CSC 212 Midterm 1 - Fall 2017

College of Computer and Information Sciences, King Saud University
Exam Duration: 90 Minutes

9/11/2017

Question 1 [30 points]

1. Choose the correct frequency for every line and the total big O:

```
1 for (int i = 0; i < 2 * n; i++) {
2         System.out.println(i);
3         for (int j = 0; j < i; j++)
4                 System.out.println(j);}
```

	a	b	c	d	e
1	$n+2$	$2n$	$2n-1$	$2n+1$	$2n+2$
2	$2n$	$2n+2$	$2n-1$	$n+1$	$2n+1$
3	$2n^2+1$	$n(n+1)/2$	$n^2(n^2+1)/2$	$n^2(n+1)/2$	$2n^2+n$
4	$n^2(n)/2$	$2n^2-n$	$n^2(n^2)/2$	$n(n-1)/2$	$2n^2$
O	1	n	n^2	n^3	n^4

2. Choose the correct frequency from the line 2 to 7 **in the worst case** and the total big O:

```
1 public void func(int[] A, int n) {
2         for (int i = 0; i < n; i++)
3                 if (A[i] % 2 == 0)
4                         for (int j = 0; j < n; j++)
5                                 A[i] = A[i] + A[j];
6                 else    for (int j = 1; j < n; j *= 2)
7                                 A[i] = A[i] + A[j];
8 }
```

	a	b	c	d	e
2	0	n	$n-1$	$n+1$	$n+2$
3	$n+1$	n	$n-1$	0	$n/2$
4	0	n^2	$2n+1$	n^2+n	$n^2/2+1$
5	n^2	n^2+1	$2n$	$n^2/2-1$	$n-1$
6	$\log(n)+n$	$2n^2+n$	$n\log(n)$	0	$2n^2\log(n)+1$
7	$\log(n)$	$2n^2$	0	$2n^2\log(n)$	$n\log(n)+n$
O	n^3	n	n^2	$n\log(n)$	$n^2\log(n)$

Question 2 [35 points]

1. Write the method **public static <T> void** `removeAllX(List<T> l, T[] X, int n)`, user of the ADT List, which removes all elements of X from the list l, where X has n elements.

   Example 2.1. *If* $l : D, A, B, E, C, A, B, D$, *and* $X = \{A, C, B\}$, *then calling* `removeAllX(l, X, 3)` *results in* $l : D, E, D$.

2. Write the method **public static <T> void** `reverse(Queue<T> q)` (user of ADT) that takes as input a queue and reverse its order. You must use a double linked list.

   Example 2.2. *If* $q : 1, 4, 8, 10, 13, 14$, *calling* `reverse(q)` *results in* $q : 14, 13, 10, 8, 4, 1$

Question 3 [35 points]

1. Write the method **public void** `moveToEnd(int p)`, member of the class `LinkedList` that moves all elements that are before the position p to the end of the list. **The first element is set as the current element**. Assume that p is a valid position in the linked list ($0 < p < n$, where n is the length of the list) and that the first element is at position 0. **Do not call any methods and do not use any auxiliary data structures**. Assume that the list is not empty.

   Example 3.1. *If* $l : A, C, G, R, H$, *then calling* `moveToEnd(2)` *results in* $l : G, R, H, A, C$.

2. Write the method **private void** `shiftSeg(Node<T> startN, Node<T> endN, int n)`, member of the class `DoubleLinkedList`, which shifts the segment that starts at node `startN` and ends at node `endN` by `n` nodes to the right. Assume that the segment is a valid segment in the double linked list and $n \geq 0$. If n exceeds the number of nodes that comes after *endN*, the segment is inserted at the end of the double linked list. The position of the `current` must not change. **Do not call any methods and do not use any auxiliary data structures**.

   Example 3.2. *Given the double linked list* $A \leftrightarrow \boldsymbol{B} \leftrightarrow \boldsymbol{C} \leftrightarrow \boldsymbol{D} \leftrightarrow E \leftrightarrow F$, `startN` *points to* B *and* `endN` *points to* D, *calling* `shiftSeg(startN, endN, 1)` *results in* $A \leftrightarrow E \leftrightarrow \boldsymbol{B} \leftrightarrow \boldsymbol{C} \leftrightarrow \boldsymbol{D} \leftrightarrow F$. *Calling* `shiftSeg(startN, endN, 10)` *on the initial list results in* $A \leftrightarrow E \leftrightarrow F \leftrightarrow \boldsymbol{B} \leftrightarrow \boldsymbol{C} \leftrightarrow \boldsymbol{D}$.

Specification of ADT List

- findFirst ( ): **requires**: list L is not empty. **input**: none. **results**: first element set as the current element. **output**: none.
- findNext ( ): **requires**: list L is not empty. Current is not last. **input**: none. **results**: element following the current element is made current. **output**: none.
- retrieve (Type e): **requires**: list L is not empty. **input**: none. **results**: current element is copied into e. **output**: element e.
- update (Type e): **requires**: list L is not empty. **input**: e. **results**: the element e is copied into the current node. **output**: none.
- insert (Type e): **requires**: list L is not full. **input**: e. **results**: a new node containing element e is created and inserted after the current element in the list. The new element e is made the current element. If the list is empty e is also made the head element. **output**: none.
- remove ( ): **requires**: list L is not empty. **input**: none. **results**: the current element is removed. If L is empty, current will point to null. If the next element exists, it is made current, else the first element is made current. **output**: none.
- full (boolean flag): **requires**: none. **input**: none. **results**: if the number of elements in L has reached the maximum then flag is set to true otherwise false. **output**: flag.
- empty (boolean flag): **requires**: none. **input**: none. **results**: if the number of elements in L is zero, then flag is set to true otherwise false. **output**: flag.
- last (boolean flag): **requires**: L is not empty. **input**: none. **results**: if the last element is the current element then flag is set to true otherwise false. **output**: flag.

Specification of ADT Double Linked List

In addition to the operations of the ADT List:

- findPrevious ( ): **requires**: list L is not empty. Current is not first. **input**: none. **results**: element preceding the current element is made current. **output**: none.
- first (boolean flag): **requires**: L is not empty. **input**: none. **results**: if the first element is the current element then flag is set to true otherwise false. **output**: flag.

Specification of ADT Queue

- enqueue (Type e): **requires**: Queue Q is not full. **input**: Type e. **results**: Element e is added to the queue at its tail. **output**: none.
- serve (Type e): **requires**: Queue Q is not empty. **input**: none. **results**: the element at the head of Q is removed and its value assigned to e. **output**: Type e.
- length (int length): **requires**: none. **input**: none. **results**: The number of elements in the Queue Q is returned. **output**: length.
- full (boolean flag): **requires**: none. **input**: none. **results**: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.