		ID	
CS212 (MT1)	Name	ID	
C3212 (IVIII)	Name		



King Saud University College of Computer and Information Sciences Department of Computer Science

CS 212 Midterm 1 – 1st Semester 2011

Question 1 (30 Marks)

Find the total number of primitive operations and Big Oh notation of the following methods.

a)

	Statements	S/E	Frequency	Total
1	void sumFirstTen()	ľ	1	1
2	\ 	-	-	+
3	int total=0;	i	l	1
4		1	10	1
5	for(int count=1;count<=10;count++) {	=	#	+
6	total=total+count;	1	10	(C
7	System.out.println(" count "+ count + " total "+ total);	Ì	10	10
8	System.out.printin(count + co	-	**	
9	}	_		
	Total Operations →		13	3
	Big Oh →	0(/)	

		ID
CS212 (MT1)	Name	

) 		S/E	Frequency	Total
	Statements		,	
1	int crossSum(int n)	1	1	'
2	{			
3	int sum=0;	((1
4	for(int r=1;r<=n;r++)	1	N-1	n-1
5	{			
6	for(int c=1;c<=n;c++)	n.	in	N2
7				
8	{	n	n	N2
9	sum=r+c; System.out.println("r="+r+"c="+c+"sum="+sum);	in	N	NT
10	System.out.printin(1= +1+ c= 1c. sum //			
11	}			
	_] }			
12	return sum;	\		
13	$oldsymbol{oldsymbol{\sqcup}}$		_	
-	Total Operations →			
-	Big Oh → O(へ)			

		i	
CS212 (MT1)	Name	ID	
' '			

Question 2 (30 Marks)

Following is the specification of ADT List.

Operations: We assume all operations operate on a list L.

Method FindFirst ()

requires: list L is not empty. input: none results: first element set as the current element. output: none.

Method FindNext ()

requires: list L is not empty. Current is not last. input: none

results: element following the current element is made the current element. output: none.

Method FindPrevious ()

requires: list L is not empty. Current is not Head. input: none

results: element before the current element is made the current element. output: none.

Method Retrieve (Te)

requires: list L is not empty. input: none

results: current element is copied into e. output: element e.

Method Update (Te).

requires: list L is not empty. input: e.

results: the element e is copied into the current node. output: none.

Method Insert (T 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.

Method Remove ()

requires: list L is not empty. input: none

results: the current element is removed from the list. If the resulting list is empty current is set to NULL. If successor of the deleted element exists it is made the new current element otherwise first element is made the new current element. **output**: none.

Method Full (boolean flag)

input: none. returns: if the number of elements in L has reached the maximum number allowed then flag is set to true otherwise false. output: flag.

Method Empty (boolean flag).

input: none. results: if the number of elements in L is zero, then flag is set to true otherwise false.

Output: flag.

Method Last (boolean flag).

input: none. **requires**: L is not empty. **Results**: if the last element is the current element then flag is set to true otherwise false. **Output**: flag

	1	
CC242 (NATA) Name	ID	
CS212 (MT1) Name		

The ADT List is implemented as a linked list in the following code . complete the body of the methods implementing the ADT List operations.

```
Class Node
      public class Node<T> extends Object {
         public T data;
         public Node<T> next;
         public Node () {
            data = null; next = null; }
         public Node (T val) {
            data = val; next = null; }
Class LinkList
      public class LinkList<T> extends Object {
          private Node<T> head;
          private Node<T> current;
          public LinkList () {
             head = current = null; }
    public boolean empty () {
       return head = = null;
    public boolean last () {
         return cornect. Next == nuil;
    public boolean full () {
     return false;
     public void findfirst () {
         Convert = head ;
```

public void findnext () {	rieve () { correct. Nort; findPrevious () { prevat; prevat; in the prevat	2 (MT1) Name	ID
public T retrieve () { retern convoid Data, } public void findPrevious () { valuation - Need; while (i) Next; = convoid) i's i Meet; convoid it? } public void update (T val) { convoid Data = vaid } public void insert (T val) { See stick } public void remove () {	rieve () { correct. Nort; findPrevious () { prevat; prevat; in the prevat		
public T retrieve () { retern curred. Data, } public void findPrevious () { Notation - Medic white (P that != curred) P=P Next; curred. Palice vail } public void update (T val) { curred. Data = vail } public void insert (T val) { See stick. } public void remove () {	rieve () { converte. Dota, findPrevious () { predistrictionset; results and update (T val) { politered insert (T val) { a slick. d remove () {	<pre>public void findnext () {</pre>	
public void findPrevious () { public void findPrevious () { public () Mark != convex) public void update (T val) { convert Pri } public void insert (T val) { See stick } public void remove () {	findPrevious () {	correct = convect . Most	
public void findPrevious () { public void findPrevious () { public () Mark != convex) public void update (T val) { convert Pri } public void insert (T val) { See stick } public void remove () {	findPrevious () {	}	
public void findPrevious () { Node (T) Predict white (T) West != convex!) Per Meet; convert : I': } public void update (T val) { convert Data = vect } public void insert (T val) { See Slick } public void remove () {	findPrevious () {	public T retrieve () {	
public void findPrevious () { Node (T) Predict white (T) West != convex!) Per Meet; convert : I': } public void update (T val) { convert Data = vect } public void insert (T val) { See Slick } public void remove () {	findPrevious () {	return current. Data,	
public void insert (T val) { See stick } public void remove () {	update (T val) { Drite val d remove () {		
public void insert (T val) { See stick } public void remove () {	update (T val) { Drite val d remove () {	} public void findPrevious () {	
public void update (T val) { correct - Drite = vaid } public void insert (T val) { See slick } public void remove () {	update (T val) { Date val d insert (T val) { d remove () {	public void ! = Messo à	
public void update (T val) { correct Palice vai } public void insert (T val) { See state } public void remove () {	update (T val) { Data val d insert (T val) { d remove () {	(A short in convect)	
public void update (T val) {	update (T val) { Drize vai d insert (T val) { d remove () {	white Critical	
public void update (T val) {	update (T val) { Drive val d insert (T val) { d remove () {		
public void insert (T val) { See stick } public void remove () {	d insert (T val) { d remove () {	(Interpreted 1)	
public void insert (T val) { See stick } public void remove () {	d insert (T val) { d remove () {		
public void insert (T val) { See stick } public void remove () {	d insert (T val) { e stick d remove () {	} public void update (T val) {	
public void insert (T val) { See slick } public void remove () {	d insert (T val) { d insert (T val) { d remove () {	public volumpus (
public void insert (T val) { See slick } public void remove () {	d insert (T val) { e slick d remove () {	annat Date val	
} public void remove () {	ייי אונגיב. d remove () {		
} public void remove () {	ייי אונגיב. d remove () {		
} public void remove () {	ייי אונגיב. d remove () {	}	
} public void remove () {	d remove () {	<pre>public void insert (T val) {</pre>	
} public void remove () {	d remove () {		
} public void remove () {	d remove () {	Sag Sticke	
		}	
See Sinke.	Title.	public void remove () {	
See Side.	Title.		
Size Sinke.	Birthe.		
		See Sinte.	

,	
	ID
CS212 (MT1) Name	

Question 3 (40 Marks)

The following is the specification for the Queue ADT along with the implementation.

Element	Specification
public class Node <t> {</t>	
public T data;	
<pre>public Node<t> next;</t></pre>	
public Node () {	The elements are of a variable T <t>. In a linked</t>
<pre>data = null; next = null; }</pre>	implementation elements are placed in nodes.
<pre>public Node (T val) {</pre>	
<pre>data = val; next = null; }</pre>	
}	
ADT Queue (Linked Implementation)	
<pre>public class LinkQueue <t> { private Node<t> head, tail; private int size;</t></t></pre>	
<pre>/** Creates a new instance of LinkQueue */ public LinkQueue() { head = tail = null; size = 0; }</pre>	
Operation	
<pre>public void enqueue (T e) { if (tail == null){ head = tail = new</pre>	Method Enqueue (T e)
Node(e);	requires: Queue Q is not full. input: T e.
<pre>else { tail.next = new Node(e); tail = tail.next; }</pre>	results: Element e is added to the queue at its tail. output: none.
size++;	
<pre>public T serve() {</pre>	Method Serve (T e)
T x; x = head.data;	requires: Queue Q is not empty.
head = head.next; size; if (size == 0) tail = null;	results: the element at the head of Q is removed and its value assigned to e. output: T e.

return x; } }	
<pre>public int length (){ return size;}</pre>	Method Length (int length) results: The number of element in the Queue Q is returned. output: length.
<pre>public boolean full() { return false;}</pre>	Method Full (boolean flag). results: If Q is full then flag is set to true, otherwise flag is set to false. output: flag.
<pre>public boolean empty() { return size = = 0;}</pre>	Method empty (boolean flag). results: If Q is empty then flag is set to true, otherwise flag is set to false. output: flag.

(a) Using the ADT Queue Operations:

1. Write a method, called enquiry, to return the first element of the queue without changing the queue head.

```
Riblic T enquiry (Linksither D)

T x = Diservel)

Dierque(x)

In (intizo, i < Qienthol-1, i++)

Qienque (Diservel);

return x;
```

CS212 (MT1) Name	ID
2. Find the B	ig-Oh notation for the method enquiry.
	O(n)

(b) 1. Write a **Member** method (Operation) in the ADT Queue called enquiry that will return the first element of the queue without changing the queue head.

```
rable Tenquery ()

return Head Date;
```

2. Find the Big-Oh notation for the operation enquiry.

```
0(1)
```

(c) Following is the implementation of ADT ListQueue, which is a queue implemented using a List (see specification in Question 2)

```
public class ListQueue <T> {
    private LinkList <T> Q;
    private int size;
public ListQueue() {
        Q=new LinkList <T>();
        size = 0;}
public void Enqueue (T e){}
public T Serve (){}
```

1. Write the body of the Method Enqueue.

public Void Enquere (Te) {

while (O.405+()) Q. InelMext();

Q. Insert (e);

Size ++;
}

2. Write the body of the Method Serve.

Public T Serve () {

D. find first()

X = D. remove () 7

Size - - 2

return X ;

}