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()	1	1	1
2	{	~~—~~	~~—~~	~~—~~
3	int total=0;	1	1	1
4	for(int count=1;count<=10;count++)	1	10	1
5	{	~~—~~	~~—~~	~~—~~
6	total=total+count;	1	10	10
7	System.out.println(" count "+ count + " total "+ total);	1	10	10
8	}	~~—~~	~~—~~	—
9	}	—	—	—
	Total Operations →	13		
	Big Oh →	O(1)		

b)

	Statements	S/E	Frequency	Total
1	int crossSum(int n)	1	1	1
2	{	—	—	—
3	int sum=0;	1	1	1
4	for(int r=1;r<=n;r++)	1	$n-1$	$n-1$
5	{	—	—	—
6	for(int c=1;c<=n;c++)	$n-1$	n	n^2-n
7	{	—	—	—
8	sum=r+c;	n	n	n^2
9	System.out.println("r="+r+"c="+c+"sum="+sum);	n	n	n^2
10	}	—	—	—
11	}	—	—	—
12	return sum;	1	1	1
13	}	—	—	—
	Total Operations →			
	Big Oh →	$O(n^2)$		

Question 2 (30 Marks)

Following is the specification of ADT List.

<u>Operations:</u> 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 (T e)
requires: list L is not empty. **input**: none
results: current element is copied into e. **output**: element e.

Method Update (T e).
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

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    current.Next == null;

}
```

```
public boolean full () {



return false;
}
```

```
public void findfirst () {

   ~~head =~~
     current = head;
}
```

```
public void findnext () {
    current = current.Next
}
```

```
public T retrieve () {
    return current.Data;
}
```

```
public void findPrevious () {
    Node<T> P = Head;
    while (P.Next != current)
        P = P.Next;
    current = P;
}
```

```
public void update (T val) {
    current.Data = val
}
```

```
public void insert (T val) {
        See slide
}
```

```
public void remove () {
        See Slide.
}
```

Question 3 (40 Marks)

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

Element	**Specification**

public class Node<T> {
public T data;
public Node<T> next;
public Node () {
data = null; next = null; }
public Node (T val) {
data = val; next = null; }
}
	The elements are of a variable T <T>. In a linked implementation elements are placed in nodes.

ADT Queue (Linked Implementation)	

public class LinkQueue <T> {
private Node<T> head, tail;
private int size;
/** Creates a new instance of
LinkQueue */
public LinkQueue() {
head = tail = null;
size = 0;
}	

Operation	

public void enqueue (T e) {
if (tail == null){
head = tail = new
Node(e);
}
else {
tail.next = new Node(e);
tail = tail.next;
}
size++;
}
	Method Enqueue (T e)
	requires: Queue Q is not full. **input:** T e.
	results: Element e is added to the queue at its tail. **output:** none.

public T serve() {
T x;
x = head.data;
head = head.next;
size--;
if (size == 0)
tail = null;
	Method Serve (T e)
	requires: Queue Q is not empty.
	results: the element at the head of Q is removed and its value assigned to e. **output**: T e.

return x;
}
}	

public int length (){
return size;}
	Method Length (int length)
	results: The number of element in the Queue Q is returned. **output**: length.

public boolean full() {
return false;}
	Method Full (boolean flag).
	results: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.

public boolean empty() {
return size = = 0;}
	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.

```
Public T enquiry (LinkedQue Q)
{
    T x = Q.Serve()
    Q.enque(x)
    for (int i = 0, i < Q.length()-1, i++)
        Q.enque (Q.Serve());
    return x;

}
```

2. Find the Big-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 .

```
Public T enquiry ( )
{
  return Head . Data ,
}
```

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

O(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 Enqueue(T e){
    while (!Q.last()) Q.findNext();
     Q.insert(e);
     Size ++ ;
{
```

2. Write the body of the Method Serve.

```
Public T Serve (){

    Q.findfirst()
   ~~T~~ X = Q.remove();
    Size - - ;
    return X ;
 }
```