

(d) Submit your implementation to `alg.mit.edu`.